IEICE TRANSACTIONS

on Fundamentals of Electronics, Communications and Computer Sciences

DOI:10.1587/transfun.2024CIP0010

Publicized:2024/09/25

A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER *Special Section on Cryptography and Information Security*

Practical Randomness Effects on Physical Security in Second-Order Threshold Implementation of AES*

Maki TSUKAHARA[†a)], ***Nonmember***, **Yusaku HARADA**[†b)], ***Student Member***, **Haruka HIRATA**[†], ***Nonmember***, **Daiki MIYAHARA**[†], **Yang LI**[†], **Yuko HARA-AZUMI**[††], ***and*** **Kazuo SAKIYAMA**[†c)], ***Members***

SUMMARY Physical attacks against cryptographic hardware have become a major threat. For example, side-channel attacks (SCAs) exploit information leakage from power consumption and electromagnetic radiation during encryption to recover secret keys. We recognize them as a powerful threat because the attackers can conduct them using relatively inexpensive equipment. Thus, embedded systems based on cryptographic hardware need to be secure against SCAs. Threshold Implementation (TI) is widely studied as an effective countermeasure against SCAs. Each sensitive intermediate value is divided into multiple values called shares using random bits, and each share is performed to realize the cryptographic algorithm securely. TI requires three important properties for secure computation: correctness, non-completeness, and uniformity. Note that non-linear operation, e.g., AES S-box, cannot preserve perfect uniformity. Compensating for the lack of uniformity, the intermediate values must be re-masked using a large amount of fresh random numbers, called refreshing. Therefore, it is necessary to use random numbers in random number generators (RNGs) to implement TI, but the security requirements for randomness in such RNGs are not yet well-discussed. In this paper, we investigate the impact of practical randomness on security against SCAs. More specifically, we implement AES hardware protected by second-order TI on an FPGA to evaluate the security in cases where the random number used for dividing the secret value into shares is fixed or random. Furthermore, we also explore information leakage in the case where randomized or fixed seed values are sent to the RNG used in refreshing or where the frequency of random number updates is reduced. Based on these results, we discuss practical randomness suitable for TI-based hardware countermeasures.

key words: *AES, side-channel analysis, masking, TI, randomness, M&M*

1. Introduction

In a digital society, as information technology evolves and becomes increasingly prevalent, we communicate large amounts of data in various situations in daily life. For example, personal information and confidential data are constantly being sent and received through the usage of credit cards and smartphones. Security is an extremely important factor in such transactions, and cryptography plays an important role in information security functions such as confidential communication and authentication. Advanced Encryption Standard (AES) [2], the most widely used symmetric-key cipher in use today, provides high security and ensures data confidentiality. Over the years, this cipher scheme has withstood most cryptanalysis attacks. However, attacks through physical access to cryptographic hardware devices such as IC cards and credit cards, a.k.a. physical attacks, can invalidate the data confidentiality and disclose secret data.

Examples of physical attacks include side-channel attacks (SCAs) that predict the secret key for encryption by observing physical information generated as a side effect of the encryption calculation. One of the principal SCAs is a correlation power analysis (CPA) that extends the analysis method of differential power analysis (DPA) [3] proposed by Brier *et al.* in 2004 [4]. The attacker estimates the leakage using the guessed key and power model, e.g., Hamming distance model, which approximates the power consumption from the number of transition bits in a target register. Then the attacker tries to identify the secret key by calculating the correlation coefficients between the predicted leakage and the actually measured data. Previous work on electromagnetic analysis (EMA) that uses electromagnetic radiation instead of power consumption has also been intensively researched [5–7].

Masking has proven to be one of the best approaches, as shown by several experimental results and theoretical observations about the countermeasures against SCAs. The fundamental idea behind the masking scheme is to randomize intermediate values dependent on the cipher key and to amplify noise. The best-known approach in masking is Boolean masking [8], where the confidential value is split into multiple shares, where the XOR of these shares maintains the original value.

Threshold Implementations (TI), proposed by Nikova *et al.* [9] in 2006, is a countermeasure based on multiparty computation (MPC). It addresses the above-mentioned problem theoretically. This approach has been extended to higher orders in [10]. In this technique, the input shares are defined as $td + 1$, where t is the algebraic order of the function and d is the order of the security. As the number of shares increases the cost of implementation, an important issue for TI is to configure the function with the smallest number of shares. It is discussed that $d + 1$ input shares are sufficient when d-th order security is needed in [11, 12]. In the case of implementing TI for an AES S-box, it is necessary to use at least five shares to protect against second-order attacks

[†] The authors are with the Department of Informatics, The University of Electro-Communications, Tokyo 182-8585, Japan.

[††] The author is with the Department of Information and Communications Engineering, Tokyo Institute of Technology, Tokyo 152-8550, Japan.

[*] This paper is a revised and extended version of [1]. We have conducted additional experiments in Section IV of [1] and discussed the randomness for sharing and for refreshing in the context of the security of the second-order masked AES.

a) E-mail: tsukasec09@gmail.com
b) E-mail: Y.Harada@uec.ac.jp
c) E-mail: sakiyama@uec.ac.jp

on nonlinear functions [10]. By contrast, the use of three shares for second-order security requires a lot of fresh randomness to compensate for the lack of uniformity, which is one of the essential requirements of TI. The process is called refreshing. In 2016, De Cnudde *et al.* [13] proposed the first and second-order masked implementation of AES with Consolidated Masking Scheme (CMS) [12] using $d+1$ input shares. Subsequently, De Meyer *et al.* [14] proposed M&M (Masks and Macs) implementing the AES S-box with $d + 1$ input shares in [13]. This technique is a promising countermeasure resistant to both SCAs and differential fault analysis (DFA) [15–17].

Over the years, there has been extensive study of first-order, second-order, and higher-order masking of AES, and numerous papers have appeared. Despite various discussions, all solutions for masking AES require a large number of randomness resources, i.e., a large number of random bits. Therefore, it is necessary to use random number generators (RNGs) in TI and to focus on the relationship between their randomness and physical security. However, how the randomness of RNG affects masking security is not well-discussed, and to the best of our knowledge, no such paper has been found.

Furthermore, the increasing need to incorporate SCA countermeasures into commercial equipment requires a method that can quickly and robustly evaluate side-channel leakage. For example, DPA [3] and CPA [4] are practical methods for verifying the feasibility of attacks, but they are time-consuming due to the need to assume many intermediate values and leakage models. These methods are also impractical because the types and number of SCAs are increasing day by day. Therefore, there is a growing demand to establish evaluation methods that are independent of attacks, intermediate values, and leakage models. One such approach is known as the leakage assessment method using Welch's t-test [18]. This method can efficiently inspect leakage without actual attacks. Becker *et al.* [19] introduced the TVLA (Test Vector Leakage Assessment), a more specific procedure for evaluating leakage using Welch's t-test.

(1) Contributions

To the best of our knowledge, no previous studies have discussed and evaluated the randomness of RNGs used for masking countermeasures. In this paper, for the first time, we quantitatively investigate the impact of randomness used in TI on security against SCAs using an FPGA to implement a second-order masked AES with three shares. The contributions of this paper are summarized as follows:

i) We present that the randomness of sharing is directly related to side-channel leakage. The TVLA evaluation observes leakage when the secret data is divided into shares using fixed random numbers for each encryption process.

ii) Next, we show that insufficient randomness for refreshing leads to information leakage regardless of the randomness of sharing. Specifically, in addition to the contributions of [1], we observed leakage as a result even when the frequency of random number updates was reduced.

iii) Finally, based on the TVLA results, we propose randomness suitable for implementation, both when used in share sharing and when used in S-box calculations masked by TI featuring a pipeline structure. The empirical experiments show that it is important to use different seed values and to keep updating the random numbers since any loss of randomness in either of these cases can have a negative impact on physical security.

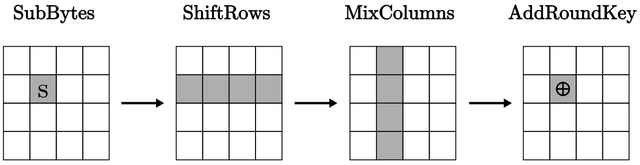


Fig. 1: Algorithm of the AES cipher

(2) Organization

The remainder of this paper is organized as follows. In Section 2, we describe the background knowledge: AES, the compact S-box, TI, and RNGs. Section 3 introduces a case study of M&M AES with second-order SCA security, and explains a masked compact S-box based on tower-field decomposition. Section 4 shows the target evaluation device and the experiment setup. Section 5 evaluates the impact of randomness used in masking cryptographic hardware. In Section 6, we discuss practical randomness suitable for masking implementations. Finally, we conclude the paper with Section 7.

2. Preliminaries

2.1 Description of AES

AES (Advanced Encryption Standard) [2] is a 128-bit block cipher proposed by Daemen and Rijmen, and adopted by NIST (National Institute of Standards and Technology) in 2000. The key length can be selected from 128, 192, and 256 bits, while this paper focuses on the 128-bit key. The cipher applies ten round functions consisting of SubBytes for nonlinear operations, ShiftRows and MixColumns for linear operations, and AddRoundKey for key addition. The overview of the AES algorithm is illustrated in Fig. 1.

1. **SubBytes**: Calculate by S-box for each byte. The S-box combines the calculation of the multiplicative inverse in GF(2^8) and affine transformation.

2. **ShiftRows**: Shift the left 4-byte row by a constant rule.

3. **MixColumns**: Multiply a constant matrix by 4-byte in the column direction.

4. **AddRoundKey**: Calculate XOR with the round key.

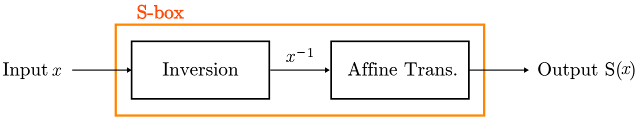


Fig. 2: Compact AES S-box architecture

In this paper, the S-box output result for input value x is denoted as $S(x)$.

2.2 S-box in hardware implementation

To implement the S-box on an FPGA, look-up tables (LUTs) are normally used, but in scenarios with limited resources, such as memory usage or circuit area, the following two-step computation scheme, as shown in Fig. 2, is also used.

1. Inversion: Calculate the inverse multiplicative inverse y for input x over GF(2^8).

$$y = \begin{cases} 0 & \text{if } x = 0, \\ x^{-1} & \text{otherwise.} \end{cases} \tag{1}$$

2. Affine transformation: Apply the following matrix transformation.

$$\begin{pmatrix} S_7 \\ S_6 \\ S_5 \\ S_4 \\ S_3 \\ S_2 \\ S_1 \\ S_0 \end{pmatrix} = \begin{pmatrix} 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \\ 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \end{pmatrix} \begin{pmatrix} y_7 \\ y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}, \tag{2}$$

where S is the output value of the S-box, and subscripts indicate bit positions (# 7 is the most significant bit).

However, computing multiplicative inverses is generally time-consuming and increases circuit area, thus requiring efficient implementation. Accordingly, compact S-boxes with composite fields have been proposed so far [20, 21]. In particular, Canright [21] designed a very compact S-box implementation in tower fields using a normal-basis representation.

2.3 Threshold Implementation

The TI countermeasure, proposed by Nikova *et al.* at ICICS 2006 [9], is one of the Boolean masking methods and is based on multiparty computation (MPC). TI is a technique that divides each secret variable in a circuit into pairs of values called shares. In this paper, we use shares with three elements. Given a variable x over GF(2^8), its shared value is represented as $\bar{x} = \{x_a, x_b, x_c\}$ such that $x = x_a + x_b + x_c$. The benefit of representing a value x by a share $\bar{x}$ is that if each share is uniformly distributed, no information about x is revealed, even with the proper subset of the shares. In addition, in representing data by shares, d-th order TI is provably secure against d-probing model[†] attacks. If not only the data but also the mapping is made to have similar properties, the entire cryptographic process can be made resistant to the probing attacks.

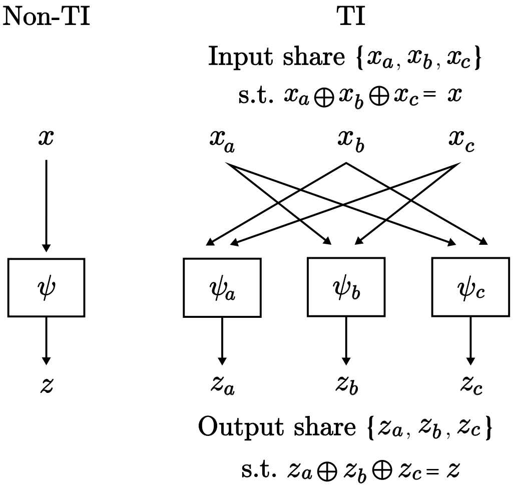

Non-TI
TI
Input share $\{x_a, x_b, x_c\}$
s.t. $x_a \oplus x_b \oplus x_c = x$
x
x_a
x_b
x_c
ψ
ψ_a
ψ_b
ψ_c
z
z_a
z_b
z_c
Output share $\{z_a, z_b, z_c\}$
s.t. $z_a \oplus z_b \oplus z_c = z$

Fig. 3: Overview of TI and Non-TI

Here, a mapping $z = \psi(x)$ is considered, where z represents the final output result and x is the original input value. When computing ψ in a secure way using TI's share, the mapping ψ is split into $\{\psi_a, \psi_b, \psi_c\}$.

$$\begin{aligned} z_a &= \psi_a(x_b, x_c), \\ z_b &= \psi_b(x_c, x_a), \\ z_c &= \psi_c(x_a, x_b), \end{aligned} \tag{3}$$

where $\{z_a, z_b, z_c\}$ is an output share. The above pair is called a share map. An overview of the share map in TI illustrating Eq. (3) and Non-TI is shown in Fig. 3. TI requires the share map to have three properties: Correctness, Non-completeness, and Uniformity.

1. **Correctness**: The output result must be the same as the original mapping ψ using the share map $\psi_a, \psi_b, \psi_c\}$. That is, a shared map is correct if it satisfies the following Eq. (4).

$$\begin{aligned} \psi(x) &= z \\ &= z_a + z_b + z_c \\ &= \psi_a(x_b, x_c) + \psi_b(x_c, x_a) + \psi_c(x_a, x_b). \end{aligned} \tag{4}$$

2. **Non-completeness**: A share map is non-complete if all share maps $\{\psi_a, \psi_b, \psi_c\}$ use only a proper subset of the input share $\bar{x}$. This does not provide any information about the original value x.

3. **Uniformity**: Shares must appear equally, i.e., a probability of occurrence of each input share must be equal

[†] This model was proposed by Ishai *et al.* [22], where the attacker has d probes and can see up to d gates or wires in a circuit in a given time, e.g., one clock cycle.

for any input. Let $\bar{X}$ be a random variable for the share. A share $x = \{x_a, x_b, x_c\}$ is said to be uniform if it occurs with equal probability for a given x, as in Eq. (5).

$$\forall x_a, x_b, x_c,\ \Pr[\bar{X} = \{x_a + x_b + x_c\}] = \alpha, \tag{5}$$

where α is a constant. Also, the share map $\{\psi_a, \psi_b, \psi_c\}$ is uniform if for a map

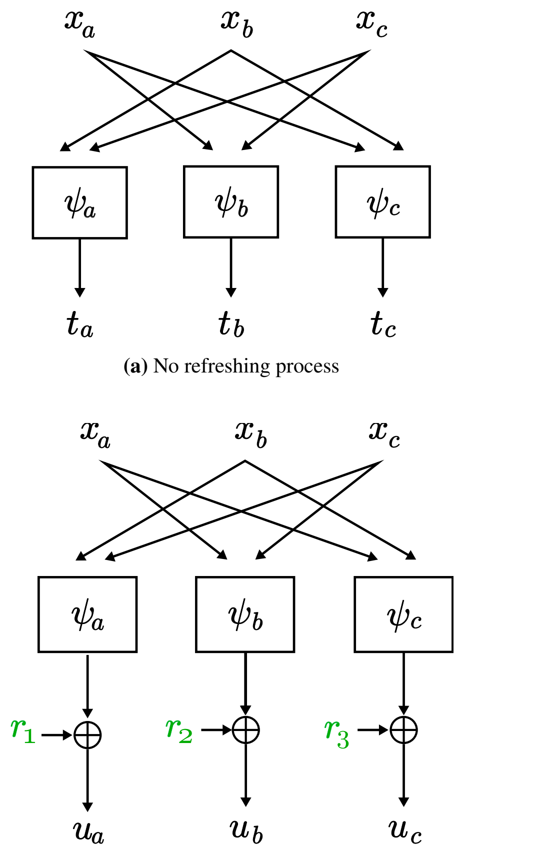
$$\{x_a, x_b, x_c\} \overset{\{\psi_a,\psi_b,\psi_c\}}{\longmapsto} \{z_a, z_b, z_c\},$$

the input share $\{x_a, x_b, x_c\}$ is uniform, then the output share $\{z_a, z_b, z_c\}$ is also uniform.

Let $\{x_a, x_b, x_c\}$ and $\{t_a, t_b, t_c\}$ denote the shares of x and t, respectively, and $\{\psi_a, \psi_b, \psi_c\}$ denotes the share maps of ψ. Consider the relationship between share and share map as in Eq. (6).

$$\{x_a, x_b, x_c\} \overset{\{\psi_a,\psi_b,\psi_c\}}{\longmapsto} \{t_a, t_b, t_c\}. \tag{6}$$

The map of Eq. (6) is shown in Fig. 4a, where the input share $\{x_a, x_b, x_c\}$ is assumed to be uniform. Notice that if the share map $\{\psi_a, \psi_b, \psi_c\}$ satisfies uniformity, the output share $\{t_a, t_b, t_c\}$ is uniform. On the other hand, consider the case where the share map $\{\psi_a, \psi_b, \psi_c\}$ does not satisfy uniformity, e.g., a two-input nonlinear operation (such as AND gates) whose number of elements in the input share is three. Due to the fact that the share $\{t_a, t_b, t_c\}$ is not uniform [23], it is necessary to add fresh random bits (r_1, r_2, r_3) such that $r_1 + r_2 + r_3 = 0$ to the output of the share map $\{\psi_a, \psi_b, \psi_c\}$ to ensure the uniformity, as shown in Fig. 4b. This process is called refreshing.



(a) No refreshing process

(b) Refreshing process by random bits (r_1, r_2, r_3) such that $r_1 + r_2 + r_3 = 0$

Fig. 4: Share maps $\{\psi_a, \psi_b, \psi_c\}$

2.4 Random Number Generator

In order to implement SCA countermeasures, such as the randomization of target values in the masking and refreshing process, it is necessary to implement random number generators (RNGs) and add random numbers. There are two types of RNGs: true random number generators and pseudo-random number generators.

True random number generators (TRNGs) generate random numbers by exploiting the entropy source in the behavior of physical phenomena. TRNGs are in high demand in the security field due to their high level of security. The random number needs to be robust against reproduction, i.e., the next random bit is unpredictable. However, TRNGs are normally not cost-effective, mainly due to their low throughput, high power consumption, and difficulty in generating reliable random numbers. Furthermore, it is known that the frequency injection attack [24] significantly reduces the randomness of ring-oscillator-based TRNGs by applying electromagnetic waves of a certain frequency to them.

On the contrary, pseudo-random number generators (PRNGs) generate random numbers using a deterministic algorithm. Compared to TRNGs, they are relatively easy to implement, convenient, and environment-independent. It generates arbitrary-length bit strings from fixed-length initial inputs called seed values so that the generated outputs could generate unique random numbers. In addition, the same seed value will produce the same random number, so it is necessary to use a new seed value each time. In this paper, we employ Keccak [25] as PRNG.

(1) Keccak

Keccak is one of the cryptographic hash functions selected by NIST as the SHA-3 standard in 2012 [25]. Keccak can output random numbers of arbitrary length for variable-length input based on a sponge structure [26]. The Sponge structure can be divided into two main categories: absorbing and squeezing. In absorbing, the input is absorbed into the internal state and then replaced. In contrast, in squeezing, the value is squeezed out and then replaced. The substitution process converts $5 \times 5 \times w$ bits of state to another state for a word length w ($w = 2^l$) and uses a round function consisting of five steps: θ, ρ, π, χ, and ι. Due to the sponge structure, Keccak outputs the hash value after repeating all of these steps $12 + 2l$ times. However, for efficiency, we use output values from applying the round function to states as new random numbers.

2.5 TVLA

The test vector leakage assessment (TVLA) [19], a method for assessing physical vulnerabilities in cryptographic hardware, uses Welch's t-test. The Welch's t-test is a statistical method to test the hypothesis that two populations have equal means. The purpose of TVLA is not to obtain confidential information, such as secret keys from side-channel information, but to investigate the dependencies between measured power consumption and internal data, assuming that the plaintext and secret key are known. Therefore, TVLA can efficiently evaluate security because it does not depend on specific architectures, intermediate values, power models, etc., inside cryptographic modules.

In this paper, we perform the first- and second-order t-test on fixed-vs-random plaintexts using the following input datasets, where n is the number of encryptions.

- Fixed and random plaintext key
  K = 0x0123456789abcdef123456789abcdef0

- Fixed plaintext input data
  I = 0xda39a3ee5e6b4b0d3255bfef95601890

- Random plaintext input data
  I_0 = 0x00000000000000000000000000000000
  $I_{j+1} = \mathrm{AES}(K, I_j)$ for $0 \le j < n$

The sample mean and sample variance are obtained as explained in [19, 27]. The waveform data collected before the t-test must be preprocessed, and the squared data for each waveform is used in the second-order evaluation. Let us first denote the dth-order ($d > 1$) central moment by $CM_d = \frac{1}{n}\sum_{i=0}^{n-1}(l[i] - M_1)^d$, with n the number of waveforms, $l[i]$ the i-th waveform data and M_1 the mean of waveforms. Equation (7) shows how to obtain the sample mean μ and the sample variance v used in the first-order t-test.

$$\begin{aligned}\mu &= M_1 = \frac{1}{n}\sum_{i=0}^{n-1} l[i], \\ v &= CM_2.\end{aligned} \tag{7}$$

In addition, the method for obtaining the sample mean μ and the sample variance v used in the second-order t-test is expressed in Eq. (8).

$$\begin{aligned}\mu &= CM_2, \\ v &= CM_4 - (CM_2)^2.\end{aligned} \tag{8}$$

Equation (9) shows how to calculate the t-value, with μ_1 (μ_2) the mean, v_1 (v_2) the variance, and n_1 (n_2) the cardinality of each data set.

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{v_1}{n_1} + \frac{v_2}{n_2}}}, \tag{9}$$

where the subscript indicates each dataset (a set of fixed plaintexts or a set of random plaintexts).

In the t-test, $\mu_1 = \mu_2$ is set as the null hypothesis to be rejected. If the absolute value of the derived t-value exceeds the threshold of 4.5, we reject the null hypothesis at the 99.999 % confidence level. That is, there is a clear difference between the power waveforms for the fixed plaintext and the ones for the random plaintexts, which indicates that sensitive information may leak depending on the plaintexts. However, it is worth mentioning that this is just a necessary condition for a successful attack, not a sufficient condition in terms of the SCA countermeasure. It is also worth noting that the absolute value of the derived t-value below 4.5 only means the null hypothesis is not rejected at the 99.999 % confidence level, but not that there is no leakage.

3. A case study of M&M AES with second-order SCA security

This section deals with M&M AES as a case study of cryptographic hardware protected by TI technology. We briefly give an overview of M&M and describe a compact S-box protected by second-order TI with 3 input shares.

3.1 M&M (Masks and Macs)

M&M was proposed by De Meyer *et al.* at CHES 2019 [14]. It is a promising countermeasure that combines a masking scheme against SCAs and infective countermeasures against DFA. De Meyer *et al.* adopted TI [9] with Consolidated Masking Scheme (CMS) [13] as masking [14]. The infective countermeasure against DFA is based on redundancy in computation, i.e., two circuits are used: one for the plaintext and the other for the information-theoretic MAC tag for the plaintext. The output ciphertexts are then verified with the MAC tag. If a failure occurs, the error caused by the fault is diffused using random numbers. This makes it difficult for the DFA attackers to obtain the secret key even if they obtain the faulty ciphertext. In [14], they also designed a second-order masked AES on an FPGA to evaluate SCAs in M&M implementations and assessed information leakage using TVLA [19], and revealed that there is no leakage identified by the first- and second-order t-test.

3.2 Compact S-box protected by second-order TI with 3 shares

M&M is protected by second-order TI with 3 shares proposed by De Cnudde *et al.* [13]. This masked implementation is based on Canright's compact S-box [21] of tower field decomposition. The S-box has a pipeline structure consisting of six stages, each of which is separated by registers that are introduced to suppress the glitch propagation.

a) **First Stage**: The first operation is to convert the basis from 8-bit input to 8-bit output through a linear map for each share, as shown in Fig. 5. Circled Roman numbers in the figure are registers between stages.

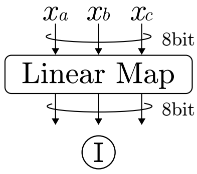


Fig. 5: Stage 1 process

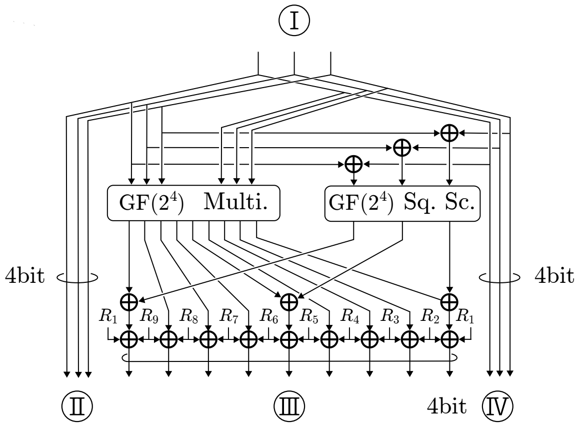


Fig. 6: Stage 2 process

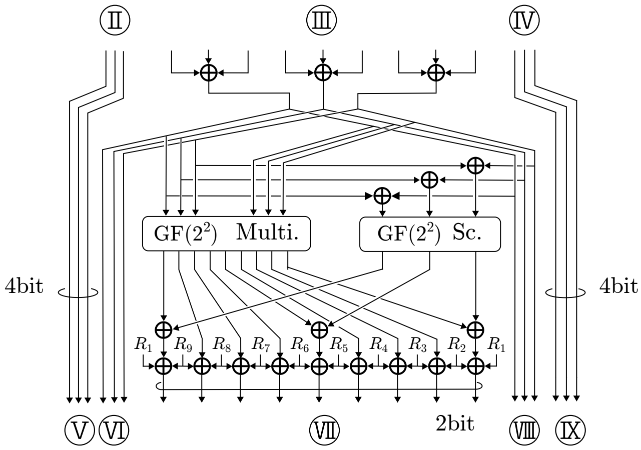


Fig. 7: Stage 3 process

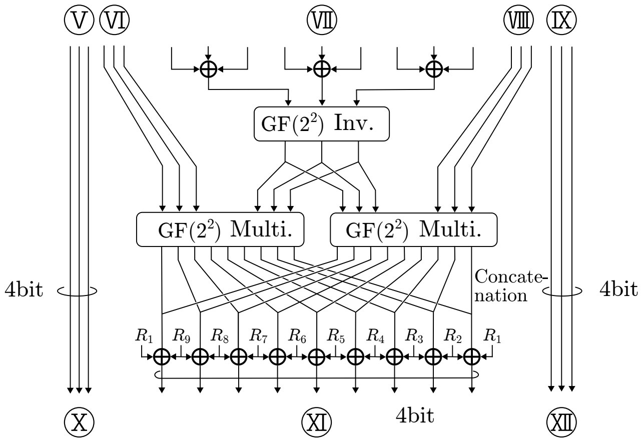


Fig. 8: Stage 4 process

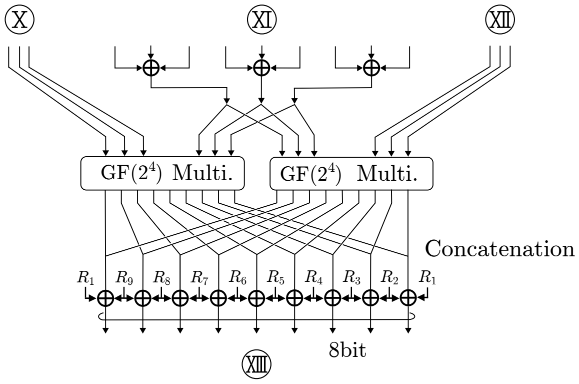


Fig. 9: Stage 5 process

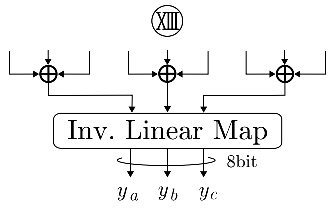


Fig. 10: Stage 6 process

Registers between stages 1 and 2 (Register Ⓘ) are necessary for security. Lack of these registers leads to non-completeness violations in the operation of the second stage. There is no need to re-mask at this stage because the operation in stage 1 is a linear operation only.

b) **Second Stage**: In the second stage, as shown in Fig. 6, we perform multiplication (Multi.), which is a nonlinear operation, and square scaling (Sq. Sc.) over in parallel. These outputs are then combined and re-masked with a total of 36 fresh random bits before being clocked in the register.

c) **Third Stage**: From Fig. 7, the process at stage 3 is similar to that at stage 2. Before the operation, the 4-bit input value is divided into 2-bit values. Fresh randomness of 18 bits is needed because the nine 2-bit output values must be re-masked.

d) **Fourth Stage**: The fourth stage, as shown in Fig. 8, consists of inversion and multiplication. The inverse operation in GF(2^2) is realized by swapping the signal using wires. Finally, the results of multiplication are concatenated and re-masked with 32 bits of fresh randomness.

e) **Fifth Stage**: From Fig. 9, the process at stage 5 is the same as that at stage 4. The difference from the fourth stage is the bit width of the value and the calculation of the inverse. After concatenating the results of the multiplication, we re-mask them with 72 bits of fresh randomness.

Table 1: Total number of random numbers used in each stage of the S-box

S-box	(Bit width)×(# 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

f) **Sixth Stage**: The sixth stage is depicted in Fig. 10. The last operation of the S-box is an inverse linear map. As in the first stage, this stage does not require fresh randomness.

3.3 Refreshing

The existence of uniformity is essential to guarantee security by non-completeness. In the compact S-box calculation featuring tower field decomposition described in Section 3.2, the output share of each stage is used as the input share of the subsequent stage. For example, consider the case where the output of a function f is the input to the next nonlinear function g and f is not uniform. For the purpose of satisfying uniformity in the subsequent stage, fresh random bits also need to be added to the output share of f.

From Figs. 6, 7, 8, and 9, the refreshing with a ring structure using fresh randomness is performed before the result of multiplication, which is a nonlinear operation, is stored in registers. Let $\{t_1, t_2, \ldots, t_9\}$ be the share after the nonlinear operation, $\{u_1, u_2, \ldots, u_9\}$ be the value clocked to the register, and $(r_1, r_2, \ldots, r_9)$ be the random bits. The details of the ring-refreshing are shown in Eq. (10).

$$\begin{aligned}
u_1 &= t_1 + r_1 + r_2,\\
u_2 &= t_2 + r_2 + r_3,\\
u_3 &= t_3 + r_3 + r_4,\\
u_4 &= t_4 + r_4 + r_5,\\
u_5 &= t_5 + r_5 + r_6,\\
u_6 &= t_6 + r_6 + r_7,\\
u_7 &= t_7 + r_7 + r_8,\\
u_8 &= t_8 + r_8 + r_9,\\
u_9 &= t_9 + r_9 + r_1.
\end{aligned} \tag{10}$$

The value XORed with all the elements of share $\{u_1, u_2, \ldots, u_9\}$ is equal to the value XORed with all the elements of share $\{t_1, t_2, \ldots, t_9\}$ (satisfying the correctness), with the random number $(r_1, r_2, \ldots, r_9)$ disappearing. One advantage of ring-refreshing is that the total of the random numbers does not need to be stored in a new register. The total number of fresh random bits used in each stage is shown in Table 1.

Table 2: 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
DC power supply	KEITHLEY 2260B-30-36
Oscilloscope	Tektronix MSO64
Amplification	ZFL-1000LN+
Implementation tools	Xilinx ISE Design Suite 14.7
Programming language	MATLAB R2020b

4. Experimental Preparation

4.1 Platform

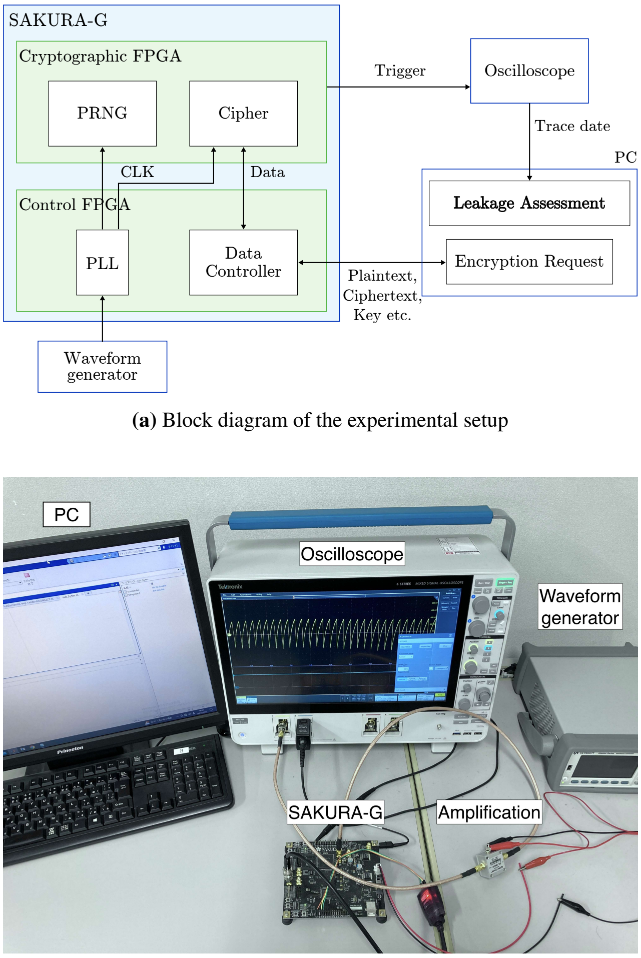
In the security evaluation for the usage of PRNGs, we use SAKURA-G [28] board for the evaluation of SCAs. This evaluation board features two Spartan-6 FPGAs: a control FPGA to handle sending and receiving plaintexts and keys from the PC and a cryptographic FPGA to execute AES. The separation of these two FPGAs allows power consumption to be measured only during the encryption process, reducing the effect of noise on waveform acquisition.

4.2 Experimental setup

Table 2 shows the details of the equipment used in the experiments. Figures 11a and 11b show an overview of the block diagram of the experimental setup and photographs of the experimental environment. We implement second-order masked M&M AES and Keccak in the cryptographic FPGA of SAKURA-G. We select the KEEP HIERARCHY option to prevent optimization and preserve architectural hierarchy at the synthesis. In the encryption run, the MATLAB code is first executed on a PC, and then the plaintext, secret key, the seed value for PRNG, etc., are sent to the control FPGA. The control FPGA then divides the received value into shares and sends them to the cryptographic FPGA for the encryption process. The waveform generator supplies a clock signal to the control FPGA and outputs two clock signals using a PLL (Phase-Lock Loop). We measure the voltage drop across a 1 Ω shunt resistor placed on the VDD path of the FPGA. We use an oscilloscope with a 12-bit analog-to-digital converter and a sampling rate of 1.25 GS/s to obtain the power consumption waveforms for the first round. We evaluate the information leakage from the acquired waveforms.

5. Leakage assessment

This section quantitatively investigates the impact of randomness used in second-order TI-AES, especially focusing on the SCAs resistance using TVLA. We assume two specific patterns of randomness in TI-AES: randomness used for sharing and randomness used for refreshing.



(a) Block diagram of the experimental setup

(b) Environment of experiments

Fig. 11: Experiment setup and Environment of experiments

5.1 Randomness for sharing

Share $\{x_a, x_b, x_c\}$ such that $x = x_a + x_b + x_c$ for sensitive data x is obtained from Eq. (11) using (r_a, r_b) as a random number.

$$\begin{aligned} x_a &= x + r_a, \\ x_b &= r_a + r_b, \\ x_c &= r_b. \end{aligned} \tag{11}$$

The generation of the share is performed on the control FPGA side, which uses as random numbers the AES ciphertext executed with the seed value given by the PC. If the seed value is a fixed value, output random numbers are also fixed.

Here, we compare the information leakage evaluation between the case with fixed random numbers and the case with normal random numbers. Figures 12 to 17 show the results of the t-test for the fixed-vs-random plaintexts. All figures (a) show the sample traces, (b) and (c) the results of the first- and second-order t-test, and (d) the maximum value (absolute value) of the t-statistic with the number of traces.

Table 3: Correspondence between experimental conditions and results in randomness for refreshing

Seed value for each encryption		Frequency of random number updates No update	 Once per round	 Every clock cycle
	Same	Fig. 13	N/A	Fig. 14
	Different	Fig. 15	Fig. 16	Fig. 17

Figure 18 plots each condition's maximum absolute number of t-statistics using 200 K, 300 K, 862 K, and 1 M traces, i.e., the t-statistic on the corresponding point in Fig. (d). Regarding the randomness used to divide the confidential value into shares, Fig. 12 shows the result with fixed random numbers, and Fig. 17 shows the result with normal random numbers. Note that refreshing processes appropriately both in Figs. 12 and 17. It is obvious from Fig. 12, where the t-statistic is much higher than the threshold 4.5 in absolute value. We observe side-channel leakage when the random number used to divide the secret data into shares is fixed, and no leakage when the random number used to divide the shares is random. This indicates that it is essential to create shares using different random numbers for each encryption process in order for the cryptographic hardware to have SCA resistance.

5.2 Randomness for refreshing

We use the output of the Keccak PRNG function, implemented on a cryptographic FPGA, as the random number for refreshing. The PRNG function takes the seed value, different from the seed value for resharing, from the control FPGA and outputs new random numbers every clock cycle. Accordingly, we focus on the seed value and the frequency of random number updates, i.e., freshness, as parameters of randomness for refreshing. Table 3 shows the corresponding experiments and results for different conditions of seed value and random number update frequency. Note that in Section 5.2, we use the appropriate random numbers for sharing in order to investigate only randomness for refreshing.

(1) Seed value for each encryption

We investigate the security impact of inputting fixed or random seed values into PRNG for each encryption process. The random number output from the PRNG is determined by the input or seed value. Therefore, if fixed seed values are input for each encryption, the same sequence of random numbers is output repeatedly.

First, from Figs. 13 and 14, when the same seed value is given for each encryption process, both t-test results exceed the threshold 4.5 in absolute value, regardless of the frequency of random number updates. In addition, comparing Figs. 14 and 17, neither the first- nor second-order t-statistics surpass the threshold 4.5 with up to 1 million power traces when different seed values are given for each encryption. In other words, this experiment confirms that using the same random number sequence for each encryption compromises TI security.

(2) Frequency of random number updates

The implemented PRNG outputs every clock cycle, and hence the fresh random number is available for refreshing TI circuits. However, we do not yet understand the effect on SCA leakage when the frequency of random number updates is reduced. Here, we evaluate the information leakage, especially focusing on the three prominent cases: the case when the random number update is stopped in Fig. 15, the case when the random number is updated once per round, i.e., once every 22 clock cycles in Fig. 16, and the case when the random number is updated every clock cycle as usual in Fig. 17. Figs. 15 and 16 show that even if a different seed value is provided for each encryption, we see first- and second-order leakage as the frequency of random number updates decrease. On the other hand, in Fig. 17, the t-statistic does not increase with the number of traces. This leads to the use of non-fresh randomness in the S-box calculation, which threatens the robustness of the TI, even if different seed values are input.

5.3 Overall comparison

The t-test result provides not only information on whether side-channel information leakage exists but also on the strength of leakage. Then we compare the t-test result of each condition in Figs. 12 to 17 and evaluate the quantitative security impact of randomness for share sharing and refreshing. First, the tendency of t-statistics as the number of traces increases shown in Figs. 12(d) to 17(d) is compared. T-statistics increased proportionally to the number of traces in Figs. 12(d) to 16(d), whereas in Fig. 17(d) the t-statistic oscillates independently of the number of traces. Note that the t-statistic momentarily exceeds the threshold because the test in a small number of traces is statistically unstable, and the reliability of the test increases as the number of traces increases. Next, the maximum absolute value of the t-statistics is compared with a certain number of traces. T-statistics is an indicator of the strength of leakage because a t-statistic is a statistic that expresses the confidence level of a hypothesis test and the higher the t-statistics, the higher the confidence level a hypothesis is rejected with. The t-statistics comparison in 200 K, 300 K, 862 K, and 1M traces between different conditions in Figs. 12 to 17 is shown in Fig. 18. The intensity of leakage in descending order is condition of Figs. 12, 13, 16, 14 and 15. These results show that first- and second-order leaks are detected in at least 1 M traces in the case where we use well-worn random numbers in the TI. This is synonymous with the need to generate fresh random numbers each time the use of random numbers is requested.

6. Discussion

In this section, we discuss the causes of the results of the leakage assessment in Section 5 for both randomness for sharing and randomness for refreshing and discuss practical randomness.

6.1 Randomness for sharing

We discuss the reason why information leakage happens when the random number for sharing is fixed. We focus on the registers between each stage of the S-box. Considering that power consumption is proportional to the number of transition bits in the registers, it depends on Hamming distance between adjacent bytes, i.e., the n-th and $(n+1)$-th bytes ($1 \leq n < 16$), due to the fact that the S-box calculation is a pipeline process. Furthermore, the generation of the share is based on Eq. (11), where x_b and x_c are also fixed if the random numbers (r_a, r_b) are fixed. Therefore, Hamming distance of x_b and Hamming distance of x_c for the register after the first stage processing is zero, and Hamming distance of x_a depends on the secret data before being split into shares. This means that we observe information leakage because the power consumption depends on the plaintext. From Fig. 12, we consider that the reason why the t-statistic becomes smaller after 9000 sample points, i.e., the second round, is due to the addition of random numbers to the intermediate values by refreshing. It is essential to divide the confidential value into shares using different random numbers when performing the encryption process to ensure TI security.

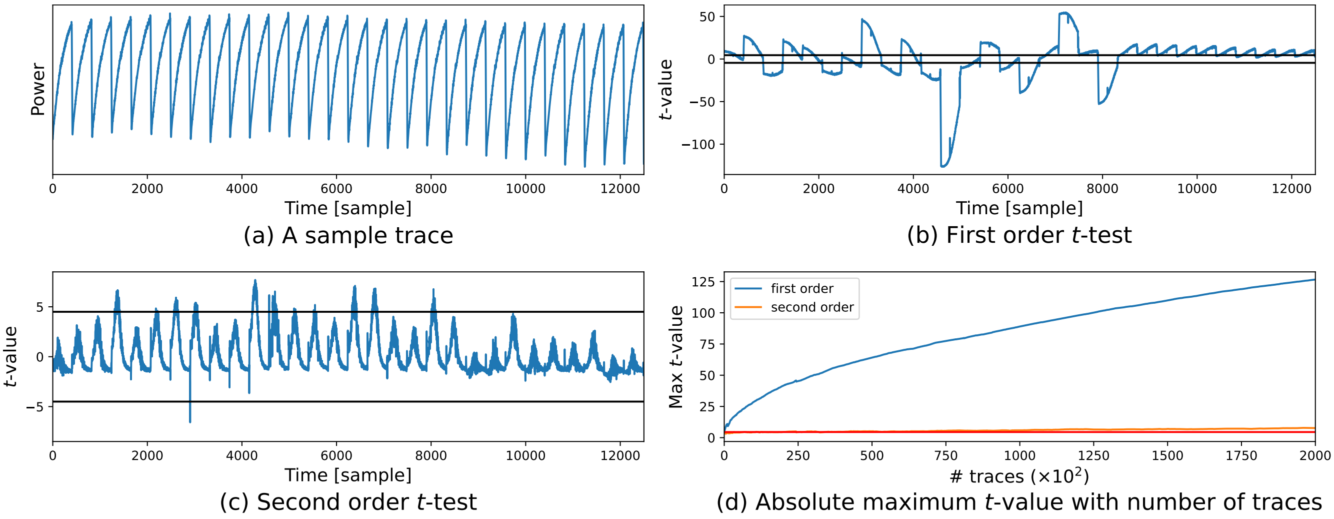
6.2 Randomness for refreshing

(1) Seed value for each encryption

It is necessary to send a different seed value to PRNG for each encryption to escape the side-channel leakage. We focus on the register after the refreshing process in stages 2 to 5 of the S-box. Because the same seed value is used, if we look at a certain clock cycle in each encryption execution, shares $\{u_1, u_2, \ldots, u_9\}$ stored in the registers, described in Eq. (10), use the same random number. In other words, it is equivalent to adding a constant value rather than a random number after nonlinear operations. Because of the variation in the probability of occurrence for each share, Hamming distance between non-uniform shares in the fixed plaintexts has a skewed distribution that does not follow a normal distribution. Thus, we consider that the t-test shows a significant difference between the fixed and random plaintexts.

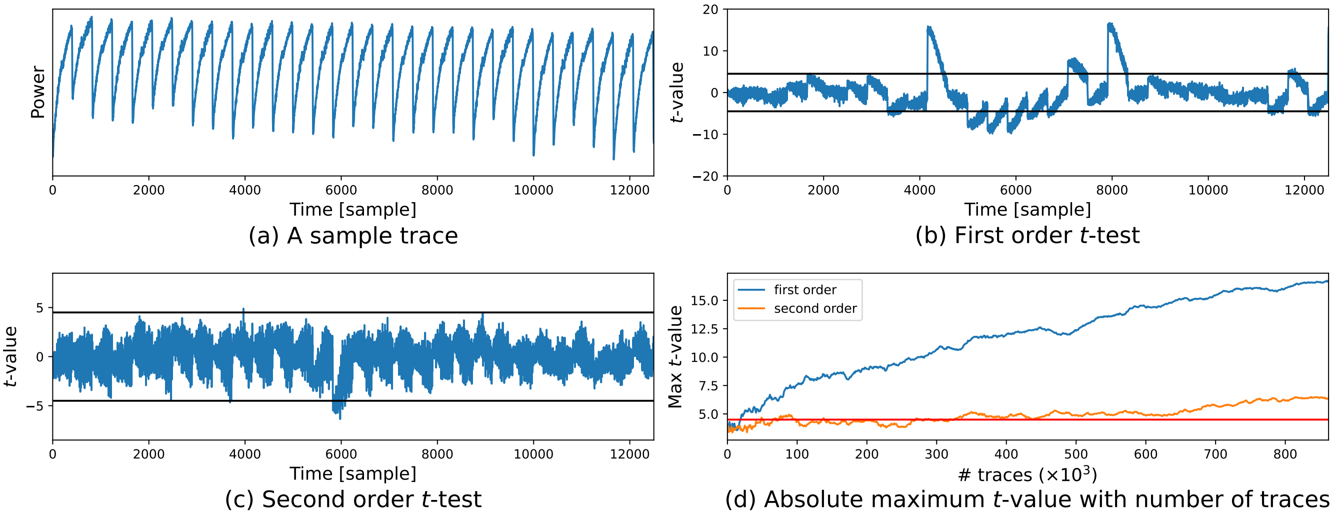
(2) Frequency of random number updates

From the experiments, it is clear that fresh random numbers should be updated every clock cycle to prevent information leakage. As in (1), we focus on shares $\{u_1, u_2, \ldots, u_9\}$ clocked to the registers after the refreshing process in stages 2



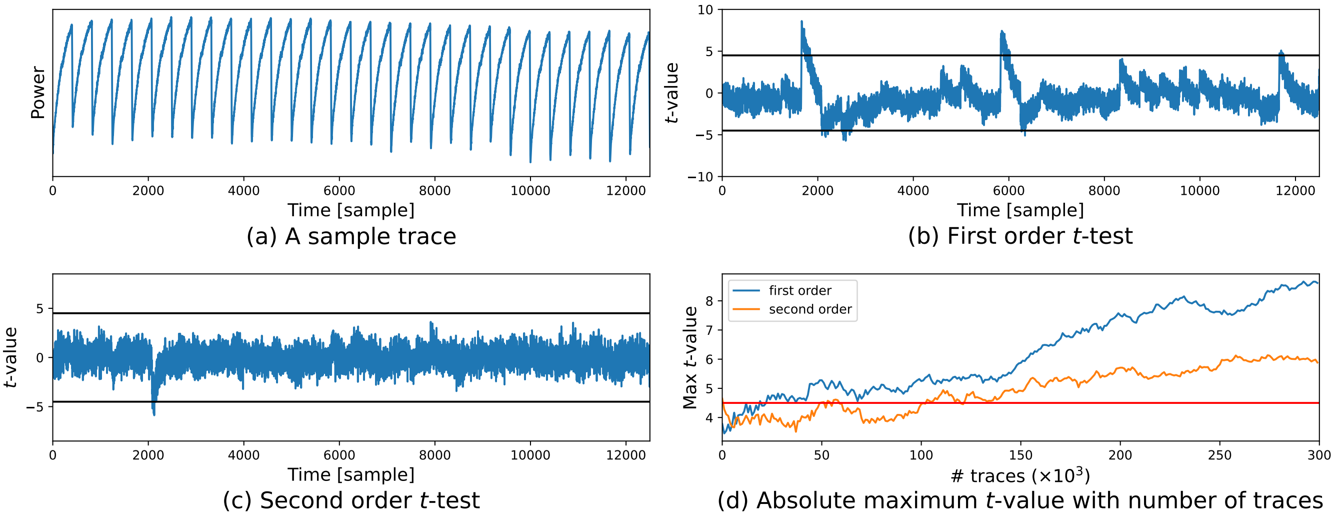
(a) A sample trace (b) First order *t*-test (c) Second order *t*-test (d) Absolute maximum *t*-value with number of traces

Fig. 12: Fixed-vs-random plaintexts t-test in the case where the random number for sharing is **fixed**, the random number for refreshing is **updated every clock cycle**, and its seed value is sent a **different** value for every encryption (200 K traces).



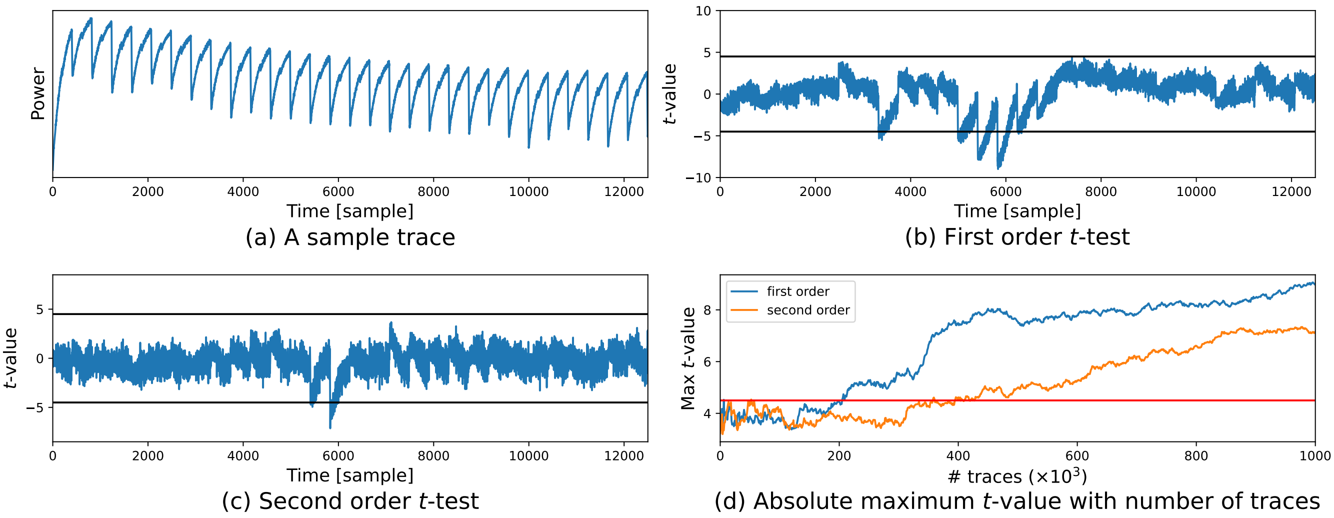
(a) A sample trace (b) First order *t*-test (c) Second order *t*-test (d) Absolute maximum *t*-value with number of traces

Fig. 13: Fixed-vs-random plaintexts t-test in the case where the random number for sharing is **randomized**, the random number for refreshing is **stopped updating**, and its seed value is sent the **same** value for every encryption (862 K traces).



(a) A sample trace (b) First order *t*-test (c) Second order *t*-test (d) Absolute maximum *t*-value with number of traces

Fig. 14: Fixed-vs-random plaintexts t-test in the case where the random number for sharing is **randomized**, the random number for refreshing is **updated every clock cycle**, and its seed value is sent the **same** value for every encryption (300 K traces).



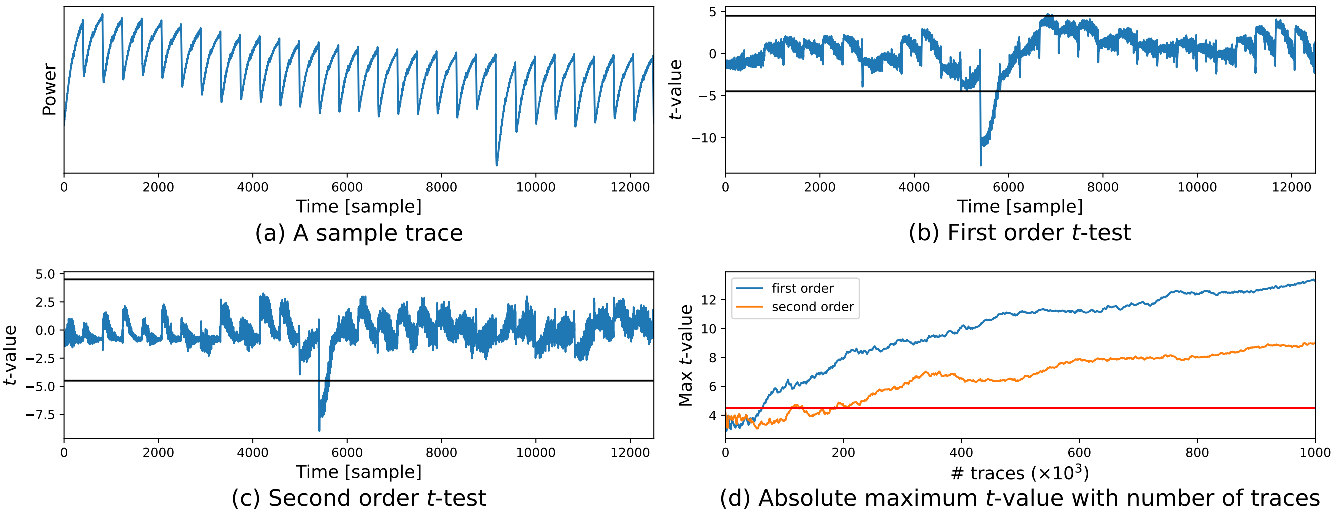
(a) A sample trace

(b) First order *t*-test

(c) Second order *t*-test

(d) Absolute maximum *t*-value with number of traces

Fig. 15: Fixed-vs-random plaintexts t-test in the case where the random number for sharing is **randomized**, the random number for refreshing is **stopped updating**, and its seed value is sent a **different** value for every encryption (1 M traces).



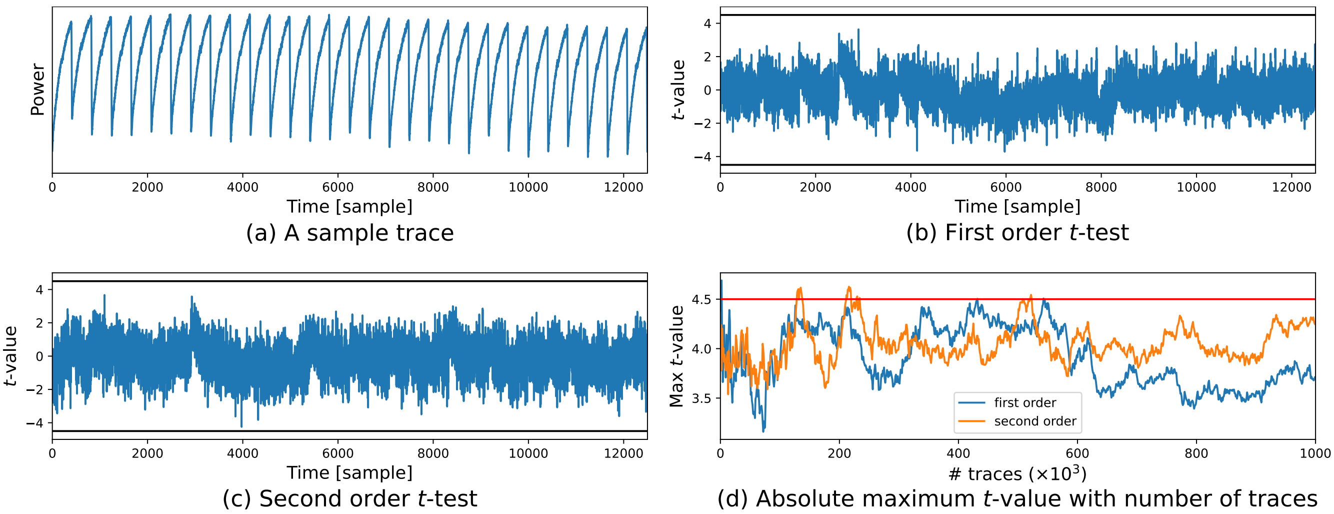
(a) A sample trace

(b) First order *t*-test

(c) Second order *t*-test

(d) Absolute maximum *t*-value with number of traces

Fig. 16: Fixed-vs-random plaintexts t-test in the case where the random number for sharing is **randomized**, the random number for refreshing is **updated once per round of AES**, and its seed value is sent a **different** value for every encryption (1 M traces).



(a) A sample trace

(b) First order *t*-test

(c) Second order *t*-test

(d) Absolute maximum *t*-value with number of traces

Fig. 17: Fixed-vs-random plaintexts t-test in the case where the random number for sharing is **randomized**, the random number for refreshing is **updated every clock cycle**, and its seed value is sent a **different** value for every encryption (1 M traces).

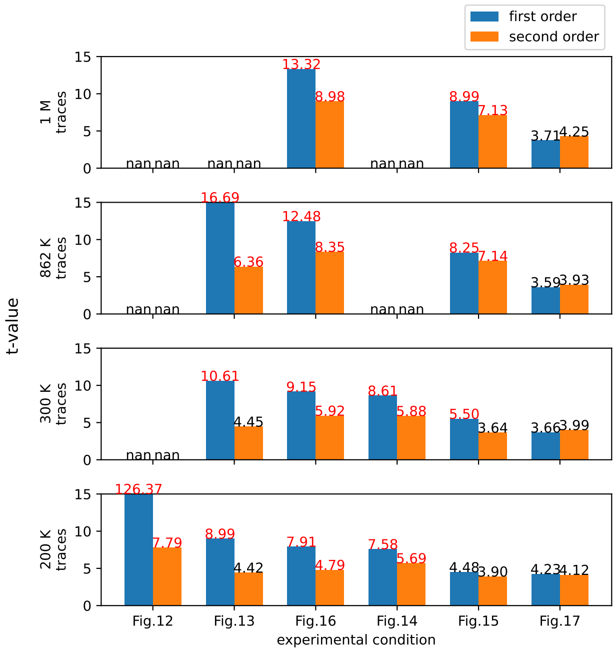


Fig. 18: Comparison of t-statistics between different conditions. Red-colored t-values are exceeding threshold 4.5.

to 5. As long as the random number is not updated, the same random number is added to each successive byte share. That is, the difference between shares $\{u_1, u_2, \ldots, u_9\}$ of adjacent bytes depends on shares $\{t_1, t_2, \ldots, t_9\}$ after the nonlinear operation, independent of the value of the random number. Because shares $\{t_1, t_2, \ldots, t_9\}$ do not satisfy uniformity and the probability of occurrence of each share is inconsistent, Hamming distance between shares $\{u_1, u_2, \ldots, u_9\}$ in the fixed plaintexts has skewed distribution that does not follow a normal distribution. Therefore, we consider that there is a significant difference between fixed and random plaintexts because a biased set of Hamming distances appears in the average of the power consumption traces.

7. Conclusions

To the best of our knowledge, there are no previous studies that discuss and evaluate the randomness of RNGs used in masking countermeasures. In this paper, for the first time, we quantitatively investigated the effect of randomness on physical security in second-order masked AES. Our study assessed information leakage by TVLA using a Xilinx FPGA implementing M&M AES as a case study of cryptographic hardware protected by TI technology. Three contributions were reported in this paper. First, we identified that the randomness used to divide sensitive data into shares is directly related to side-channel leakage. Based on the results of fixed-vs-random plaintext t-tests, we observed leakage when the random number for sharing was fixed. Second, we confirmed that insufficient randomness for refreshing leads to information leakage, regardless of whether the randomness for sharing is satisfactory or not. Specifically, the first- and second-order t-statistics surpassed the threshold of ±4.5 when the same seed value was input to the PRNG for each encryption or when random numbers were not updated, indicating the existence of leakage that could be exploited. Finally, we proposed a practical randomness suitable for TI-based hardware countermeasures. Experimental results have shown that either randomness for sharing or randomness for refreshing, whichever is compromised, has a negative impact on physical security. For making the AES cryptographic hardware resistant to SCAs, random numbers in TI must be updated each time. Our study consistently used the Keccak PRNG algorithm to investigate the effect of its randomness on SCA resistance. Future work includes the physical security evaluation when the PRNG algorithm is replaced. Also, we continue the study about what security requirements should be satisfied by PRNG and how they are implemented in practice.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers JP20H00590 and JP23K28083.

References

[1] M. Tsukahara, H. Hirata, M. Yang, D. Miyahara, Y. Li, Y. Hara-Azumi, and K. Sakiyama, "On the practical dependency of fresh randomness in AES S-box with second-order TI," Proc. 2023 Eleventh International Symposium on Computing and Networking Workshops (CANDARW), pp.286–291, IEEE, 2023.

[2] V. Rijmen and J. Daemen, "Advanced encryption standard," Federal Information Processing Standards Publications, vol.19, p.22, 2001.

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

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

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

[6] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," Cryptographic Hardware and Embedded Systems—CHES 2001: Third International Workshop Paris, France, May 14–16, 2001 Proceedings 3, pp.251–261, Springer, 2001.

[7] D. Agrawal, B. Archambeault, J.R. Rao, and P. Rohatgi, "The EM side—channel(s)," Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4, pp.29–45, Springer, 2003.

[8] L. Goubin and J. Patarin, "DES and differential power analysis the "duplication" method," Cryptographic Hardware and Embedded Systems: First InternationalWorkshop, CHES'99 Worcester, MA, USA, August 12–13, 1999 Proceedings 1, pp.158–172, Springer, 1999.

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

[10] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, "Higher-order threshold implementations," 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, pp.326–343, Springer, 2014.

[11] H. Groß, S. Mangard, and T. Korak, "An efficient side-channel protected aes implementation with arbitrary protection order," Cryptographers' Track at the RSA Conference, pp.95–112, Springer, 2017.

[12] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, "Consolidating masking schemes," Proc. CRYPTO 2015, pp.764–783, Springer, 2015.

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

[14] 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.

[15] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," Proc. CRYPTO'97, LNCS, vol.1294, pp.513–525, Springer, 1997.

[16] G. Piret and J.J. Quisquater, "A differential fault attack technique against SPN structures, with application to the AES and KHAZAD," Cryptographic Hardware and Embedded Systems-CHES 2003: 5th International Workshop, Cologne, Germany, September 8–10, 2003. Proceedings 5, pp.77–88, Springer, 2003.

[17] C. Giraud, "DFA on AES," Advanced Encryption Standard–AES: 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers 4, pp.27–41, Springer, 2005.

[18] B.J. Gilbert Goodwill, J. Jaffe, and P. Rohatgi, "A testing methodology for side-channel resistance validation," NIST Non-Invasive Attack Testing Workshop, NIST, vol.7, pp.115–136, 2011.

[19] G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi, and S. Saab, "Test vector leakage assessment (TVLA) methodology in practice," International Cryptographic Module Conference, ICMC, vol.1001, p.13, 2013.

[20] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact rijndael hardware architecture with S-box optimization," International Conference on the Theory and Application of Cryptology and Information Security, pp.239–254, Springer, 2001.

[21] D. Canright, "A very compact S-box for AES," Proc. CHES 2005, LNCS, vol.3659, pp.441–455, Springer, 2005.

[22] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," Proc. CRYPTO 2003, pp.463–481, Springer, 2003.

[23] S. Nikova, V. Rijmen, and M. Schläffer, "Secure hardware implementation of nonlinear functions in the presence of glitches," Journal of Cryptology, vol.24, pp.292–321, 2011.

[24] A.T. Markettos and S.W. Moore, "The frequency injection attack on ring-oscillator-based true random number generators," International Workshop on Cryptographic Hardware and Embedded Systems, pp.317–331, Springer, 2009.

[25] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Keccak," Proc. EUROCRYPT 2013, pp.313–314, Springer, 2013.

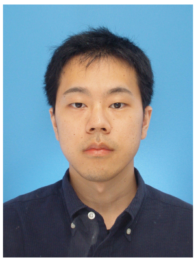
[26] B. Guido, D. Joan, P. Michaël, and V. Gilles, "Cryptographic sponge functions," 2011.

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

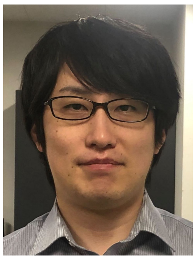
[28] "Sakura hardware security project." `http://satoh.cs.uec.ac.jp/SAKURA/index.html`.

Maki Tsukahara received his B.E. and M.E. degrees from The University of Electro-Communications (UEC), Tokyo, Japan, in 2022 and 2024. His major research interest is side-channel attacks on cryptographic hardware.

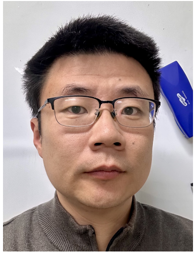
Yusaku Harada received his B.E. degree from The University of Electro-Communications (UEC), Tokyo, Japan, in 2024. Now he is a master's student at the Department of Informatics, UEC. His research interests include hardware security.



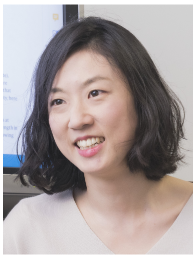
Haruka Hirata received the B.E. and M.E. degrees from The University of Electro-Communications (UEC), Tokyo, Japan, in 2021 and 2023. Currently, he is a doctoral program student at UEC. His main research interests include fault attacks on cryptographic devices.



Daiki Miyahara received his B.E., M.E., and Ph.D. degrees from Tohoku University in 2017, 2019, and 2021, respectively. He has been an Assistant Professor at The University of Electro-Communications since 2021. His research interests include theoretical computer science, information security, and cryptography. He is a member of IEICE and IPSJ.



Yang Li received his Ph.D. degree in engineering from The University of Electro-Communications, Tokyo, Japan, in 2012. He is currently an Associate Professor at the Department of Informatics, The University of Electro-Communications. His main research interests include security evaluation and improvement for cryptographic hardware and IoT devices.



Yuko Hara-Azumi received her Ph.D. degree in information science from Nagoya University, Japan, in 2010. She was a JSPS Postdoctoral Research Fellow with Ritsumeikan University, from 2010 to 2012, during which she was also a Visiting Scholar with the University of California, Irvine, USA, and Karlsruhe Institute of Technology, Germany. In 2012, she joined Nara Institute of Science and Technology as an Assistant Professor. Since 2014, she has been with Tokyo Institute of Technology, where she is currently an Associate Professor. Her research interests include system-level design automation, especially on high-level and logic synthesis, microprocessor architectures, and hardware/software co-design for embedded/IoT systems. She currently serves as an Organizing and a Program Committee Member for several premier conferences, including DAC, ICCAD, DATE, CASES, ASP-DAC, and FPL.

Kazuo Sakiyama is a professor at The University of Electro-Communications (UEC), Tokyo, Japan. At UEC, he leads the hardware security research for embedded cryptosystem, cyber-physical system, and physical authentication. Before joining UEC in 2008, he worked for Hitachi, Ltd., Japan (now Renesas Electronics) as a digital hardware designer, and later at Katholieke Universiteit Leuven (KU Leuven), Belgium as a Ph.D. research assistant. He received his B.E. and M.E. degrees from Osaka University, Japan in 1994 and 1996, respectively, the M.S. degree from The University of California, Los Angeles (UCLA) in 2003, and the Ph.D. degree in electrical engineering from KU Leuven in 2007. He is a member of IACR and IPSJ, and a senior member of IEEE.