SCIENCE CHINA
Information Sciences

• LETTER •

March 2020, Vol. 63 139104:1–139104:3
https://doi.org/10.1007/s11432-018-9540-y

An improved PAAs countermeasure based on permutation tables and latch PUF

Bing LI, Shuai CHEN* & Kai WANG

Advanced Cloud-Systems Research Center, School of Microelectronics, Southeast University, Nanjing 210000*, China*

Received 27 April 2018/Accepted 6 August 2018/Published online 10 February 2020

Citation Li B, Chen S, Wang K. An improved PAAs countermeasure based on permutation tables and latch PUF. Sci China Inf Sci, 2020, 63(3): 139104, https://doi.org/10.1007/s11432-018-9540-y

Dear editor,

Maintaining the secrecy of secret keys under the so-called power analysis attack (PAA) is challenging (the summary of PAAs countermeasure was shown in Appendix A). Prior studies have demonstrated that conventional permutation tables masking scheme is a powerful method for PAAs countermeasure [1]. However, this masking scheme lacks consideration of the leakages derived from the simultaneously updated registers [2] (shown in Appendix B). In view of this vulnerability, we propose an improved permutation tables masking scheme which can manipulate the sensitive values simultaneously without leaking key information in the Hamming distance model.

Furthermore, we propose a Latch physical unclonable functions (PUF) [3, 4] based a true random number generator (TRNG) to generate random bits as masks. The throughput of the TRNG is 12.5 Mbps in 25 MHZ clock which satisfies the requirements of masks in our masking scheme. Also, experiment results demonstrate that our TRNG pass the National Institute of Standards and Technology (NIST) random number tests. For security issue, experiment results show that the Pearson coefficient reduced from 0.8 to nearly 0, which demonstrate that our improved permutation tables masking scheme provides a relatively higher security against first-order PAAs than conventional permutation tables masking schemes.

Our improved permutation tables masking scheme. As shown in Figure 1(a), a 16-bytes data was operated in an AES (advanced encryption standard) round function for encryption, and each byte of this data was transformed in the same transformations. Based on this property, we proposed to randomly exchange the contents of masked intermediate value $P(z_i)$ where $0 \leqslant i \leqslant 15$ in each operation process to remove the dependency between the distributions of L_{sum1} and $\Delta(S_h(z))$.

Our masked AES scheme works as follows. Upon each invocation of our masked AES, M was randomly selected from a set of permutations $\mathbb{M}$ which defined over $\mathbb{F}_2^4$, for example, $M = [14, 6, 0, 5, 9, 1, 4, 15, 8, 10, 12, 2, 3, 13, 11, 7]$. Here, $\mathbb{M}$ cannot be defined as the set of all the permutations over $\mathbb{F}_2^4$. Essentially, in order to ensure the correctness of AES, $\mathbb{M}$ must include all numbers between 0 and 15. Moreover, M was generated by a random number k_M in this study.

After the randomized AddRoundKey transformation, masked intermediate value $P(z_i)$ were operated to $P(z_i)'$ according to M: $P(z_i)' = [P(z_{14}), P(z_6), P(z_0), P(z_5), P(z_9), P(z_1), P(z_4), P(z_{15}), P(z_8), P(z_{10}), P(z_{12}), P(z_2), P(z_3), P(z_{13}), P(z_{11}), P(z_7)]$, where $0 \leqslant i \leqslant 15$.

At the beginning of randomized SubBytes transformation, S-box was accessed at the address $P(z_i)'$, for $0 \leqslant i \leqslant 15$. Essentially, after that, the values of registers R_1 and R_2 were no longer based on $P(z_i)$, but associated with $P(z_i)'$.

At last, function M^{-1} (which was inversed from M) was applied to exchange the output of regis-

* Corresponding author (email: chenshuai_ic@seu.edu.cn)

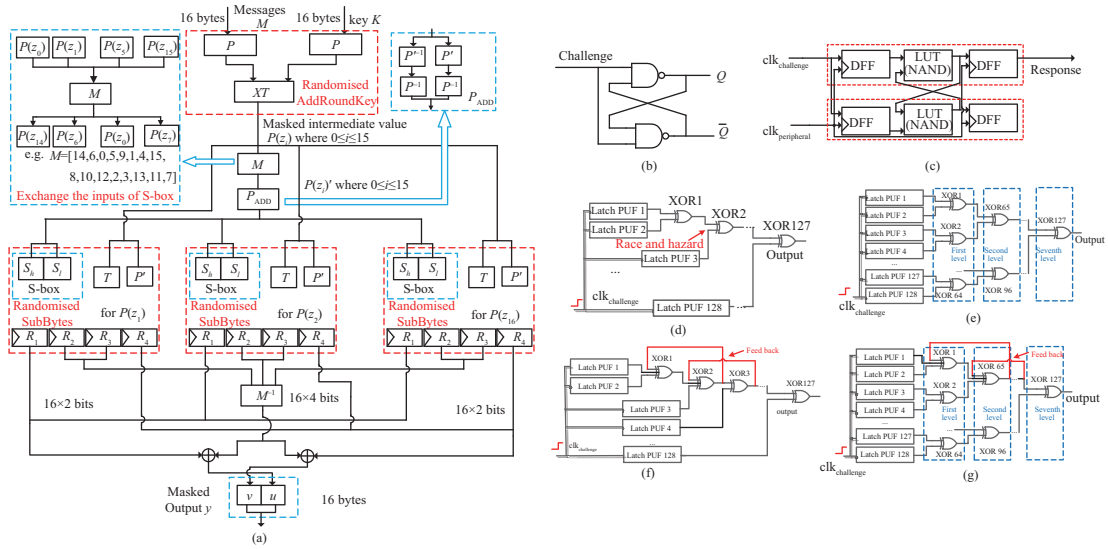

Figure 1 (Color online) The structure of our improved permutation tables masking scheme and Latch PUF based TRNG. (a) The structure of improved permutation tables masking scheme; (b) the structure of Latch PUF cell; (c) the structure of Latch PUF cell in FPGA [5]; (d) XOR corrector in [5]; (e) symmetrically placed XOR corrector; (f) XOR corrector with feed-back; (g) symmetrically placed XOR corrector with feed-back.

ters in the 16 times of randomized SubBytes transformation, in order to get the correct encrypting output.

Here, we will analyze the leakage of simultaneously updated registers R_1, R_2, R_3 and R_4 in our masked AES. Take one byte of randomized intermediate value $P(z_1)$ for example, the global leakage of registers R_1 and R_2 (also denoted by L_{sum1}) satisfies

$$\begin{aligned} L_{\text{sum1}} = {} & \Lambda(\mu(\Delta(S_l(z_1))) \oplus \mu(\Delta(S_h(P(z_1))))) \\ & + \Lambda(\mu(\Delta(S_h(M(P(z_1)))))) \\ & + N_{R1} + N_{R2}. \end{aligned} \tag{1}$$

As shown in (1), except for the condition that $M(P(z_1)) = P(z_1)$, the distribution of L_{sum1} in our masked AES is independent of sensitive variable $\Delta(S_h(z))$. In our masked AES, M was freshly selected from $\mathbb{M}$ by a uniformly distributed mask k_M before each execution. Therefore, without the knowledge of k_M, an attacker who makes statistics for the power consumption of L_{sum1} only can get random values. Therefore, the randomness of masks (k_M, k_4, k_3, k_2 and k_1) is the most important issue to against PAAs and some other attacks [2]. In this study, a Latch PUF based TRNG was used to generate the uniformly distributed masks. The FPGA implementation of this Latch PUF based TRNG will be given bellow.

FPGA Implementation of Latch PUF Based TRNG. PUF is a kind of device with unique "binary behavior" which depends on Integrated Circuit(IC) manufacturing variations [4]. PUF based TRNGs extracts randomness directly from the response of PUF insides. The advantage of this TRNG includes much better resiliency against physical attacks, the absence of side channel information, and much lower overheads [5]. Therefore, many studies so far have focused on PUF based TRNGs [6] (the summary of PUF based TRNG was shown in Appendix C).

Although, the masks in our improved permutation tables masking scheme do not need a high throughput. It just requires 48-bits random number in each encryption process (256 clock cycles), it must be produced by a uniformly distributed random number generator. Here, we choose the Latch PUF based TRNG for this application scenario.

One of the methods for implementing Latch PUF (Figure 1(b)) as a TRNG on Xilinx Virtex-4 FPGAs is proposed in [5]. Hata et al. [5] adopted a XOR corrector to link many Latch PUF FPGA implementation (Figure 1(c)) to generate sufficient entropy (Figure 1(d)). However, the placement and routing of this XOR corrector is actually performed by automatic place-and-route software. This method, as shown in Figure 1(d), causes a race and hazard problem at the output of gate XOR 2. Therefore, the value of eventual output only depends on parts of Latch PUF cells, and the entropy of the random bits might be decreased.

In order to decrease the issue of race and hazard aforementioned, we implemented the XOR corrector symmetrically. The schematic of our XOR corrector is shown in Figure 1(e). In the first level of our XOR corrector, 64 XOR gates were closely connected to 128 Latch PUF cells. Similar property existed in the rest five levels of our XOR cor-

rector. Consequently, for every XOR gate, the two input signals arrived almost at the same time. Unfortunately, our prior studies show that the improvement are limited, the entropy of random bits are still not enough.

To improve the entropy of this PRNG, we proposed to use feed forward to introduce nonlinearity in the circuits. As shown in Figure 1(f) and (g), the output of XOR 2 was fed back to the input of XOR 1. Then the output of XOR 3 is fed back to the input of XOR 2. Furthermore, we use feed forward in the symmetrically placed XOR corrector to improve the entropy.

In our work, we carefully implemented these four different Latch PUF based TRNGs on a Xilinx SP3E FPGAs boards. The sampling signal $\mathrm{clk}_{\mathrm{sampling}}$ was fed by the peripheral clock (50 MHz), and $\mathrm{clk}_{\mathrm{sampling}}$ (25 MHz) was generated from peripheral clock by using a digital clock manager (DCM).

Experiment Results. There are two important requirements for the masks in our masking scheme: enough throughput and uniformly distribution.

As aforementioned, the throughput must $\geqslant$ 4.688 Mbps when the clock frequency of the encryption is 25 MHz. As shown in Appendix D, compared with SRAM PUF based TRNG, our Latch PUF based TRNG reduces the area overhead (from 55k gates to 2480 gates) as well as the throughput (12.5 Mbps $>$ 4.688 Mbps). Further, the ring oscillator physical unclonable functions (RO PUF) based TRNG costs the lowest area resources among these PUF based TRNGs (only 750 gates), but cannot satisfy the throughput requirements. Hence, our Latch PUF-based TRNG is relatively more suitable for our masking scheme.

We applied the NIST tests [7] to examine the distribution of the random sequences. The test results in Appendix D show that the symmetrically placed XOR corrector has relatively higher entropy than the automatically placed XOR corrector, and the feed forward improves the randomness of sequences.

There are two experiments that can be used to evaluate the leakages of the sensitive variables in this study (shown in Appendix E). At first, we perform a first-order CPA attack to evaluate the Hamming weight leakage. This attack makes statistics about power consumption at a single point. Afterward, a Hamming distance leakage evaluation which correlates to the power consumptions of simultaneously updated registers was presented. The test results illustrate that compared to the original permutation tables masking scheme [1], our improved permutation tables masking scheme leaks relatively few information of sensitive variables in the Hamming distance model.

Conclusion. We proposed an improved permutation tables masking scheme to overcome the security issues in conventional permutation tables masking scheme. This method consists in (1) using a novel Latch PUF based TRNG with symmetry XOR corrector and feed forward to generate the uniformly distributed masks, and (2) randomly exchanging the inputs of S-boxes with a new permutation table. Furthermore, we exemplified this Latch PUF based TRNG on Xilinx SP3E FPGAs. The throughput of this TRNG is 12.5 Mbps in 25 MHZ clock which satisfies the requirements of masks in our masking scheme. NIST test results demonstrate that the random numbers generated by this TRNG are uniformly distributed. On the other hand, this solution has been evaluated within information-theoretic experiments, proving its security against first-order PAAs, the Pearson coefficient reduced from 0.8 to nearly 0. However, prior studies have demonstrated that the generation of permutation tables cost too many clock cycles. In the future works, we will find ways to solve the flaws existed in our design.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 61571116).

Supporting information Appendixes A–E. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Coron J. A new DPA countermeasure based on permutation tables. In: Proceedings of International Conference on Security and Cryptography for Networks, 2008. 278–292
2 Maghrebi H, Guilley S, Prouff E, et al. Register leakage masking using gray code. In: Proceedings of IEEE International Symposium on Hardware-Oriented Security and Trust, 2012. 37–42
3 Maes R. Physically Unclonable Functions: Constructions, Properties and Applications. Berlin: Springer, 2016
4 Li B, Chen S. A dynamic PUF anti-aging authentication system based on restrict race code. Sci China Inf Sci, 2016, 59: 012108
5 Hata H, Ichikawa S. FPGA Implementation of metastability-based true random number generator. IEICE Trans Inf Syst, 2012, 95: 426–436
6 Chen S, Li B, Zhou C J. FPGA implementation of SRAM PUFs based cryptographically secure pseudo-random number generator. Microprocess MicroSyst, 2018, 59: 57–68
7 Rukhin A, Soto J, Nechvatal J, et al. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Technical Report SP 800-22 Rev. 1a, 2010