SCIENCE CHINA
Information Sciences

• LETTER •

June 2022, Vol. 65 169102:1–169102:3
https://doi.org/10.1007/s11432-019-1469-7

A small first-order DPA resistant AES implementation with no fresh randomness

Man WEI[1,2,3], Siwei SUN[1,2,3*], Zihao WEI[1,2,3], Zheng GONG[4] & Lei HU[1,2,3]

[1]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093*, China;*
[2]*Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing* 100093*, China;*
[3]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing* 100049*, China;*
[4]*School of Compute Science, South China Normal University, Guangzhou* 510631*, China*

Received 10 April 2019/Revised 22 July 2019/Accepted 26 July 2019/Published online 21 May 2021

Citation Wei M, Sun S W, Wei Z H, et al. A small first-order DPA resistant AES implementation with no fresh randomness. Sci China Inf Sci, 2022, 65(6): 169102, https://doi.org/10.1007/s11432-019-1469-7

Dear editor,

Cryptographic algorithms exist in the form of software or hardware components inside security-critical information systems, which are often deployed in a hostile environment. Therefore, other than guaranteeing the security of the cryptographic algorithms in the black-box model, ensuring the security of the software or hardware components against the attacks exploiting the physical leakage of the underlying devices is considerably important. Among these attacks, the so-called differential power analysis (DPA) [1], which is one of the most powerful techniques, exploits the correlation between the instantaneous power consumption and intermediate values of a cryptographic algorithm. To counteract DPA, various countermeasures applied on different levels of a circuit have been proposed previously.

At ICICS 2006, Nikova et al. [2] proposed a masking scheme, called threshold implementation (TI), that is inherently immune against first-order DPA attacks even in the presence of glitches. The TI technique divides the input of a function into at least $t+1$ shares by Boolean masking and computes with a non-complete set of input shares such that sensitive values cannot be reconstructed using the knowledge of t shares, thereby resulting in provable security against first-order attacks. Moreover, herein, we mention the so-called domain-oriented masking (DOM) technique proposed by Gross et al. [3]. DOM uses only two input shares for first-order security to implement a function with any algebraic degree provided that the input variables of the unshared function are shared independently.

The block cipher advanced encryption standard (AES) is one of the most investigated cryptographic algorithms in TI because of its importance and popularity. Most TI schemes of AES are implemented in a serialized approach, whose S-box is built on the tower field architecture [4]. We chose to break the AES S-box into operations over $\mathbb{F}_{2^4}$ and present some new TIs for some subcomponents. To the best of our knowledge, no scheme with uniform output has been found when the TI technique was applied to the $\mathbb{F}_{2^4}$ multiplier and the $\mathbb{F}_{2^4}$ square scalar separately. Therefore, we consider the XOR of the $\mathbb{F}_{2^4}$ multiplier and the $\mathbb{F}_{2^4}$ square scalar (denoted by SqSc) as a single function Muls from $\mathbb{F}_{2^8}$ to $\mathbb{F}_{2^4}$: $y = x^m \times x^l + \mathrm{SqSc}(x^m + x^l)$, where x^m and x^l represent the high 4 bits and the low 4 bits of the 8-bit input x, respectively. Further, we try to construct a shared implementation for Muls that satisfies all the three properties of TI: correctness, non-completeness, and uniformity. The pseudo-code for searching a uniform sharing for Muls is shown in Appendix A. Fortunately, we find a uniform sharing, which also satisfies non-completeness and correctness:

$$\begin{aligned}
y_1 &= (x_3^m + x_4^m) \times (x_2^l + x_3^l) + \mathrm{SqSc}(x_2^m + x_2^l),\\
y_2 &= (x_1^m + x_3^m) \times (x_1^l + x_4^l) + \mathrm{SqSc}(x_1^m + x_1^l),\\
y_3 &= (x_2^m + x_4^m) \times (x_1^l + x_4^l) + \mathrm{SqSc}(x_4^m + x_4^l),\\
y_4 &= (x_1^m + x_2^m) \times (x_2^l + x_3^l) + \mathrm{SqSc}(x_3^m + x_3^l).
\end{aligned}$$

We propose a new TI for the $\mathbb{F}_{2^4}$ inverter, where the input and output are divided into four shares and two shares, respectively. The inverter is executed using four subcomponents $\mathrm{Inv}_{ij} : \mathbb{F}_{2^4} \times \mathbb{F}_{2^4} \times \mathbb{F}_{2^4} \to \mathbb{F}_{2^4}$ for $i, j \in \{1, 2\}$ (see Appendix B). To ensure the uniformity, Inv_{11}, Inv_{12} and Inv_{21}, Inv_{22} are computed in separate cycles. The recombination occurs after a register-stage in the next clock cycle and thus does not violate the non-completeness [5]. Certainly, each share of the subcomponent is individually uniform, which ensures that no first-order leakage occurs in the registers. Moreover, the outputs of the $\mathbb{F}_{2^4}$ inverter y_1, y_2 are jointly uniform.

The first-order DPA resistant implementation of AES. First, the linear operations of AES can be implemented using two or more shares that are processed independently to meet the requirement of first-order security. The implementation of the AES S-box, which somehow employs techniques

* Corresponding author (email: sunsiwei@is.ac.cn)

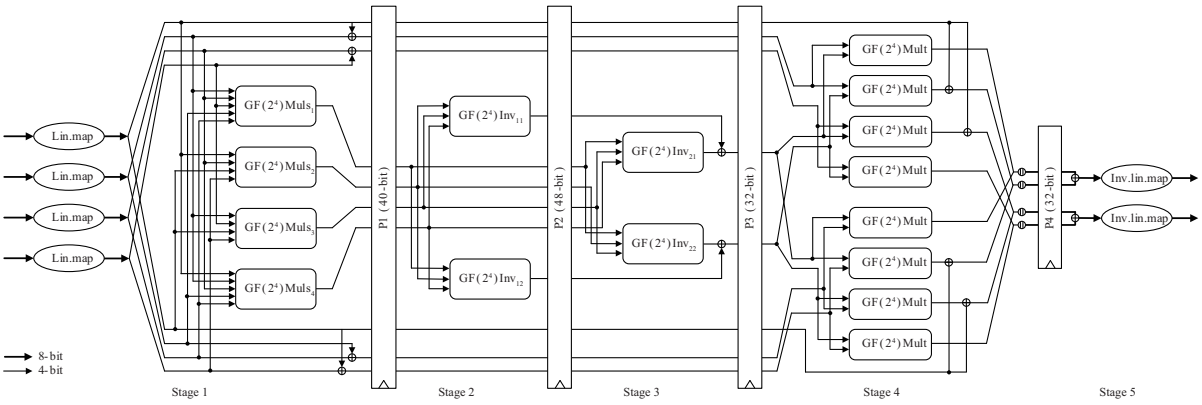

Figure 1 The TI design of the AES S-box.

from both TI and DOM uses four input shares and two output shares and demands no fresh randomness. The most challenging part of the implementation of the AES S-box is depicted in Figure 1.

Stage 1. The TI of the initial linear transformation can be realized using four shares as all TIs are linear functions, which is shown in Figure 1. Further, each 8-bit output of linear transformation is divided into two 4-bit parts. We consider the $\mathbb{F}_{2^4}$ multiplier and the $\mathbb{F}_{2^4}$ square scalar as a single function and implement it using the new TI sharing scheme presented above. The outputs of this stage are synchronized with registers before going into the next stage. Notably, some signals are XOR-ed to reduce the size of the registers, where the four-share inputs of the $\mathbb{F}_{2^4}$ multiplier are decreased to two shares, independent with the remaining first share.

Stage 2. We employ the scheme described earlier for the $\mathbb{F}_{2^4}$ inverter.

Stage 3. The $\mathbb{F}_{2^4}$ inverter is implemented in two stages and outputs two shares. The outputs are considered as the inputs of the last two multipliers together with other signals, which can be verified uniformly and independently.

Stage 4. The two $\mathbb{F}_{2^4}$ multipliers in this stage are implemented using the DOM technique. The fresh numbers employed for resharing are substituted by signals from stage 1, which is the key to eliminate the usage of fresh randomness. Because each 4-bit signal is independent with the inputs of the corresponding $\mathbb{F}_{2^4}$ multiplier by test, the reshared outputs satisfy uniformity. To avoid first-order leakages in the subsequent computation, the outputs of this stage are synchronized with registers before going into the next stage.

Stage 5. Finally, the output linear transformation is performed using two shares.

Hardware architecture. In our AES TI, we use an 8-bit serial architecture similar to [6] with necessary tweaks for the TIs, and the overall architecture is shown in Appendix C. There are three 128-bit registers in our design. Two of which are used to keep the intermediate values of the shared encryption, and one is for the key schedule algorithm, which is implemented in an unshared form.

However, the TI-based AES implementation requires 24 cycles to complete one round of our implementation. In the first 16 cycles, the XOR of the plaintext and the key is sent to the S-box. According to the previous discussion (see Figure 1), 5 cycles are required to complete the S-box computation; therefore, the output of the S-box is available after a delay of 4 cycles. Because there are 16 bytes in the state, $16 + 4 = 20$ cycles are required to complete addRoundKey and SubBytes. Moreover, the ShiftRows is performed in the 20th cycle while the state is loading the output bytes of the S-box. Then, the MixCols is performed in the last four cycles. Therefore, we need $24 \times 10 + 16 = 256$ cycles to complete one AES encryption.

The state for the key schedule algorithm is depicted in Appendix C. Apart from using an unshared S-box, the implementation of the key schedule largely follows [3] with necessary tweaks to tune it with the encryption process.

Our S-box operates on four-share input and outputs two shares. We can use the method proposed in [7] to extend two shares to four shares, where we choose locally-independent bytes m_1, m_2 (see Appendix C) from the state arrays to extend two shares a_1, a_2 to four shares b_1, b_2, b_3, b_4 as follows:

$$b_1 = m_1,\ b_2 = m_1 + a_1,\ b_3 = m_2,\ b_4 = m_2 + a_2.$$

Performance. We synthesized our design using Synopsys Design Compiler 2014.09-SP3 with the UMC 180 nm standard cell library, and the results are summarized in Appendix D. The results show that our realization costs 5850 GE and 256 clock cycles totally, whereas our shared AES S-box costs 2034 GE. Moreover, compared with the previous implementation that demands no fresh randomness [7], our implementation considerably reduces the number of clock cycles required for one encryption from 2804 to 256.

Leakage analysis. To prevent the tool from optimizing over module boundaries, we implement our design on the SAKURA-G board using Xilinx ISE with the "keep hierarchy" constraint. Total 10 million power traces covering the ninth and part of the tenth round of AES are collected for the design using a PicoScope 3203D oscilloscope sampling at 250 MS/s.

We also implement another design where the corresponding internal state is replaced with fresh randomness and perform leakage evaluations against these two designs to give a comparison. We evaluate the effectiveness of our designs by applying the test vector leakage assessment (TVLA) based on Welch's t-test (a.k.a non-specific t-test [8]), and the results obtained using 10 million traces are shown in Appendix E for TI realization and TI realization with PRNG. Therefore, we conclude that our AES TI is secured against standard first-order attacks under the condition of 10 million traces. Conversely, as shown in Appendix E, our TI exhibits obvious leakage using second-order analysis.

Conclusion. We provide new TIs for some components of the tower field architecture of the AES S-box. Based on

these new implementations, we present a new AES TI with reduced cost. In the future, it will help in investigating how to apply other recently developed techniques to further reduce the cost (e.g., the latency [9]) of the overall TI circuit.

Acknowledgements The work was supported by National Key R&D Program of China (Grant No. 2018YFB-0804402), Chinese Major Program of National Cryptography Development Foundation (Grant Nos. MMJJ20180102, MMJJ20180206), National Natural Science Foundation of China (Grant Nos. 61732021, 61802400, 61772519, 61802-399, 61572028), Youth Innovation Promotion Association of Chinese Academy of Sciences, and Project of Science and Technology of Guangzhou (Grant No. 201802010044).

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 Kocher P, Jaffe J, Jun B. Differential power analysis. In: Proceedings of Annual International Cryptology Conference, Santa Barbara, 1999. 388–397
2 Nikova S, Rechberger C, Rijmen V. Threshold implementations against side-channel attacks and glitches. In: Proceedings of International Conference on Information and Communications Security, Raleigh, 2006. 529–545
3 Gross H, Mangard S, Korak T. Domain-oriented masking: compact masked hardware implementations with arbitrary protection order. In: Proceedings of the ACM Workshop on Theory of Implementation Security, Vienna, 2016
4 Canright D. A very compact S-box for AES. In: Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems, Edinburgh, 2005. 441–455
5 Chen C, Farmani M, Eisenbarth T. A tale of two shares: why two-share threshold implementation seems worthwhile — and why it is not. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, 2016. 819–843
6 Moradi A, Poschmann A, Ling S, et al. Pushing the limits: a very compact and a threshold implementation of AES. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, 2011. 69–88
7 Wegener F, Moradi A. A first-order SCA resistant AES without fresh randomness. In: Proceedings of International Workshop on Constructive Side-Channel Analysis and Secure Design, Singapore, 2018. 245–262
8 Goodwill G, Jun B, Jaffe J, et al. A testing methodology for side-channel resistance validation. In: Proceedings of NIST Non-Invasive Attack Testing Workshop, Nara, 2011. 115–136
9 Li S, Sun S W, Li C Y, et al. Constructing low-latency involutory MDS matrices with lightweight circuits. IACR Trans Symmetric Cryptol, 2019, 1: 84–117