SCIENCE CHINA
Information Sciences

• LETTER •

May 2021, Vol. 64 159102:1–159102:3
https://doi.org/10.1007/s11432-018-9794-6

Unbalanced sharing: a threshold implementation of SM4

Man WEI[1,2,3], Siwei SUN[1,2,3*], Zihao WEI[1,2,3] & 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*

Received 3 November 2018/Accepted 20 February 2019/Published online 19 March 2021

Citation Wei M, Sun S W, Wei Z H, et al. Unbalanced sharing: a threshold implementation of SM4. Sci China Inf Sci, 2021, 64(5): 159102, https://doi.org/10.1007/s11432-018-9794-6

Dear editor,

In recent years, we have witnessed a rapid development of the side-channel attacks, which deviate from the traditional block-box model and exploit the leakages of cryptographic devices. Among those attacks, the so-called differential power analysis (DPA) exploiting the correlation between the instantaneous power consumption and the sensitive intermediate values of a cryptographic algorithm is one of the most powerful techniques [1]. Various countermeasures have been proposed in the literature to make the implementation of cryptographic algorithms immune to DPA. In this study, we focus on the approach called threshold implementation (TI) initially proposed by Nikova et al. [2] at ICICS 2006, which is provable secure against first-order DPA under certain leakage assumptions. The TI technique divides the input into several shares by Boolean masking, and the sharing scheme should satisfy three properties: correctness, non-completeness, and uniformity. Correctness states that the sum of output shares equals the correct output. Non-completeness requires each output share to be computed independent of at least one input share such that the unshared intermediate results are not revealed. Uniformity requires the input shares to be uniformly distributed. Moreover, for an iterative cryptographic algorithm, the output shares should also be uniform for subsequent computations. The TI technique has been widely applied to many symmetric-key cryptographic algorithms. After a series of work, the technique of TI has been largely extended and generalized.

The minimum number of shares required for a TI scheme is the algebraic degree of the function being shared plus 1 [2]. However, the uniform sharings for some degree-d functions with $d+1$ shares have not been found. Besides remasking with fresh randomness, here we provide another technique for achieving uniformity.

Unbalanced sharing. In existing sharing schemes, every input variable has the same number of shares. However, it is perfectly possible that a sharing scheme in which different input variables are split into different numbers of shares still maintains the three essential properties for TIs. Such a scheme is named an unbalanced sharing scheme. Taking the multiplication function $y = f(x^1, x^2) = x^1 x^2$ as an example, we split x^1 into 3 shares (x_1^1, x_2^1, x_3^1), and x^2 into 5 shares $(x_1^2, x_2^2, x_3^2, x_4^2, x_5^2)$, and the sharing is given by

$$\begin{aligned} y_1 &= \left(x_2^1 + x_3^1\right)\left(x_2^2 + x_3^2 + x_4^2 + x_5^2\right) + x_4^2 + x_5^2, \\ y_2 &= x_1^1\left(x_1^2 + x_3^2 + x_4^2 + x_5^2\right) + x_3^1 x_1^2 + x_5^2, \\ y_3 &= x_1^1 x_2^2 + x_2^1 x_1^2 + x_4^2. \end{aligned} \tag{1}$$

It is obvious that Eq. (1) satisfies the non-completeness property. For correctness, the reader can check that $f_1 + f_2 + f_3 = (x_1^1 + x_2^1 + x_3^1)(x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2)$. For uniformity, if the input masking is uniform, the distribution of output masking is also uniform. In general, the unbalanced sharing not only enlarges the design space of TIs, but also can be regarded as a new method for achieving uniformity. Therefore, we could have more candidate TI schemes for the given cryptographic algorithm, which typically leads to more flexible area-randomness trade-offs.

TI of the SM4 S-box. We apply the TI technique with unbalanced sharing to SM4, an internationally standardized block cipher. For any linear operation, it can be shared in a straight-forward way. Therefore, we only focus on the S-box in SM4. The SM4 S-box is affine equivalent to the AES S-box, which is affine equivalent to the inverse operation over $\mathbb{F}_{2^8}$ [3]. We also use the tower field approach [4] to divide the S-box into three pipeline stages with operations over $\mathbb{F}_{2^4}$. The concrete architecture of our design is shown in Figure 1.

Stage 1. Since the degree of the $\mathbb{F}_{2^4}$ multiplier is 2, the minimum number of input shares is 3 to achieve 1st-order security. Though there is no known TI with uniform output with 3 input shares. To reduce the area cost, we implement the $\mathbb{F}_{2^4}$ multiplier directly with 3 input shares and 3 output shares. The output of the multiplier and the output of the

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

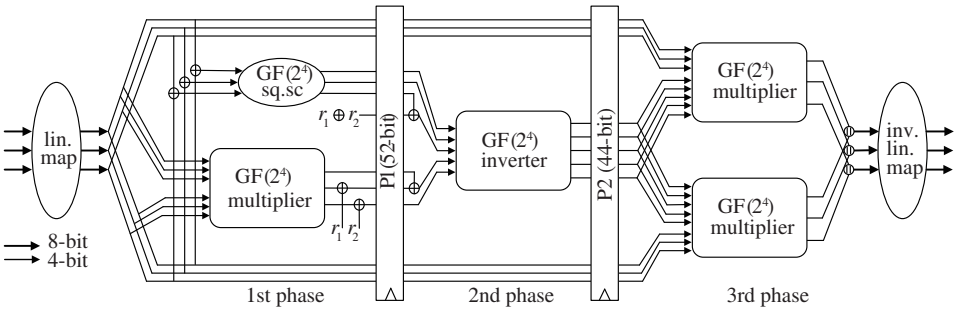

Figure 1 The TI of the SM4 S-box.

$\mathbb{F}_{2^4}$ square scalar (linear) form the input to the next stage. Uniformity is achieved with 8 random bits by partial remasking [5], and the concrete remasking scheme is depicted in Figure 1.

Stage 2. For the $\mathbb{F}_{2^4}$ inverter with algebraic degree 3, any uniform sharing has not been found using 4 shares unless the inverter is decomposed into three stages, which costs two more cycles and much more area. Therefore, we use the uniform sharing scheme proposed in [5] for the $\mathbb{F}_{2^4}$ inverter with 5 shares. Then the output of the $\mathbb{F}_{2^4}$ inverter and the former output of the input linear transformation are stored in the registers as the input to the two multipliers in the next stage. It can be verified by simulation that the inputs to the last two multipliers are uniform respectively.

Stage 3. We apply the unbalanced sharing scheme shown in Eq. (1) over $\mathbb{F}_{2^4}$ to the upper $\mathbb{F}_{2^4}$ multipliers in Figure 1, and a new unbalanced sharing with uniform output to the other $\mathbb{F}_{2^4}$ multipliers as follows:

$$\begin{aligned} y_1 &= \big(x_2^1 + x_3^1\big)\big(x_2^2 + x_3^2 + x_4^2 + x_5^2\big) + x_5^2, \\ y_2 &= x_1^1\big(x_1^2 + x_3^2 + x_4^2 + x_5^2\big) + x_3^1 x_1^2 + x_4^2 + x_1^1, \\ y_3 &= x_1^1 x_2^2 + x_2^1 x_1^2 + x_4^2 + x_5^2 + x_1^1. \end{aligned}$$

Although the three 4-bit output shares of each $\mathbb{F}_{2^4}$ multiplier are uniform respectively, the concatenated three 8-bit output shares are not uniform anymore. However, we verified by simulation that each share individually is uniform, which ensures that there is no first-order leakage in the following registers [5]. Besides, for Feistel ciphers such as SM4, the output of the S-box is then XORed with the former byte, which is not involved the computation of this S-box. In summary, if the input is uniformly shared, the round output is always uniform.

Hardware architectures. SM4 is a 32-round iterative unbalanced Feistel block cipher with 128-bit block size and 128-bit key. Our design is based on architectures proposed in [6] with necessary tweaks to make it suitable for our TI construction. We use a small width for process unit to make our SM4 implementation compact. And to reduce clock cycles, we implement the round key expansion and encryption in parallel. In this study, we only apply the TI technique to the encryption process. The key schedule uses the similar iteration structure in a nonshared form. The detailed process can be referred in Appendix A.

The serialized encryption architecture uses only one shared S-box. The data path has a 16-bit width, and operates with two shares except for the shared S-box and the linear transformation L, which process three-share data. Thus the inputs of the shared S-box and outputs of the linear transformation L need more attention later.

Our implementation requires 6 clock cycles to complete one round. In the first 4 clock cycles, data enters the shift register arrays, and is selected to perform the XOR operations with the round key. Its output is stored in register from the 4th to the 6th clock cycles, since 3 clock cycles are required to complete the TI-based S-box computation. After the shared S-box is operated 4 times, the forth output together with the former three outputs performs the L linear transformation.

From the above, $6 \times 32 = 192$ cycles are required to complete one SM4 encryption. Key expansion has the same timing flow. To get the round key ready for the encryption round, key expansion is kept one round (6 clock cycles) ahead of encryption. The constant key CK_i is generated dynamically. Thus, our complete implementations needed 198 clock cycles totally.

We synthesized the architecture of the serialized SM4 TI design with synopsys design compiler 2014.09-SP3 and UMC 180 nm standard cell library. We apply the compile and compile_ultra commands to each component of the implementation. The results in Appendix B show that our shared SM4 S-box with unbalanced sharing only costs 2000 GE, and our realization costs 7316 GE totally. Besides, our SM4 TI only needs 8-bit randomness per S-box, less than most AES TIs.

Leakage analysis. The security evaluation is performed with the SAKURA-G board, which includes a main FPGA (Xilinx Spartan-6 XC6SLX75) and a control FPGA (Xilinx Spartan-6 XC6SLX9). Our design is implemented on the main FPGA, while the control FPGA manages the communication between the device under test (DUT) and the PC. The power traces covering the last two rounds of SM4 are sampled at 500 MS/s with an oscilloscope. In our implementation, we use a pseudo-random number generator (PRNG) implemented on the FPGA to provide fresh randomness. The PRNG is asynchronously active with the encryption algorithm to minimize its side-effect on the leakage analysis.

We collect 1 million power traces when the PRNG is off, and 10 million power traces when the PRNG is on. To detect leakages, we apply the well-known non-specific t-test [7], which can detect leakages for any order by examining the t-values. The first-order and second-order t-test results can be referred to Appendix C. When the PRNG is off, the implementation suffers the first-order leakages only under 1 million power traces. Therefore, we conclude that our TI of SM4 is secure against standard first-order attacks under the condition of 10 million traces. On the other hand, our TI exhibits obvious leakage with respect to second-order analysis.

Conclusion. By breaking the balance between the numbers of shares for different input variables of a TI, we introduce the so-called unbalanced sharing technique. The unbalanced sharing is intrinsically versatile than standard sharing

schemes since it imposes no condition on the relationship between the numbers of shares for different variables, and therefore offers more possibilities for secure implementations and area-randomness trade-offs. We demonstrate the usefulness of the unbalanced sharing technique by presenting a TI of SM4 whose security against first-order DPA attacks are confirmed by performing leakage analysis on a real FPGA implementation. In the future, it is interesting to investigate how to apply similar techniques to higher-order TIs with reduced resource consumption.

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 No. MMJJ20180102), National Natural Science Foundation of China (Grant Nos. 61732021, 61802400, 61772519, 61802399), and Youth Innovation Promotion Association of Chinese Academy of Sciences.

Supporting information Appendixes A–C. 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 Liu F, Ji W, Hu L, et al. Analysis of the SMS4 block cipher. In: Proceedings of Australasian Conference on Information Security and Privacy, Townsville, 2007. 158–170
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 Bilgin B, Gierlichs B, Nikova S, et al. Trade-offs for threshold implementations illustrated on AES. IEEE Trans Comput-Aided Des Integr Circ Syst, 2015, 34: 1188–1200
6 Shang M, Zhang Q L, Liu Z B, et al. An ultra-compact hardware implementation of SMS4. In: Proceedings of International Congress on Advanced Applied Informatics, Kokura Kita-ku, 2014. 86–90
7 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