TBOX-Based Mask Scrambling Against SCA

João Carlos Resende, Ricardo J. R. Maçãs and Ricardo Chaves
Instituto Superior Técnico, Universidade de Lisboa / INESC-ID
Rua Alves Redol 9, 1000-029 Lisbon, Portugal
Email: {joaocresende, ricardo.macas, ricardo.chaves}@tecnico.ulisboa.pt

***Abstract*—In the last years Side-Channel Attacks have become a significant threat against security devices. Given this, several countermeasures have been proposed, ranging from reducing the leaked power consumption to masking schemes. However, these solutions imply a cost, typically in terms of resources, performance, and power consumption. This work re-adapts the masking scheme of Block Memory Content Scrambling (BMS) to the AES Look-up tables for System-on-Chip (SoC) FPGAs, namely the SmartFusion 2 FPGA. The solution is further improved resource-wise by making use of the embedded ARM Cortex-M3 processor for updating the masks.**

The AES algorithm is a 128-bit block cipher, processed over $N = 10 \vee 12 \vee 14$ rounds, depending on the key size. The 128-bit block (processed as a 4x4 byte matrix) is named *State*. Each round is composed of the following sequence of operations: **ShiftRows; SubBytes; MixColumns** and; **AddRoundkey**. **ShiftRows** is a byte-level permutation of the *State*, and **AddRoundKey** is a 16-byte-wide XOR operation with the corresponding Round Key. **SubBytes** and **MixColumns** operations can be combined into a 1B-by-4B lookup table, named TBox, followed by a 4-to-1 32b XOR-trees [1].

In [2], *Güneysu and Moradi* introduce the idea of applying BMS in order to randomly permute the TBoxes' contents, effectively masking their operation. BMS consists of two steps: mask the 32-bit output of each 16 TBoxes with a different random Sub-Mask (SUBM); offset the content's address of each TBox (*Scrambling*) with one of 16B of the Data Mask (DM), that relates to SUBM through $DM[x][y]\big|_{8b} = \bigoplus_{n=0}^{3} SUBM[4x+n][y]\big|_{8b}$.

This way, BMS provides masking security by mixing (XOR) the DM with the *State*. The cipher algorithm remains correct as the Masked TBoxes (MTB^i) will perform *SubBytes+MixColumns* on the State, while preserving the DM for the next round, as: $MTB^i(s_{x,y}\big|_{8b} \oplus DM[x][y]\big|_{8b}) = SUBM[i]\big|_{32b} \oplus TB(s_{x,y})\big|_{32b}$, but adding de-correlating entropy to the power consumption profile.

A Final Mask is used in the last round, due to the different XOR combination by: $FM[x][y]\big|_{8b} = \bigoplus_{n=0}^{3} SUBM[4x+y][n]\big|_{8b}$.

The original BMS design [2] implements a parallel *Scrambling* circuit to allow for a progressive update of TBox masks through the use of switchable address spaces within the BRAMs. However, this solution has two drawbacks. Not only does the *Scrambler* requires extra resources aside from the AES core, but in order to work, it also requires the BRAMs to be pre-initialized. Pre-initialization is a non-standard feature on Xilinx FPGAs, that is not easily exportable to other technologies. In the work herein proposed, resources are freed by removing the *Scrambler* and delegate its function to the embedded ARM Cortex-M3 processor included in a Microsemi SmartFusion2 SoC (M2090TS), which is a common feature of most recent SoCs. The latter becomes therefore responsible for the initialization and update of the BRAMs/TBoxes (SUBM;DM;FM).

The addition/removal of the DM/FM into the State, omitted in [2], is performed by mixing the DM and FM directly to the initial and last round key respectively ($DM \oplus RK^0$; $FM \oplus RK^N$). This can also be done by the processor, since it has access to the Round Keys memory bank. With this approach, the actual AES structure does not change, being identical to the unprotected version, thus not impacting the available resources and operating frequency.

The obtained results for the developed structure, on a M2S090TS-1FG484 chip using Libero SoC v12.2 SP3, suggest a maximum throughput of 1856 Mbps (@145MHz) at a cost of 1318 LUTs and 16 BRAMs, for both protected and unprotected designs. The similar state of the art solutions [2], suggest a significantly higher area cost and reasonably lower efficiency metrics (Mbps/LUT) even though implemented on more powerful FPGA technologies.

These results show that with the presented AES structure and given the characteristics of this technology, BMS masking can be deployed at no additional area or frequency cost.

Acknowledgment

This work was supported by national funds through FCT, Fundação para a Ciência e a Tecnologia, under project UIDB/50021/2020 and the ERDF, European Regional Development Fund, under project LISBOA-01-0145-FEDER-031901 (PTDC/CCI-COM/31901/2017, HiPErBio).

References

[1] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J. Legat, "Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications," in *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, vol. 2. IEEE, 2004, pp. 583–587.

[2] T. Güneysu and A. Moradi, "Generic side-channel countermeasures for reconfigurable devices," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2011, pp. 33–48.

2576-2621/20/$31.00 ©2020 IEEE
DOI 10.1109/FCCM48280.2020.00039