VOL. 11, NO. 17, SEPTEMBER 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

TRADE-OFFS FOR THRESHOLD IMPLEMENTATIONS ILLUSTRATED ON ADVANCED ENCRYPTION STANDARDS

Abhilasha Kumari and Maflin Shaby
Sathyabama University, Chennai, India
E-Mail: Abhilasha0002@gmail.com

ABSTRACT

In this growing information age there is an increase in the cryptographic technology but this also has lead to numerous legal issues as well. A cryptographic device or mechanical encryption/decryption embedded devices can be easily attacked by hackers. Threshold implementation on data is one of the ways to make it secure against the first order power analysis attacks. This can be performed on both hardware and software also this is different from masking as TI provides protection from the glitches in the circuit. The existing system consists of two methods viz. raw implementation and adjusted implementation and the proposed one is nimble implementation which shows the trade-offs between the circuit area and the randomness. In this paper we compare the existing and the proposed algorithm to conclude a better method.

Keywords: Threshold implementation, encryption, trade-offs, proposed algorithm

1. INTRODUCTION

AES or advanced encryption standards is capable of encrypting and decrypting of electronic data or information .There are many applications of AES like in ATM machines, wireless communication, mobile phones, image processing, network security and many more. It's chosen by the NSIT (national institute of standard and technology) in 2001 and has its origin in the Rijndael block cipher. It's a block cipher that coverts clear text block data of 128bits, 192bits or 256 bits into cipher text(the original data or text to be encrypted using key) blocks of same length. The key required is again of the same respective lengths of 128, 192, 256 bits. The AES encryption standard is completely based on pure combinational circuitry. In this paper we investigate both existing as well as the new architecture that leads towards the most optimum composite field AES S-box constructions. The encryption algorithm in this is a set of iterations known as round transformations. AES-Rijndael has become the most preferred choice over all the cryptosystems (DES, RSA etc...) because it provides high security over them which are the most important factor when it comes to applications like e-commerce, military field and many more. This is achieved by using both FPGA and ASIC technology. For a key length of 128 bits there are 10 rounds. According to studies the non linear sub byte transformations is the best way for achieving both small area and high speed VLSI implementations.

1.1 Existing systems

The existing systems perform the 8 bits Galois field inversion of the s-box using sub fields of 4 bits and 2 bits. This approach minimizes the chip area required and the circuitry required for s-box. The best case of this reduces the number gates required in s box by 20%.

There are two algorithms in the existing system. These are raw implementation and adjusted implementation. Let's have a look at the raw implementation.

In raw implementation, we have four shares of input and the number of keys used is only 2 that is for every two input shares there is one key which is XORed before the s-box operation. There are two methods of doing this , the first one required 4 input shares which is the minimum number of shares required for a uniform implementation class C4 282, and decomposing the function into three uniform sub functions as Inv(x)=F(G(H(x))).the second method is done by using five shares without any decomposition. The area required for both the cases is same the difference lies in the number of clock cycles. The version with five shares is more preferred since it reduces the number of logic gates used when compared to the other method.

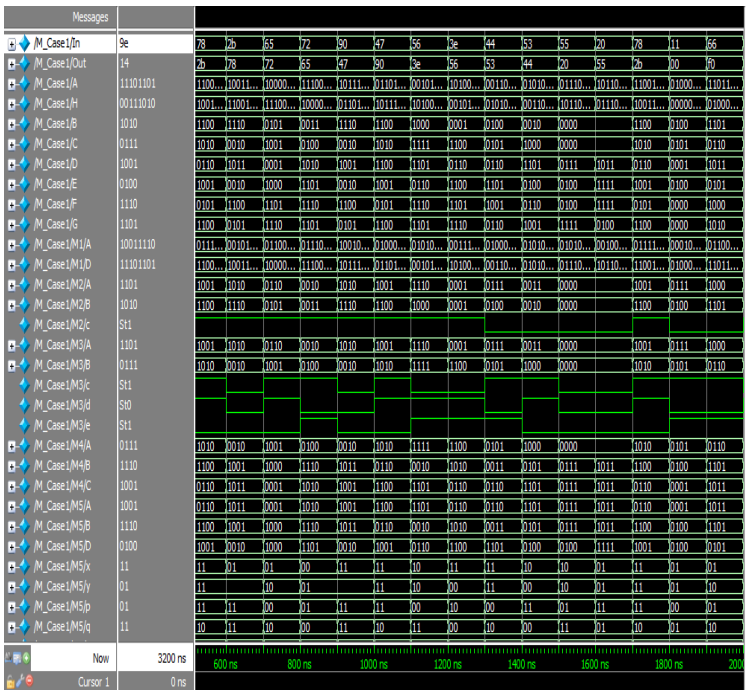
The difference between the raw and the adjusted implementation is that the adjusted one requires at least three shares for all the blocks including the linear operations in the s-box. This s box requires 44 bits of randomness per iteration. Each of the existing three shares is XORed with a random byte and the sum of these random bytes is taken as the fourth share. This also ensures uniformity of the S-box input. Together with the state, the number of shares for Mix Columns and Key XOR increases to three.

This version works on three shares for both the state and the key schedule which increases the area significantly. The S-box still requires four input shares and outputs three shares, hence the register P0 is reduced to 8-bits (one share) and the register P3 is not required. Similar to the raw implementation, we use 24-bits of randomness to increase the number of shares from three to four one cycle before the S-box, i.e. each of the existing three shares is XORed with a random byte and the sum of these random bytes is taken as the fourth share. This also ensures uniformity of the S-box input. Together with the state, the number of shares for Mix Columns and Key XOR increases to three.

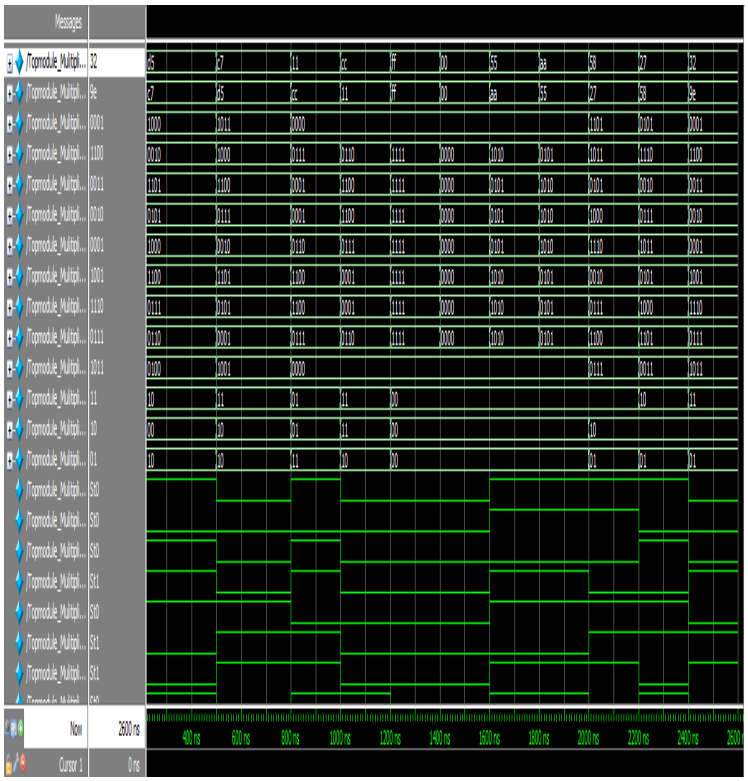
In adjusted we need 44 fresh random bits per S-box operation including increasing the number of shares of the S-box input. We use 24-bits of randomness to increase

the number of shares from three to four one cycle before the S-box, however raw requires only 32-bits of extra randomness per S-box calculation, including increasing the number of shares for the S-box input.

1.3 Raw implementation



1.2 Adjusted implementation



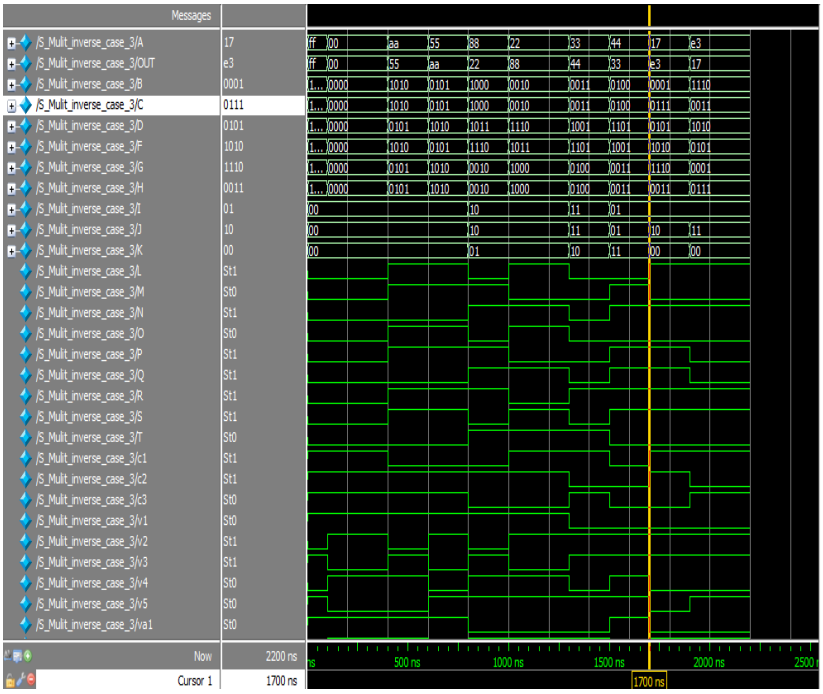
1.4 Proposed system

1.4.1 Our proposed algorithm is nimble implementation

Similar to the raw implementation, this one also uses two shares for the state and key arrays. The main difference is that the S-box needs three input shares instead of four. Hence the size of the register P0 is reduced to 8-bits (one share). As a result, we need only 16- bits of randomness to increase the number of shares from two to three before the S-box operation, i.e. each share is XORed with one byte of randomness and the XOR of the random bytes is

This construction requires only 32-bits of extra randomness per S-box calculation, including increasing the number of shares for the S-box input. We use fresh randomness at the end of the 1st phase to satisfy uniformity during the combination of the square scalar's and the multiplier's outputs, and after the inverter to break the dependency between the inputs of the multipliers in the 3rd phase. Since these re-masking steps conserve the uniformity property and the security of each block is achieved only by the correctness and non-completeness properties, we can discard the uniformity property and implement these nonlinear functions with the smallest number of shares n s. t. n > d, i.e. n = d + 1, where d is the degree of the unshared functions. We use the sharing with three input and output shares for each term of the multiplier and the sharing with four input and output shares for each term of the inverter.

2.2 Nimble implementation



1.5 Drawbacks of existing systems

One of the disadvantages of existing system is that it comes with longer critical path. The area of occupancy is high. The speed is also less with more delay.

1.6 Proposed system advantages

Its area efficient with high throughput. It's more secure compared to existing systems. This provides digital information security that is protecting information, information systems from unauthorized access, use, disclosure, disruption, modification, inspection, perusal, recording or destruction.

2. SYNTHESIS REPORT

2.1 Raw implementation

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	84	7,168	1%
Logic Distribution			
Number of occupied Slices	44	3,584	1%
Number of Slices containing only related logic	44	44	100%
Number of Slices containing unrelated logic	0	44	0%
Total Number of 4 input LUTs	84	7,168	1%
Number of bonded IOBs	16	141	11%
Total equivalent gate count for design	519		
Additional JTAG gate count for IOBs	768		

2.3 Adjusted implementation

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	67	7,168	1%
Logic Distribution			
Number of occupied Slices	35	3,584	1%
Number of Slices containing only related logic	35	35	100%
Number of Slices containing unrelated logic	0	35	0%
Total Number of 4 input LUTs	67	7,168	1%
Number of bonded IOBs	16	141	11%
Total equivalent gate count for design	411		
Additional JTAG gate count for IOBs	768		

2.4 Nimble implementation

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	62	7,168	1%
Logic Distribution			
Number of occupied Slices	33	3,584	1%
Number of Slices containing only related logic	33	33	100%
Number of Slices containing unrelated logic	0	33	0%
Total Number of 4 input LUTs	62	7,168	1%
Number of bonded IOBs	16	141	11%
Total equivalent gate count for design	372		
Additional JTAG gate count for IOBs	768		

3. CONCLUSIONS

We show that it is possible to achieve first-order DPA resistance with non-uniform shared functions if re-masking is applied properly. In the case of AES, our - non-uniform‖ nimble implementation requires less randomness than our - uniform‖ raw implementation, due to the decreased number of shares. However, for other algorithms and other S-boxes, re-masking may increase the amount of randomness required. This idea can be used to trade-off between the randomness and area requirements. Moreover, we empirically confirm that increasing the number of shares has a significant impact on the performance of higher-order attacks, which provides another trade-off between area and DPA resistance. Our most efficient implementation is approximately 8k GE small and requires only 32 bits of fresh randomness per S-box calculation, which is a significant improvement over all previous works.

REFERENCES

S. Mathew, F. Sheikh, A. Agarwal, M. Kounavis, S. Hsu, H. Kaul, M. Anders, and R. Krishnamurthy. 2010. 53 Gbps native GF (24)2 composite- field AES-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors. in Proc. IEEE Symp. VLSI Circuits (VLSIC). pp. 169-170.

V. Rijmen. 200. Efficient implementation of the Rijndael S-box. [Online]. Available: http://ftp.comms.scitech.susx.ac.uk/fft/crypto/rijndael-sbox.pdf.

Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao and P. Rohatgi. 2001. Efficient rijndael encryption implementation with composite field arithmetic. in Proc. CHES. pp. 171-184.

J.Wolkerstorfer, E. Oswald and M. Limburger. 2002. An ASIC implementation of the AES S-boxes. In: Proc. RSA Conf. pp. 67-78.

Satoh, S. Morioka, K. Takano, and S. Munetoh. 2000. A compact Rijndael hardware architecture with S-box optimization. in Proc. ASIACRYPT. pp. 239-245.

N. Men tens, L. Bateman, B. Greenland and I. Verbauwhede. 2005. A systematic evaluation of compact hardware implementations for the Rijndael S-box. In: Proc. Topics Cryptology (CT-RSA). 3376: 323-333.

D. Canright. 2005. A very compact Rijndael S-box. Naval Postgraduate School, Monterey, CA, Tech. Rep. NPS-MA-04-001.

X. Zhang and K. K. Parhi. 2006. On the optimum constructions of composite field for the AES algorithm. IEEE Trans. Circuits Syst. II, Exp. Briefs. 53(10): 1153-1157.

X. Zhang and K. K. Parhi. 2004. High-speed VLSI architectures for the AES algorithm. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 12(9): 957-967.

C. Paar. 1995. Some remarks on efficient inversion in finite fields. in Proc. IEEE ISIT. pp. 5-8.

J. L. Fan and C. Paar. 1997. On efficient inversion in tower fields of characteristic two. in Proc. IEEE ISIT. p. 20.

D. R. Wilkins. 2000. Part III: Introduction to Galois Theory.

M. M. Wong and M. L. D. Wong. 2010. A new common sub expression elimination algorithm with application in composite field AES S-box. in Proc. 10th Int. Conf. Inf. Sci. Signal Process. Their Appl. (ISSPA). pp. 452-455.

M. Chen. 1998. In Greedy Algorithms. in A Greedy Algorithm with Look Forward Strategy. Vienna, Austria: IN-TECH. pp. 1-16.