An Efficient FPGA Implementation of Advanced Encryption Standard Algorithm

DULLA BHUCHANDRASEKHAR [1] A. RAJA SEKHAR REDDY [2]
dullabhuchandrasekhar@gmail.com[1] ARSREDDY403@gmail.com[2]
[1]UG Scholar, Dept of ECE, Nalanda Group of Institutions, Kantepudi, Sattenapalli, Guntur, A.P.
[2]Assistant Professor, Dept of ECE, Nalanda Group of Institutions, Kantepudi, Sattenapalli, Guntur, A.P.

Abstract: Embedded cryptographic devices are vulnerable to power analysis attacks. Threshold implementations (TIs) provide provable security against first-order power analysis attacks for hardware and software implementations. Like masking, the approach relies on secret sharing but it differs in the implementation of logic functions. While masking can fail to provide protection due to glitches in the circuit, TIs rely on few assumptions about the hardware and are fully compatible with standard design flows. We investigate two important properties of TIs in detail and point out interesting trade-offs between circuit area and randomness requirements. We propose two new TIs of AES that, starting from a common previously published implementation, illustrate possible trade-offs. We provide concrete ASIC implementation results for all three designs using the same library, and we evaluate the practical security of all three designs on the same FPGA platform. Our analysis allow us to directly compare the security provided by the different trade-offs, and to quantify the associated hardware cost.

Keywords-AES, encryption, decryption, Field Programmable Gate Array (FPGA).

I. Introduction

The mass deployment of pervasive devices promises many benefits such as lower logistic costs, higher process granularity, optimized supply-chains, or location based services among others. Besides these benefits, there are also many risks inherent in pervasive computing: many foreseen applications are security sensitive, such as wireless sensor networks for military, financial or automotive applications. With the widespread presence of embedded computers in such scenarios, security is a striving issue, because the potential damage of malicious attacks also increases. An aggravating factor is that pervasive devices are usually not deployed in a controlled but rather in a hostile environment, i.e., an adversary has physical access to or control over the devices. This adds the whole field of physical attacks to the potential attack scenarios. Most notably are here so called side-channel attacks, especially Simple, Differential and Correlation Power Analysis. Smart cards and other types of pervasive devices performing cryptographic operations are seriously challenged by side-channel cryptanalysis. Several publications, e.g., [12] have stressed that such physical attacks are an extremely practical and powerful tool for recovering the secrets of unprotected cryptographic devices. In fact, these attacks exploit the information leaking through physical side channels and involved in sensitive computations to reveal the key materials. Amongst the known sources of side channels and the corresponding attacks most notable are power analysis attacks [18]. Many different kinds of power analysis attacks, e.g., simple and differential power analysis (SPA and DPA) [18], template-based attacks [2], and mutual information analysis [14], have been introduced while each one has its own advantages and is suitable in its special conditions. However, correlation power analysis (CPA) [6], which is a general form of DPA, got more attention since it

is able to efficiently reveal the secrets by comparing the measurements to the estimations obtained by means of a theoretical power model which fits to the characteristics of the target implementation.

II. LITERATURE SURVEY

Higher-order glitches free implementation of the AES using secure multi-party computation protocols

Higher-order side channel attacks (HO-SCA) is a powerful technique against cryptographic implementations and the design of appropriate countermeasures is nowadays an important topic. In parallel, another class of attacks, called glitches attacks, have been investigated which exploit the hardware glitches phenomena occurring during the physical execution of algorithms. We introduce in this paper a circuit model that encompasses sufficient conditions to resist glitches effects. This allows us to construct the first countermeasure thwarting both glitches and HO-SCA attacks. Our new construction requires Secure Multi-Party Computation protocols and we propose to apply the one introduced by Ben'Or et al. at STOC in 1988. The adaptation of the latter protocol to the context of side channel analysis results in a completely new higher-order masking scheme, particularly interesting when addressing resistance in the presence of glitches. An application of our scheme to the AES block cipher is detailed.

On the simplicity of converting leakages from multivariate to univariate (case study of a glitch-resistant masking scheme)

Several masking schemes to protect cryptographic implementations against side-channel attacks have been proposed. A few considered the glitches, and provided security proofs in presence of such inherent phenomena happening in logic circuits. One which is based on multi-party computation protocols and utilizes Shamir's secret sharing scheme was presented at CHES 2011. It aims at providing security for hardware implementations – mainly of AES – against those sophisticated side-channel attacks that also take glitches into account. One part of this article deals with the practical issues and relevance of the aforementioned masking scheme. Following the recommendations given in the extended version of the mentioned article, we first provide a guideline on how to implement the scheme for the simplest settings. Constructing an exemplary design of the scheme, we provide practical side-channel evaluations based on a Virtex-5 FPGA. Our results demonstrate that the implemented scheme is indeed secure against univariate power analysis attacks given a basic measurement setup. In the second part of this paper we show how using very simple changes in the measurement setup opens the possibility to exploit multivariate leakages while still performing a univariate attack. Using these techniques the scheme under evaluation can be defeated using only a moderate number of measurements. This is applicable not only to the scheme showcased here, but also to most other known masking schemes where the shares of sensitive values are processed in adjacent clock cycles.

III. Existing Work

Low-power low-area implementations of the AES have been reported requiring 3100 GE and 160 clock cycles and requiring 3400 GE and 1032 clock cycles. Both implementations use an 8-bit serialized data path and implement only a quarter of the Mix Columns operations. The first design, implements two S-boxes and performs the data path and the key schedule operations in parallel, while the latter implementation is fully serial and uses a RAM-like architecture. Canright has investigated very thoroughly how to

implement the AES Sbox in hardware with minimal area requirements [8]. On the other hand, several masking schemes have been proposed to create a masked AES S-box using either multiplicative or additive approaches. A common approach is to use the tower- field representation for an additive masking scheme because of the linearity of the inversion in GF (22).However, as expected its hardware implementation still has first-order leakage.

First contribution is a description of the smallest hardware implementation of AES known to date. Our design goal was solely low area, and thus we were able to set the time-area and the power-area tradeoffs differently, and in favor for a more compact hardware realization. To pursue our goal, we have taken a holistic approach that optimizes the total design, not every component individually. In total we achieved an implementation that requires only 2400 GE and needs 226 clock cycles, which is to the best of our knowledge 23% smaller than any previously published implementations. As a second contribution, we investigate side-channel countermeasures for this lightweight AES implementation. It turns out that when using Can right's representation, the only non-linear function is the multiplication in GF(22). Building on these findings, we applied the countermeasure to our unprotected AES implementation. For this architecture we conducted a complete side-channel evaluation based on real-world power traces that we obtain from SASEBO. We use a variety of different power analysis attacks to investigate the achieved level of resistance of our implementation against first order DPA attacks even if an attacker is capable of measuring 100 million power traces.

IV. Proposed Work

A serial implementation for round operations and key schedule as proposed which requires only one S-box instance and loads the plaintext and key byte-wise in row-wise order. We also use one Mix Columns instance that operates on the whole column and provides an output in one clock cycle. Due to this extreme serialization, one round requires at least 21 clock cycles even for the unprotected implementation. All our TIs execute one round in 23 clock cycles. In the first 16 clock cycles, the plaintext is XORed with the key and sent to the S-box. Its output will be taken from the third to the 18th clock cycles and stored in the state registers, i.e., the S-box is executed in three clock cycles. The Shift Rows operation is performed in the 19th clock cycle followed by four cycles of Mix Columns calculation. The S-box takes its input from the key schedule for four cycles starting from the 18th cycle. In the 17th, 22nd, and 23rd clock cycles, the S-box inputs and unused random bits are set to 0. Therefore, the calculation of AES takes 23 × 10 + 16 = 246 clock cycles, including 16 cycles to output the cipher text.

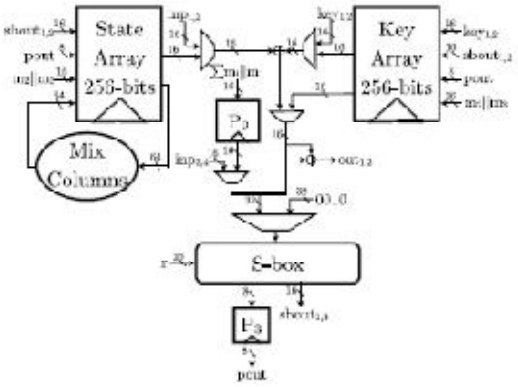


Fig. 1. Schematic of the serialized TI of raw AES-128.

1) Raw Implementation: We use two sets of state registers, each consisting of sixteen 16-bit registers, corresponding to the two shares of the state. The Mix Columns and the Key XOR operations are also performed with two shares.

As the key and the state registers are 256 bits implying the two shares. This TI of the S-box (details will be given in the following section)

requires four input shares, therefore, we initially share the plaintext in four shares. We share the key in two shares and XOR them with two of the plaintext shares before the S-box operation. More details about the key scheduling will be given later in this section. Besides the shared input, the S-box needs 20-bits of randomness *r*. whereas the remaining share sbout3 is written to register P3.

The data in the state registers are shifted to the left for the following 16 cycles so that the next output of the S-box can be stored in the same registers. During this shift, the data in P3 is XORed with the second share of the S-box output, which is in the state register S33, to reduce the number of shares from three to two. To achieve this signal, sig2 is active from the fourth to the 19th clock cycle.

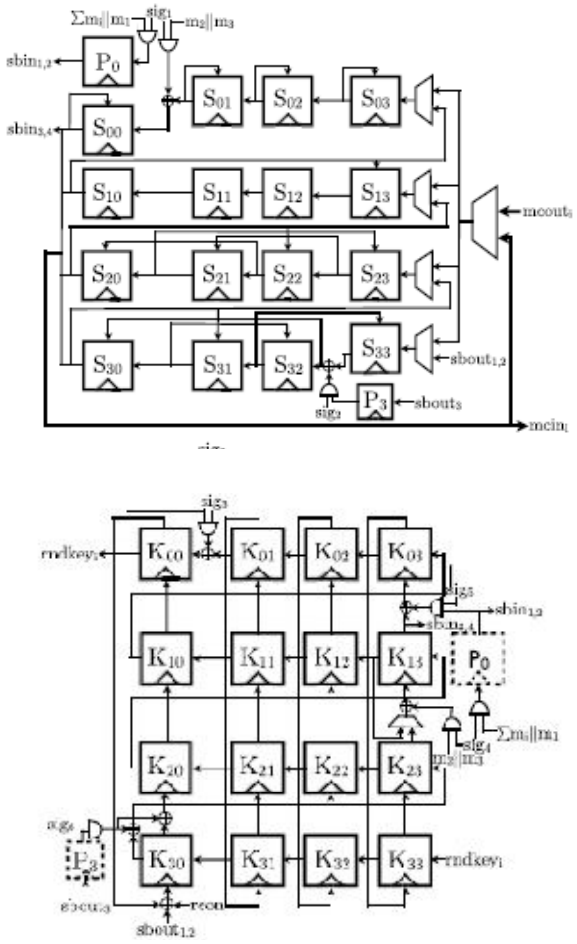


Fig. 2. Schematic of the state (top) and key (bottom) arrays for our raw implementation where S*i*, K*i*, and P0 hold two shares and P3 holds one share. The registers P0 and P3 are used by the state and the key array. The XOR of the value in P3 and S33 (resp. K30) is on one share of the value in register S33 (resp. K30) whereas all the other combinational operations are on two shares.

The S-box implementations use the tower field approach up to GF*(*22*)* for a small implementation. Therefore, the only nonlinear operation is GF*(*22*)* multiplication which must be followed by registers and remasking to avoid first order leakages. Also chose to use the tower field approach, however, decided to go until GF *(*24*)* instead of GF*(*22*)*. With this approach, the GF*(*24*)* inverter can be seen as a four bit permutation and the GF*(*24*)* multiplier as a four-bit multiplication. Therefore, we can find uniform TIs for each of these nonlinear functions. This might allow us to reduce the number of fresh random bits needed since we will have fewer nonlinear blocks compared to hence possibly require less remasking in order to use their outputs.

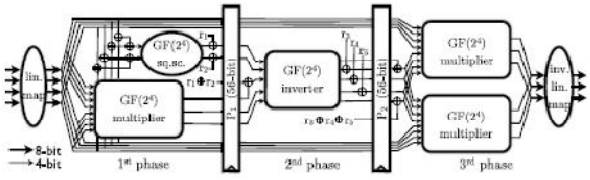


Fig. 3. S-box of the raw implementation.

Even though it is enough to use only two shares for linear operations, we sometimes chose to work on more than two shares to avoid the need of extra random bits. The linear map of the tower-field S-box operates on four shares since the multiplication needs four input shares. The inverter requires five input shares and the multiplication outputs only three shares, therefore, use two shares for the square scalar to have five shares in the beginning of the second phase. Use three shares for the inverse linear map of the tower-field S-box since the multiplication outputs three shares. For all the linear operations, the

shared functions are created as instantiations of the unshared function for the first share and as unshared function without the constant term for the other shares.

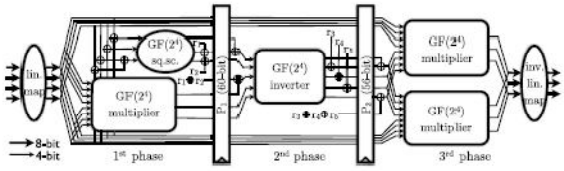


Fig. 4. S-box of the adjusted implementation.

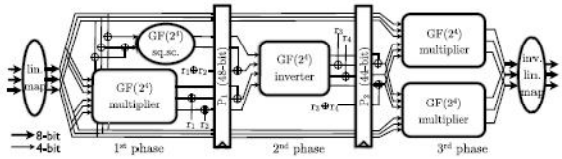


Fig. 5. S-box of the nimble implementation.

Adjusted Implementation:

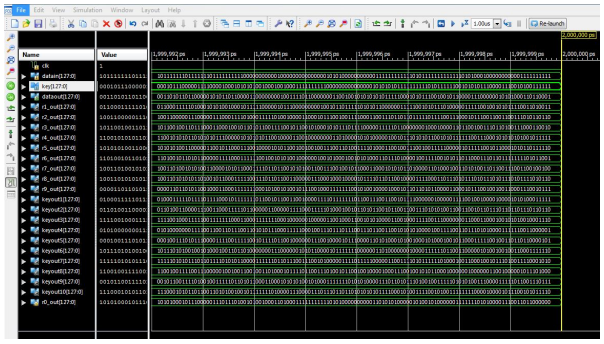
As mentioned in the earlier sections, the only difference between the raw and the adjusted implementation is that the adjusted implementation requires at least three shares for all the blocks including the linear operations in the S-box. For that reason, the shared square scaler circuit is instantiated with three shares. This S-box also requires 44-bits of randomness per iteration.

Nimble Implementation:

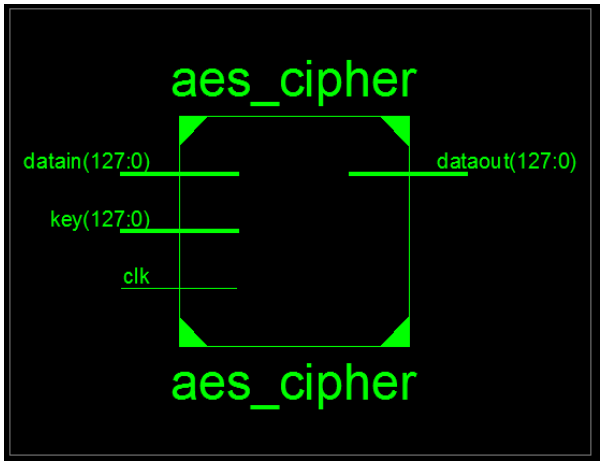
Use fresh randomness at the end of the first phase to satisfy uniformity during the combination of the square scaler's and the multiplier's outputs, and after the inverter to break the dependency between the inputs of the multipliers in the third phase. Since these remasking steps conserve the uniformity property and the security of each block is achieved only by the correctness and non completeness properties 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.

V.RESULTS

Simulation.



RTL Schematic.



Design Summary.

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	14119	4656	303%
Number of Slice Flip Flops	2419	9312	25%
Number of 4 input LUTs	27926	9312	299%
Number of bonded IOBs	385	232	165%
Number of BRAMs	8	20	40%
Number of GCLKs	1	24	4%

Timing Summary.

```
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 6.687ns (Maximum Frequency: 149.538MHz)
   Minimum input arrival time before clock: 53.347ns
   Maximum output required time after clock: 4.040ns
   Maximum combinational path delay: No path found
```

VI.CONCLUSION

Three different versions of TIs of AES. We show that it is possible to achieve first-order DPA resistance with non uniform shared functions if remasking 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, remasking 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 8 k GE small and requires only 32 bits of fresh randomness per S-box calculation, which is a significant improvement over all previous works.

References

[1]. A. Moradi and O. Mischke, "On the simplicity of converting leakages from multivariate to univariate (case study of a glitch-resistant masking scheme)," in *Cryptographic Hardware and Embedded Systems—CHES* (LNCS 8086). Berlin, Germany: Springer, 2013, pp. 1–20.

[2] L. Batina *et al.*, "Mutual information analysis: A comprehensive study," *J. Cryptol.*, vol. 24, no. 2, pp. 269–291, Apr. 2011.

[3] A. Moradi, "Statistical tools flavor side-channel collision attacks," in *Advances in Cryptology—EUROCRYPT* (LNCS 7237). Berlin, Germany: Springer, 2012, pp. 428–445.

[4] B. Bilgin, S. Nikova, V. Nikov, V. Rijmen, and G. Stütz, "Threshold implementations of all 3 × 3 and 4 × 4 S-boxes," in *Cryptographic Hardware and Embedded Systems—CHES* (LNCS 7428). Berlin, Germany: Springer, 2012, pp. 76–91.

[5] A. Poschmann *et al.*, "Side-channel resistant crypto for less than 2300 GE," *J. Cryptol.*, vol. 24, no. 2, pp. 322–345, 2011.

[6] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, "Pushing the limits: A very compact and a threshold implementation of AES," in *Advances in Cryptology—EUROCRYPT* (LNCS 6632). Berlin, Germany: Springer, 2011, pp. 69–88.

[7]. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In CHES 2004, volume 3156 of LNCS, pages 16–29. Springer, 2004.

[8]. D. Canright and L. Batina. A Very Compact "Perfectly Masked" S-Box for AES. In ACNS 2008, volume 5037 of LNCS, pages 446–459. Springer, 2008. the corrected version at Cryptology ePrint Archive, Report 2009/011.

[9]. J. Daemen and V. Rijmen. The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, 2002.

[10]. L. Goubin and A. Martinelli. Protecting AES with Shamir's Secret Sharing Scheme. In CHES 2011, volume 6917 of LNCS, pages 79–94. Springer, 2011.