Securing an Efficient Lightweight AES Accelerator

Huang, R.; Aljuffri, A.A.M.; Hamdioui, S.; Ma, K.; Taouil, M.

DOI
10.1109/TrustCom60117.2023.00121

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)

2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 979-8-3503-8199-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/TrustCom60117.2023.00121

Securing an Efficient Lightweight AES Accelerator

*Ruoyu Huang[†‡], *Abdullah Aljuffri[†], Said Hamdioui[†], Kezheng Ma[‡], Mottaqiallah Taouil[†]
[†] *Department of Computer Engineering, Delft University of Technology*, Delft, The Netherlands,
R.Huang-4@student.tudelft.nl, {A.A.M.Aljuffri, S.Hamdioui, M.Taouil}@tudelft.nl
[‡] *Silicon Integrated*, Eindhoven, The Netherlands, kezheng.ma@si-in.com
[*] *These authors contributed equally to this paper.*

***Abstract*—The Advanced Encryption Standard (AES) is generally regarded as one of the most popular cryptographic algorithms for ensuring data security. Typical lightweight implementations of the algorithm published in the literature focus on area and power optimization, while neglecting the performance. This paper presents a novel lightweight approach for the AES algorithm and considers both encryption and decryption. In terms of performance per unit area and performance per unit power, our 32-bit design outperforms the state-of-the-art by 1.69x and 1.27x, respectively. These improvements become even larger when implementing higher data-path designs, such as 64-bit or 128-bit designs. To enhance its resilience against side-channel attacks, we modified our design by adopting and further improving on the most recent countermeasure, i.e., Domain-Oriented Masking (DOM). The results demonstrate that our five-stage and eight-stage 1^{st}-order DOM SBOX designs achieve a reduction in area of 9.9% and 6.9% compared to the original proposed design, respectively.**

***Index Terms*—Advanced Encryption Standard, Domain Oriented Masking, Lightweight Accelerator, Internet of Things, Side Channel Attacks**

I. Introduction

According to a study that was recently released in the World Economic Forum [1], malicious attacks were launched against 1.5 billion Internet of Things devices during the first half of 2021. The number of instances of data breaches increased by 15.1% as compared to the previous year. As a consequence of this, the security of the Internet of Things (IoT) has become of the utmost importance and can no longer be treated as an afterthought. This is particularly true when considering the anticipated yearly increase in adoption of those devices, which is expected to reach 43 billion in the year 2023 [2]. With respect to data protection, the Advanced Encryption Standard (AES) [3] is one of the algorithms that is most generally recognized and utilized today. It was first presented to the public by the National Institute of Standards and Technology (NIST) in the year 2001. AES is currently the primary encryption method for many applications, including cloud computing [4] and health care [5]. Traditional implementations of the AES use pipelining techniques in order to achieve a high throughput [6–8]. However, as these implementations consume a significant amount of memory and power, it is unfeasible to use them in IoT devices that are limited by area and battery capacity [9]. Therefore, implementations of AES should be area and power efficient, while simultaneously minimizing the impact on throughput to the greatest degree possible.

To overcome these challenges, several lightweight AES accelerators have been proposed, which reduce area and power consumption by shortening the data-path from the conventional 128-bit to 8-bit [10–13] i.e., reducing the number of SBOXes from 16 to 1. Several researchers [14–16] further pursued the reduction of area requirements at the expense of additional cycles. In general, 8-bit data-path designs significantly effect the throughput as at least 160 cycles are required per encryption. More recently, 32-bit designs have been proposed. Such designs achieve a better trade-off between performance and energy efficiency [17]. Most of the above articles focused or reported only on the encryption module. Davis and John [18] considered actually both modules and proposed optimizations across them. However, as will be shown in this paper later, the shared modules have not been fully optimized. Additionally, their reported area measurements only consider the encryption module. To fairly evaluate the designs and realize the best power/latency/area trade-off, it is important to provide the overhead of both the encryption and decryption modules.

This paper presents a novel low-area, low-power and low-latency AES hardware accelerator. Our method takes advantage of the fact that the key remains unchanged throughout a communication session, eliminating the need for repeated execution of the key expansion module. Additionally, we integrate an improved version of the Domain-Oriented Masking (DOM) which is one of the most advanced countermeasures against side channel attacks (SCAs). Our DOM-based AES design is more area-efficient in comparison to the original DOM design. In summary, the contributions of this paper are:

- A proposal of an novel low-area, low-power, and low-latency design of the AES algorithm which is suitable for IoT applications.
- A proposal of a side channel resilient version using an improved DOM implementation.
- Evaluation of the energy consumption, area overhead and performance of the proposed design.

This paper is organized as follows. Section II introduces the background on AES, SCAs and DOM. Section III introduces our proposed AES designs and its DOM security extension. A comparison of the implementation results are provided in Section IV. Finally, Section V concludes this paper.

This work is partially funded by the "Resilient Trust" project of the EU's Horizon Europe research and innovation programme under grant agreement No. 101112282.

2324-9013/23/$31.00 ©2023 IEEE
DOI 10.1109/TrustCom60117.2023.00121

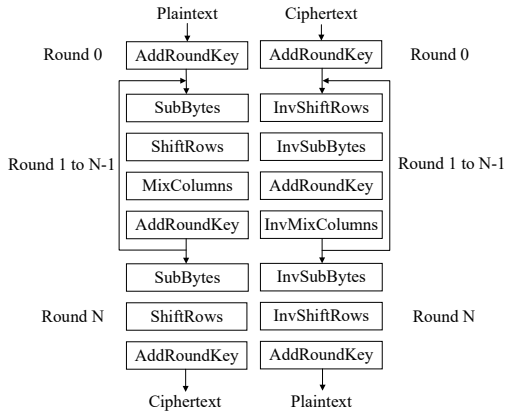


Fig. 1. AES Encryption & Decryption Flow Diagram

II. Background

This section provides a brief background on the AES algorithm, SCAs, and DOM countermeasure.

A. Advanced Encryption Standard (AES)

AES is a symmetric cryptographic algorithm that is used in the cyber world for encrypting and decrypting data to protect them from cyberattacks. It has a fixed data block size of 128 bits, and a key length of 128, 192, or 256 bits. The key length determines the number of rounds required: 10, 12, or 14 rounds for 128-bit, 192-bit, or 256-bit key lengths, respectively. The 128-bit data block is divided into 16 bytes, which are mapped to a 4×4 array referred to as the State array. The diagram of AES encryption and decryption flow is presented in Fig. 1. Each round of encryption includes four primary modules: *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey*, except for Round 0 and the last round (see Fig. 1).

AddRoundKey module involves bit-wise XOR operations of the round key and State array. *SubBytes* module is the only nonlinear module in the AES and plays a crucial role in defending against linear crypt-analysis [19]. When performing *SubBytes* module, each byte in the state array is substituted with another byte using SBOX. The SBOX is generated using a combination of a multiplicative inverse in Galois Field GF(2^8) and an affine transformation [20]. *ShiftRows* module is a transformation that cyclically shifts the second, third, and fourth rows of the State array by one, two, and three bytes to the left, respectively, while leaving the first row unchanged. The *InvShiftRows* module is computed by performing the corresponding rotations to the right. The *MixColumns* and *InvMixColumns* module perform a modular polynomial multiplication in Galois Field GF(2^8) on each column of the State array.

B. Side channel attacks (SCAs)

Side channel attacks take advantage of vulnerabilities by analyzing unintentional physical information that is disclosed during the device's normal execution [21]. Side channel attacks are capable of extracting confidential data, including cryptographic keys, from a system by analyzing the information that is unintentionally leaked. There are various physical information of a system, such as its power consumption, electromagnetic radiation, timing, and acoustic emissions, which can offer insights into its internal functioning [22, 23]. Among them, power consumption is one of the most widely abused, primarily due to its high success rate and straightforward execution. In side channel power attacks, the adversary performs a statistical analysis on power consumption measurements obtained at an intermediate target, such as an SBOX operation. These measurements will then be connected to a leakage model [24], to obtain the secret information.

C. Domain Oriented Masking (DOM)

To protect implementations of AES against SCA attacks, a widely used technique involves randomizing all intermediate results, known as masking schemes. However, in 2005, Stefan et al. [25] discovered that circuits of masked gates are vulnerable to classical DPA attacks due to glitches that occur within the circuits. To overcome this issue, the threshold implementation (TI) was proposed by Nikova et al. [26] to counter SCA attacks and glitches. However, implementing TI requires a high number of shares, which can be costly. Besides, redesigning the non-linear components of the circuit is needed to meet the requirements for higher-order TI, leading to a significant increase in design effort. To provide a lower cost of design, the DOM technique was proposed [19]. Compared with TI, the number of required shares in DOM is reduced without sacrificing the security level. DOM implementation in hardware has demonstrated its resilience to SCAs up to at least 15^{th} order attacks [19].

In DOM, each variable is represented by d + 1 shares to protect the circuit from d^{th}-order SCAs. The fundamental principle of the DOM approach is to maintain the independence of shares within each domain. This is simple for the linear modules of AES, i.e., *MixColumns*, *AddRoundKey*, and *ShiftRows*, as there are no cross-domain calculation required among the different shares. However, the *SubBytes* operation, which is the only non-linear module in AES requires a substantial amount of multiplications. When performing these multiplications, shares may cross domain borders, making it necessary to use fresh random numbers to maintain their statistical independence.

Hannes and his team [19] proposed two multipliers named DOM-indep and DOM-dep. First, the DOM-indep multiplier requires that the inputs from different domains are uniformly random and independent from each other. Take the 1^{st}-order DOM-indep multiplier as an example which consists of two share domains. Assume that the two independent inputs of multiplier are x and y, where x = $A_x \oplus B_x$, y = $A_y \oplus B_y$ ("$\oplus$" represents the xor operation). Furthermore, Z_0 denotes the fresh random number. The output q can be expressed as (1).

$$\begin{aligned} q &= x * y \\ &= (A_x \oplus B_x)(A_y \oplus B_y) \\ &= A_x A_y \oplus A_x B_y \oplus B_x A_y \oplus B_x B_y \\ &= A_x A_y \oplus (A_x B_y \oplus Z_0) \oplus (B_x A_y \oplus Z_0) \oplus B_x B_y \end{aligned} \tag{1}$$

Hannes et al. [19] categorized the product terms into two parts: inner-domain terms (A_xB_x and A_yB_y) and cross-domain (A_xB_y and A_yB_x). The inner-domain product terms do not reveal critical information since they originate from the same domain. In contrast, cross-domain calculations can only be performed on independent inputs to prevent leakage of information for either x or y. A fresh random Z_0 value is used to remain the statistical independence of the DOM-indep multiplier outputs from other values. Additionally, flip-flops are employed to prevent glitches from propagating through this block. Secondly, Hannes et al. [19] proposed the DOM-dep multiplier based on the structure of the DOM-indep multiplier, which does not require the independence of inputs. However, it requires more fresh random values and additional area. The 1^{st}-order DOM-dep and DOM-indep multipliers can be easily extended to higher order by using more shares and fresh random values.

III. Lightweight AES and DOM Extension

This section presents our proposed implementation approach for AES and its protected version using DOM. We start by motivating our approach, followed by a detailed explanation of its design and implementation.

A. Motivation

Previous research primarily focused on implementing AES on either an 8-bit [10–16] or 32-bit [17, 18] data-path to reduce area and energy consumption. However, these studies only report the area of the encryption module and neglect the decryption part. In reality, the eleven 128-bit registers required for the key expansion in the decryption module contribute significantly to the overall core area. In the decryption module, all round keys must be computed first before the decryption can start. Shortening the data-path from 128-bit to a lower-bit width has a much lower improvement on the area when the decryption module is not ignored. Therefore, in our design we focus on different data-paths in the presence of the decryption unit and compare their performance in terms of throughput, area, and power. Secondly, in actual applications, keys do not change frequently. Hence, we perform the key expansion once and store the results in the registers. As long as the key remains the same, we can skip the key expansion step, resulting in a significant power and latency reduction. In addition, we reorder the sequences of *Addroundkey* and *MixColumns* in the round function which results in further area and performance improvements.

B. Design and Implementation of Proposed Lightweight AES

Our proposed AES designs verify whether the key changes at the start of every encryption/decryption execution. In case a key change is detected, we perform the key expansion module and leave the keys inside the key registers. Otherwise, we directly execute the round modules (i.e., *AddRoundKey*, *SubBytes*, *MixColumns*, and *ShiftRows*). This reduces the execution time of the decryption part by eleven cycles. To further optimize the design area, resource sharing is employed.

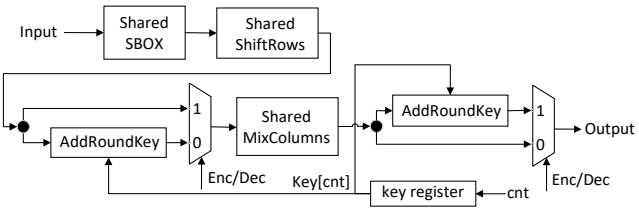


Fig. 2. Proposed Round Function for AES Encryption and Decryption

Initially, we limit the number of registers to store the state to a single 128-bit register that is shared in all the round modules of both encryption and decryption. Next, we combine the encryption and decryption modules to decrease the overall area. The proposed scheme is depicted in Fig. 2, where *cnt* represents the round index and *key[cnt]* denotes the key that needs to be XORed with the State array. The boxes containing the word "shared" represent blocks that are shared between the encryption and decryption. An in-depth explanations of the shared modules will be provided next, including *Shared SBOX, Shared ShiftRows*, and *Shared MixColumns*.

1) ***Shared SBOX****:* Akashi et al. [27] proposed a new composite field to optimize the structure of the SBOX, resulting in a significant reduction of the area compared to using a Look-up table (LUT). Thereafter, several researchers [20, 28, 29] optimized the SBOX based on the structure provided in [27]. These papers used an SBOX which is shared by both the encryption and decryption modules to reduce area. To the best of our knowledge, the SBOX design described in [29] has the lowest area. Compared with previous designs, they shared resources in three modules: preprocess, postprocess, and scalar square. The preprocess module performs the isomorphic mapping and inverse affine transformation for the decryption and the isomorphic mapping only for the encryption. The postprocess module executes the affine transformation and the inverse isomorphic mapping for the encryption and inverse isomorphic mapping only for the decryption. The scalar square performs a square and multiplication with constant $\lambda = \{1, 1, 0, 0\}$, which leads to three XOR reductions [20]. Our new SBOX is based on the SBOX proposed in [29]; it is shown in Fig. 3. It contains an optimized multiplier and a modified inverter. In addition, it combines the operations of the last two multipliers proposed in [27]. Each optimization is described next into more details.

- $GF(2^4)$ **Optimized multiplier:** Our optimized $GF(2^4)$ multiplier is based on the work in [29]. That multiplier consists of 18 XOR and 12 AND gates and its critical path consists of 4 XOR and 1 AND gate. We simplified the $GF(2^4)$ multiplier based on Equation (2), where $\{a_3, a_2, a_1, a_0\}$ and $\{b_3, b_2, b_1, b_0\}$ denote the two 4-bit inputs (see also left bottom of Fig. 3), $\{c_3, c_2, c_1, c_0\}$ denote the 4-bit output, and $\{m_4, m_3, m_2, m_1, m_0\}$ are intermediate variables defined as: $m_4 = m_0 \oplus m_1$, $m_3 = a_0 \oplus a_1$, $m_2 = a_3 \oplus a_2$, $m_1 = a_2 \oplus a_0$, and $m_0 = a_3 \oplus a_1$. Although our $GF(2^4)$ optimized multiplier utilizes 4 more AND gates compared to [29],

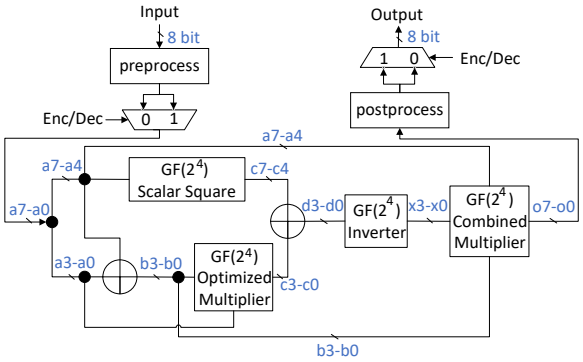


Fig. 3. Proposed Shared SBOX

it requires 1 XOR gate less and more importantly has a shorter critical path (1 AND gate and 3 XOR gates). Surprisingly, after synthesis it turns out that the area of this implementation is also better after synthesis. We believe that compiler is able to extract more common resources with this implementation.

$$
\begin{aligned}
c_3 &= [(a_3 \oplus a_1)\&(b_3 \oplus b_1) \oplus (a_3 \oplus a_1)\&(b_2 \oplus b_0) \oplus (a_2 \oplus \\
&\quad a_0)\&(b_3 \oplus b_1)] \oplus [(a_1\&b_1) \oplus (a_1\&b_0) \oplus (a_0\&b_1)] \\
&= (b_0\&a_3) \oplus (b_1\&(a_2 \oplus a_3)) \oplus (b_2\&(a_1 \oplus a_3)) \oplus \\
&\quad (b_3\&(a_0 \oplus a_1 \oplus a_2 \oplus a_3)) \\
&= (b_0\&a_3) \oplus (b_1\&m_2) \oplus (b_2\&m_0) \oplus (b_3\&m_4); \\
c_2 &= [(a_3 \oplus a_1)\&(b_3 \oplus b_1) \oplus (a_2 \oplus a_0)\&(b_2 \oplus b_0)] \oplus (a_1\&b_1) \\
&\quad \oplus (a_0\&b_0) \\
&= (b_0\&a_2) \oplus (b_1\&a_3) \oplus (b_2\&(a_0 \oplus a_2)) \oplus (b_3\&(a_1 \oplus a_3)) \\
&= (b_0\&a_2) \oplus (b_1\&a_3) \oplus (b_2\&m_1) \oplus (b_3\&m_0); \\
c_1 &= [(a_3\&b_3) \oplus (a_3\&b_2) \oplus (a_2\&b_3)] \oplus [(a_3\&b_3) \oplus (a_2\&b_2)] \\
&\quad \oplus [(a_1\&b_1) \oplus (a_1\&b_0) \oplus (a_0\&b_1)] \\
&= (b_0\&a_1) \oplus (b_1\&(a_0 \oplus a_1)) \oplus (b_2\&(a_2 \oplus a_3)) \oplus (b_3\&a_2) \\
&= (b_0\&a_1) \oplus (b_1\&m_3) \oplus (b_2\&m_2) \oplus (b_3\&a_2); \\
c_0 &= [(a_3\&b_3) \oplus (a_3\&b_2) \oplus (a_2\&b_3)] \oplus [(a_1\&b_1) \oplus (a_0\&b_0)] \\
&= (b_0\&a_0) \oplus (b_1\&a_1) \oplus (b_2\&a_3) \oplus (b_3\&(a_2 \oplus a_3)) \\
&= (b_0\&a_0) \oplus (b_1\&a_1) \oplus (b_2\&a_3) \oplus (b_3\&m_2).
\end{aligned}
\tag{2}
$$

- $GF(2^4)$ **Inverter:** To decrease the area of the inverter and make the design easier to secure (by performing less non-linear operations), we further optimized the inverter based on the structure that proposed in [20]. The improved design is shown in Fig. 4. The $GF(2^2)$ *Scalar Square* performs a $GF(2^2)$ square operation and a scalar multiplication with the constant $\varphi = \{1, 0\}$. Equation (3) illustrates the *Scalar Square* calculation of [20] (left) and our combined design (right), where d_3, d_2 are the inputs of *Square* module, e_3, e_2 denote intermediate results between *Square* and *Scalar* module, and f_3, f_2 represent the outputs of *Scalar* module (see Fig. 4). As can be seen, our design requires 2 XOR operations less.

$$
[20]\begin{cases} e_3 = d_3 \\ e_2 = d_3 \oplus d_2 \\ f_3 = e_3 \oplus e_2 \\ f_2 = e_3 \end{cases} \qquad Combined\begin{cases} f_3 = d_2 \\ f_2 = d_3 \end{cases} \tag{3}
$$

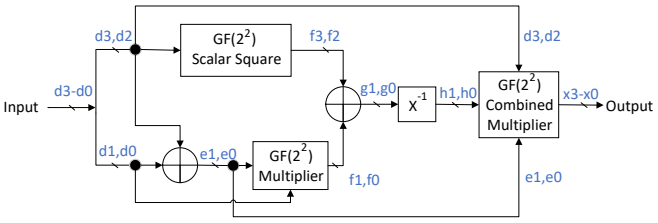


Fig. 4. Proposed $GF(2^4)$ Inverter

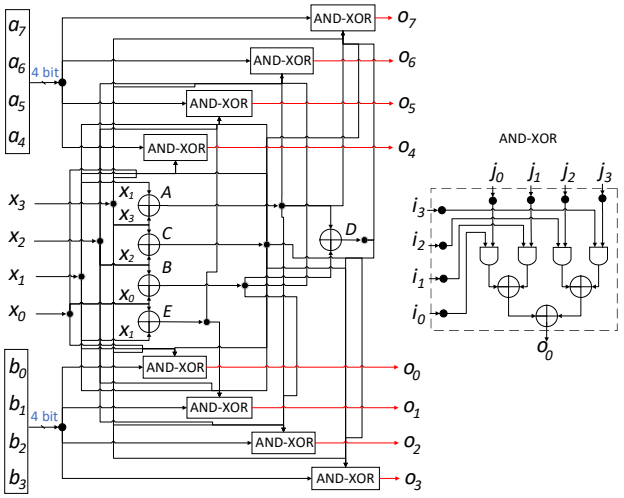


Fig. 5. $GF(2^4)$ Combined Multiplier

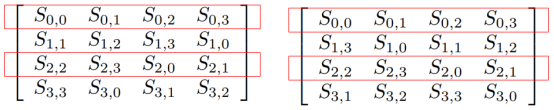
The last step of the inverter consist of two $GF(2^2)$ multipliers. Equation (4) shows the design proposed in [20]. Our optimizations are provided after the second "=" sign by factoring out $X = h_1 \oplus h_0$ commonly between both multiplications. Our combined $GF(2^2)$ multiplier achieves a reduction of 1 XOR gate and 2 AND gates, compared with the design in [20].

$$
\begin{aligned}
x_3 &= (d_3\&h_1) \oplus (d_3\&h_0) \oplus (d_2\&h_1) = (d_3\&X) \oplus (d_2\&h_1); \\
x_2 &= (d_3\&h_1) \oplus (d_2\&h_0); \\
x_1 &= (e_1\&h_1) \oplus (e_1\&h_0) \oplus (f_0\&h_1) = (e_1\&X) \oplus (f_0\&h_1); \\
x_0 &= (e_1\&h_1) \oplus (e_0\&h_0).
\end{aligned}
\tag{4}
$$

- $GF(2^4)$ **Combined Multiplier:** The last two multipliers used in the SBOX presented in [29] were treated as two separate multipliers. However, Ahmad [30] proposed that these two multipliers can be merged together, resulting a significant reduction in area as can be seen at the most right part in Fig. 3. However, it is not clear from the paper how this shared multiplier works. For clarity, we combined the multipliers ourselves and provided a detailed structure of it in Fig. 5.

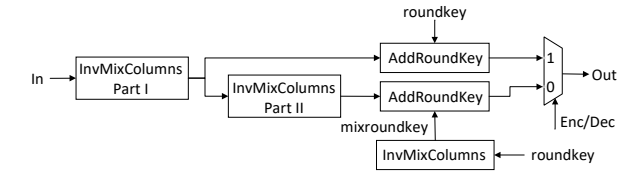
*2) **Shared ShiftRows**:* Davis and John observed that the first and third shift operations in *ShiftRows* and *InvShiftRows* can be shared [18], as both produce the same results for the decryption and encryption. This can be seen in Fig. 6. However, the other two rows (i.e., row two and four) have different behavior and multiplexers are needed to select between them.

*3) **Shared MixColumns**:* We optimize the *Shared MixColumns* based on the proposed design in paper [20], which

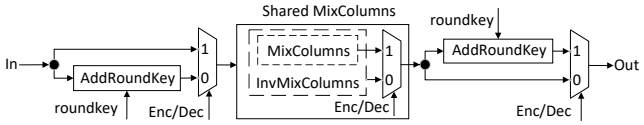


(a) ShiftRows (b) InvShiftRows

Fig. 6. Shift Transformation of ShiftRows and InvShiftRows



(a) MixColumns/InvMixColums [20]



(b) Proposed Shared MixColumns

Fig. 7. Diagram of MixColumns [20] and Proposed Shared Mixcolumns

shares resources between *MixColumns* and *InvMixColumns*. The design in [20] however requires an additional *InvMixColumns* calculation to rectify the roundkey (see Fig. 7(a)). In contrast, Fig. 7(b) shows that our proposed *Shared MixColumns* combines *MixColumns* and *InvMixColumns*, and reorganizes the sequence of *Addroundkey* and *Shared MixColumns* to avoid performing this additional *InvMixColumns* calculation. This lead to further area improvements.

C. Design and Implementation of Proposed Lightweight DOM

DOM was proposed in [19] to protect AES implementations against SCAs. The authors introduced two types of SBOXes: a five-stage SBOX and an eight-stage SBOX. The five-stage SBOX represents an optimized version of the eight-stage SBOX, resulting in a savings of three cycles per round. Hence, overall it is 33 cycles (3 cycles $\times$ 11 rounds) faster, which is a significant performance improvement. This improvement comes with only a minor increase in the overall area, from 2.6k Gates to 2.8k Gates, as documented in [19]. Therefore, we chose to start from the five-stage SBOX and integrate it into our optimized design. Note that a 1^{st}-order DOM can be easily scaled into a higher order DOM without redesigning components [19]. Therefore, without loss of generality, we focus

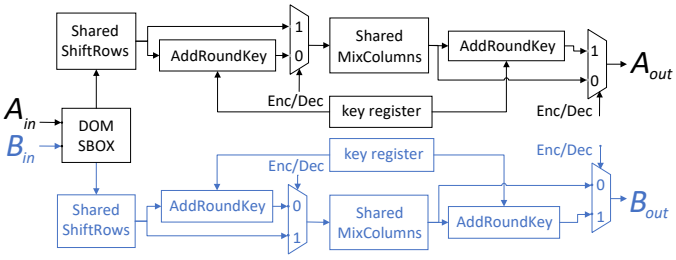


Fig. 8. Proposed Design for DOM Encryption and Decryption

on the optimization of the 1^{st}-order DOM. As our proposed low-area design considers both encryption and decryption with shared resources (see Fig. 2), the lightweight DOM AES was implemented in the same manner, diverging from the original design that concentrated only on encryption [19]. Fig. 8 shows the main part of our lightweight DOM design, where A_{in}, B_{in} represent the input shares, and A_{out}, B_{out} the output shares. As discussed in the background, the independence of d+1 shares within linear modules can be ensured by employing d+1 identical modules. However, the non-linear SBOX module requires to be carefully designed. Our design is based on the lightweight DOM SBOX proposed in [19]. In comparison to that design, our approach shares the resources between encryption and decryption parts using the preprocess and postprocess functions. In addition, we optimize the DOM-indep and DOM-dep multipliers based on our simplified and shared multipliers to further reduce the area.

Fig. 9 depicts our design of the 1^{st}-order five stages lightweight DOM SBOX, where A_{sin} and B_{sin} denote the input shares, and A_{sout} and B_{sout} denote the output shares. In the figure, $Z_0, Z_1, ..., Z_6$ and $A_{z0}, B_{z0}, ..., A_{z3}, B_{z3}$ are fresh random values of the simplified DOM-indep and DOM-dep multipliers, respectively. The flip-flops with dotted boxes are optional registers that are only necessary in pipelining scenarios. For example, when the data-path is less than 128 bits, the SBOX needs to be reused multiple times within one round, causing the input to change before the round is completed. In this case, the dotted flip-flops are necessary to ensure the design's functional correctness. Fig. 9 also highlights the parts that we improved in red, i.e., the DOM multipliers. Compared to the original DOM multipliers (see [19]), we replaced their multipliers with our simplified multipliers (see (2)) and shared multipliers (see Fig. 5), resulting in a reduction in power and area. In addition, multiplexers are used to select between inputs for the encryption and decryption units.

Fig. 10 illustrates our changes made to the 1^{st}-order Dom-dep multiplier [19]; we refer to it as simplified DOM-dep multiplier. In the figure, A_a, B_a, A_b, B_b are the inputs, while A_q and B_q correspond to the outputs. A_z, B_z, and Z_0 are random numbers that are used to ensure the independence of shares. In contrast to the design proposed in [19], our proposed simplified DOM-dep multiplier utilizes a simplified version of the DOM-indep multiplier and merges the right two multipliers, resulting in significant area reduction. The simplified DOM-indep multiplier is based on the simplified multiplier shown in Equation (2). Equation (5) illustrates the expression of our DOM-dep multiplier, where M=$(A_b \oplus B_b) \oplus (A_z \oplus B_z)$. We denote that a= $A_a \oplus B_a$, b= $A_b \oplus B_b$, z= $A_z \oplus B_z$, and q= $A_q \oplus B_q$. In the equation, A_{qi} and B_{qi} are the outputs of simplified DOM-indep multipliers. The combined multiplier (see Fig. 5) is utilized for the calculation of $(A_a * M \oplus B_a * M)$ to further reduce area. The shared DOM-indep/DOM-dep multiplier modules in Fig. 9 are implemented with the simpilified DOM-indep/DOM-dep multipliers (see Fig. 10(b)). We share the common resources between these two multipliers to further reduce the area.

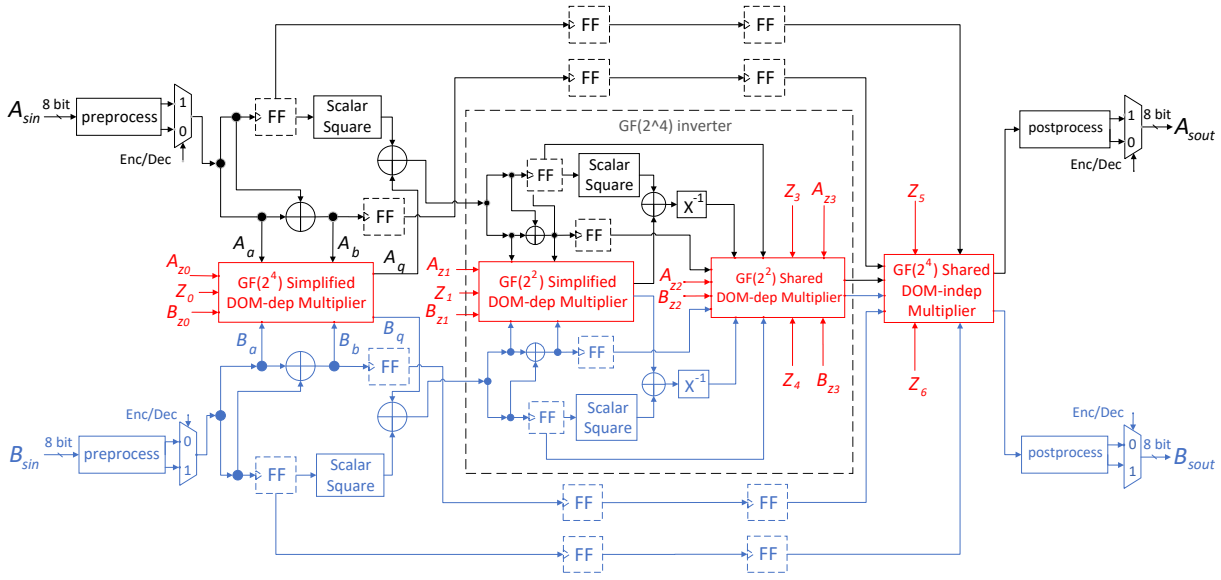
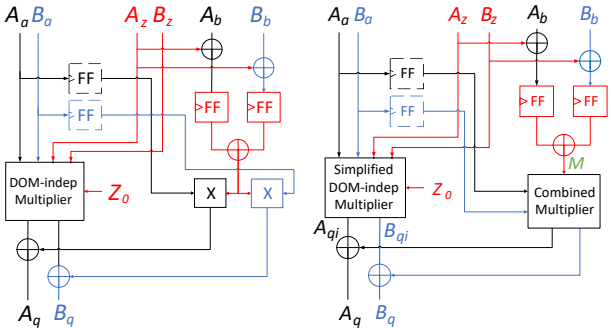


Fig. 9. Structure of the 1^{st}-order five-stage DOM SBOX Module (modified based on [19])



(a) DOM-dep [19] (b) Simplified DOM-dep

Fig. 10. DOM-dep Multiplier [19] and Our Simplified DOM-dep Multiplier

$$
\begin{aligned}
& a * b \\
&= a * (b \oplus z) \oplus a * z \\
&= (A_a \oplus B_a)(A_b \oplus B_b \oplus A_z \oplus B_z) \oplus (A_a \oplus B_a)(A_z \oplus B_z) \\
&= (A_a * M \oplus B_a * M) \oplus (A_{qi} \oplus B_{qi}) \\
&= (A_a * M \oplus A_{qi}) \oplus (B_a * M \oplus B_{qi}) \\
&= A_q \oplus B_q = q
\end{aligned} \tag{5}
$$

IV. Experimental Results

This section presents the experiment setup and the results.

A. Setup

For the majority of IoT applications, the maximum payload size for each packet is between 1600 bytes (e.g., NarrowBand-IoT [31]) and 256 megabytes (e.g., Message Queue Telemetry Transport(MQTT) [32]). Taking the lower limit into consideration, we assume that the key will stay the same during the communication session of at least one hundred encrypting/decrypting operations. For that reason, we assumed a fixed key for 100 encryption and decryption operations.

TABLE I
AES Performance Analysis at 50MHz

Design	Data-path	Freq. (MHz)	Area (μm^2)	Area Ratio	Cycle	Cycle Ratio
[18]	32	50	174156	1	11100	1
Proposed AES	32	50	139415	0.8	8211	0.74
	64	50	151677	0.87	4211	0.38
	128	50	178674	1.03	2211	0.2

To compare with the start-of-the-art, we reimplemented the state-of-the-art AES design proposed by Davis and Jones [18] and compared it with ours. All designs are synthesized using TSMC CMOS 180 nm technology. The total area and power consumption of each design are evaluated Using Synopsys Design and Power Compiler. We took the same approach with respect to the DOM design originally proposed in [19].

B. AES Performance Evaluation

In this section we compare our AES SBOX design proposed in Section III.B with the SBOX proposed in [29]. Note that this paper limited itself to only an SBOX implementation. The synthesis results show that the area of our proposed SBOX design is 8.2% lower than the design in [29]. The actual area numbers are 2558 vs 2788 μm^2, respectively.

We compare our complete non-DOM AES design proposed in Section III.B with the design proposed in [18].Tables I and II show the results for both designs synthesized at 50 MHz and maximum frequency, respectively. . From the first table, we can see that our 32-bit data-path implementation needs 20% less area than the design proposed in in [18]. Actually, our 128-bit data-path design is comparable in size to their 32-bit data-path design, while being 5x faster. When we look at Table II we see similar trends. Comparing both 32-bit data-path designs, our design has 14% less area, needs 26% less cycles and can run at 18% higher frequency.

TABLE II
AES PERFORMANCE ANALYSIS AT THE MAXIMUM FREQUENCY

Design	Data-path	Freq. (MHz)	Freq. Ratio	Area (μm^2)	Area Ratio	Cycle	Cycle Ratio
[18]	32	103.1	1	196857	1	11100	1
Proposed AES	32	121.9	1.18	169794	0.86	8211	0.74
	64	117.6	1.14	195355	0.99	4211	0.38
	128	112.4	1.09	236553	1.2	2211	0.2

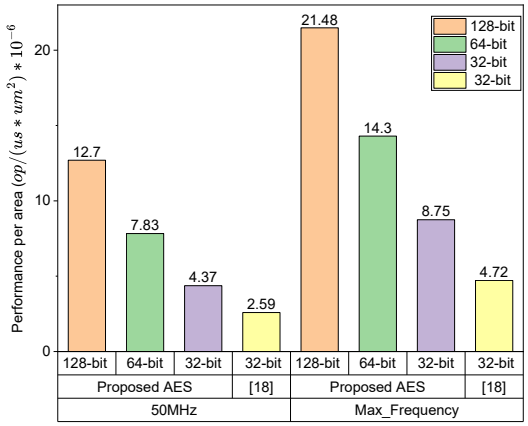


Fig. 11. Performance per Area Comparison vs AES Design in [18]

Figs. 11 and 12 depict the performance per area and performance per power of our proposed AES design and the state-of-the-art design in [18]. The figures show that among our proposed designs, the 128-bit design achieves the highest score in terms of performance per area and performance per power. Our proposed 128-bit, 64-bit, and 32-bit designs surpass the state-of-the-art [18] in terms of performance per area by a factor of 4.90x, 3.02x, and 1.69x, respectively, when operating at 50MHz and by a factor of 4.55x, 3.03x, and 1.85x, respectively, when operating at the maximum frequency. They also outperform the state-of-the-art design in terms of performance per power by a factor of 2.68x, 1.70x, and 1.27x, respectively. Note that in Fig. 12, only the performance per power for the 50 MHz implementation is displayed, as there were minimal differences observed when compared to the designs operating at their maximum frequencies.

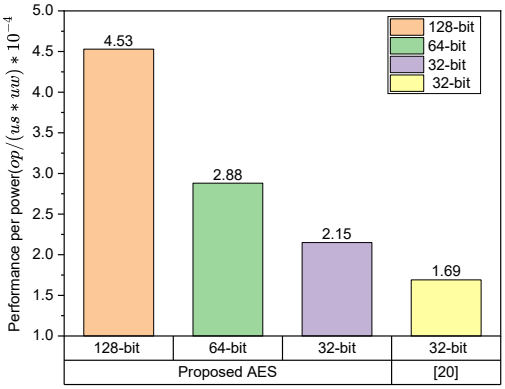


Fig. 12. Performance per Power Comparison vs AES Design in [18]

TABLE III
AREA COMPARISON OF DOM SBOX

Design	SBOX Type	Area (μm^2)	Area Ratio
[19] DOM SBOX	eight-stage	19682	1
	five-stage	21196	1.077
Proposed DOM SBOX	eight-stage	17735	0.901
	five-stage	19741	1.003

TABLE IV
DOM PERFORMANCE ANALYSIS AT 50MHZ

Data-path	Frequency (MHz)	Area (μm^2)	Area Ratio	Cycle	Cycle Ratio
128-bit	50	615172	1	10251	1
64-bit	50	445269	0.72	12251	1.2
32-bit	50	361582	0.59	16251	1.59

C. DOM Performance Evaluation

Table III shows a comparison of the area between our proposed 1^{st}-order DOM SBOX and the original 1^{st}-order SBOX proposed in [19]. Compared to their design, our eight-stage and five-stage 1^{st}-order DOM SBOX designs achieve an area reduction of 9.9% and 6.9%, respectively.

Although the 128-bit data-path design has the best performance, we have implemented also 64-bit and 32-bit data-path versions. Unfortunately, the authors in [19] focused only on the SBOX and have not evaluated the complete AES design. Nevertheless, to comprehensively assess the influence of these designs on overall performance, our proposed DOM SBOX has been incorporated into all AES configurations, encompassing the 128-bit, 64-bit, and 32-bit versions. Tables IV and V present their area and latency results for an operating frequency of 50 MHz and their maximum operating frequency, respectively. As we mentioned in the Section III-C, we chose the five-stage SBOX in our designs because it offers a substantial performance improvement with only minor sacrifices in terms of area when compared to the eight-stage SBOX. Consequently, the key expansion process takes 51 (5*10+1) cycles, while the encryption and decryption operations in the 128-bit, 64-bit, and 32-bit data-path designs require 51 (5*10+1), 61 (6*10+1), and 81 (8*10+1) cycles, respectively. As a result, it takes 10251 (51*2*100+51), 12251 (61*2*100+51), and 16251 (81*2*100+51) cycles for these designs to perform 100 encryption/decryption operations. Tables IV and V demonstrate that the 32-bit design exhibits a better area, whereas the 128-bit design has a higher performance.

The DOM SBOX contains a great number of registers and operation, resulting in a high power consumption and area. However, lower data-path designs can significantly reduce area and power consumption by utilizing fewer DOM SBOXes. Fig. 13 shows the performance per area comparison of our

TABLE V
DOM PERFORMANCE ANALYSIS AT THE MAXIMUM FREQUENCY

Data-path	Frequency (MHz)	Frequency ratio	Area (μm^2)	Area Ratio	Cycle	Cycle Ratio
128-bit	188.7	1.79	698780	1	10251	1
64-bit	190.8	1.81	522844	0.75	12251	1.2
32-bit	192.3	1.83	446528	0.64	16251	1.59

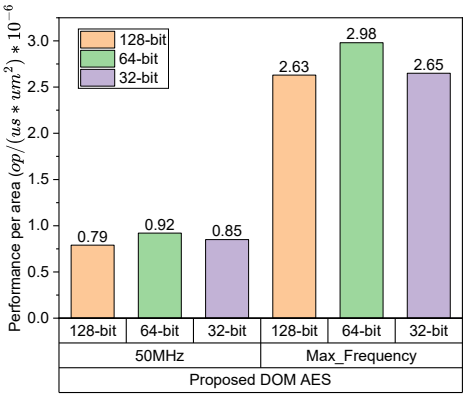


Fig. 13. Performance per Area Analysis of our 1^{st}-Order DOM AES designs

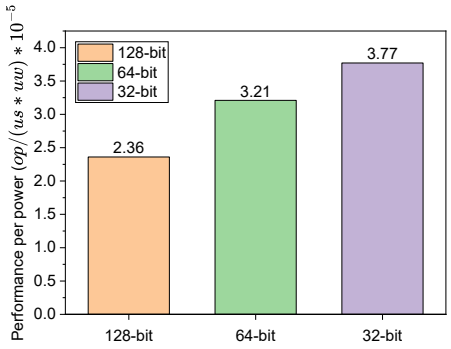


Fig. 14. Performance per Power Analysis of our 1^{st}-Order DOM AES Designs

proposed DOM designs. According to the figure, the 64-bit design performs better at both 50 MHz and the maximum frequency. Fig. 14 illustrates a comparison of the performance per power for our proposed DOM designs, where the 32-bit design outperforms the others.

V. Conclusion

In this paper we presented a novel low-power and low-area AES accelerator with high performance. A number of different optimization techniques, such as key expansion bypassing, resource sharing, and careful modules optimizations have been proposed and implemented. According to the results, our designs outperform the current state-of-the-art in terms of area, power, and performance. Subsequently, we developed an efficient version of the DOM side channel countermeasure using the same optimization techniques. The results demonstrated that also our DOM SBOX achieves a lower area than the originally proposed design.

References

[1] W. E. Forum, "Connected devices need better governance: Here's how to achieve it," https://www.weforum.org/agenda/2023/01/connected-devices-need-better-governance/, 2023.

[2] Mackinsey *et al.*, "Connected devices need better governance: Here's how to achieve it," https://www.mckinsey.com/industries/private-equity-and-principal-investors/our-insights/growing-opportunities-in-the-internet-of-things, 2023, accessed: 2023-04-15.

[3] M. Dworkin *et al.*, "Advanced encryption standard(aes)," *Federal Inf. Process. Stds. (NIST FIPS)*, 2001.

[4] P. Sivakumar *et al.*, "Securing data and reducing the time traffic using aes encryption with dual cloud," in *IEEE ICSCAN*, 2019, pp. 1–5.

[5] M. Khader *et al.*, "Simplified aes algorithm for healthcare applications on internet of thing," in *8th ICIT*, 2017.

[6] S. Mathew *et al.*, "53 gbps native $gf(2^4)^2$ composite-field aes-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors," *IEEE J. Solid State Circuits*, vol. 46, pp. 767–776, 2011.

[7] R. V. Kshirsagar *et al.*, "FPGA implementation of high speed VLSI architectures for AES algorithm," in *Fifth ICEST, Himeji, Japan*, November 5-7, 2012.

[8] S. Chellappa *et al.*, "Advanced encryption system with dynamic pipeline reconfiguration for minimum energy operation," in *Sixteenth ISQED Santa Clara, CA, USA*, March 2-4, 2015.

[9] E. Kwarteng *et al.*, "A survey on security issues in modern implantable devices: Solutions and future issues," *CoRR*, 2022.

[10] M. Lu *et al.*, "A compact, lightweight and low-cost 8-bit datapath AES circuit for iot applications in 28nm CMOS," in *TrustCom/BigDataSE 2018, New York, USA*, August 1-3, 2018.

[11] S. N. Dhanuskodi *et al.*, "Efficient register renaming architectures for 8-bit AES datapath at 0.55 pj/bit in 16-nm finfet," *IEEE Trans. Very Large Scale Integr. Syst.*, 2020.

[12] M. S. Wamser *et al.*, "Pushing the limits further: Sub-atomic AES," in *VLSI-SoC, Abu Dhabi, United Arab Emirates*, October 23-25, 2017.

[13] S. Banik *et al.*, "Atomic-aes: A compact implementation of the AES encryption/decryption core," in *Progress in Cryptology -17th INDOCRYPT in Kolkata, India*, December 11-14, 2016.

[14] A. Moradi *et al.*, "Pushing the limits: A very compact and a threshold implementation of AES," in *-30th Eurocrypt, Tallinn, Estonia*, 2011.

[15] S. Mathew *et al.*, "340 mv–1.1 v, 289 gbps/w, 2090-gate nanoaes hardware accelerator with area-optimized encrypt/decrypt $(gf(2^4))^2$ polynomials in 22 nm tri-gate cmos," in *IEEE Journal of Solid-State Circuits*, vol. 50, no. 4, 2015.

[16] J. Yu *et al.*, "Benchmarking and optimizing aes for lightweight cryptography on asics,," in *Proceedings of the Lightweight Cryptography Workshop, Gaithersburg, MD, USA.*, 4–6 November, 2019.

[17] M.-H. Dao *et al.*, "An energy efficient aes encryption core for hardware security implementation in iot systems," in *2018 International Conference on ATC*, 2018, pp. 301–304.

[18] C. Davis *et al.*, "Shared round core architecture: A novel AES implementation for implantable cardiac devices," in *65th IEEE MWSCAS, Fukuoka, Japan*, August 7-10, 2022.

[19] H. Groß *et al.*, "Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order," *IACR Cryptol. ePrint Arch.*, p. 486, 2016.

[20] X. Zhang *et al.*, "High-speed VLSI architectures for the AES algorithm," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 12, pp. 957–967, 2004.

[21] Y. Zhou *et al.*, "Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing," *IACR Cryptol. ePrint Arch.*, p. 388, 2005.

[22] G. Joy Persial *et al.*, "Side channel attack-survey," *Int. J. Adv. Sci. Res. Rev*, vol. 1, pp. 54–57, 2011.

[23] T.-H. Le *et al.*, "How can signal processing benefit side channel attacks ?" in *2007 IEEE Workshop on SAFE*, 2007, pp. 1–7.

[24] E. Brier *et al.*, "Correlation power analysis with a leakage model," in *CHES: MA, USA.*, August 11-13, 2004.

[25] S. Mangard *et al.*, "Side-channel leakage of masked CMOS gates," in *Topics in Cryptology - CT-RSA*, vol. 3376. Springer, 2005, pp. 351–365.

[26] S. Nikova *et al.*, "Threshold implementations against side-channel attacks and glitches," in *ICICS, Raleigh, NC, USA.*, December 2006.

[27] A. Satoh *et al.*, "A compact rijndael hardware architecture with s-box optimization," in *Advances in Cryptology - ASIACRYPT 2001, Gold Coast, Australia.*, C. Boyd, Ed., December 9-13, 2001.

[28] N. Ahmad *et al.*, "Low-power compact composite field AES s-box/inv s-box design in 65 nm CMOS using novel XOR gate," *Integr.*, vol. 46, pp. 333–344, 2013.

[29] Y. Teng *et al.*, "VLSI architecture of s-box with high area efficiency based on composite field arithmetic," *IEEE Access*, vol. 10, 2022.

[30] N. Ahmad, "New architecture of low area aes s-box/ inv s-box using vlsi implementation," *Jurnal Teknologi*, vol. 78, May 2016.

[31] K. Mekki *et al.*, "A comparative study of LPWAN technologies for large-scale iot deployment," *ICT Express*, vol. 5, pp. 1–7, 2019.

[32] D. Thavamani, "Mqtt messages-an overview," *International Journal of Mathematics and Computer Research*, vol. 09, 04 2021.