FPGA Implementation of AES-Based on Optimized Dynamic s-Box

Calvo Mayaudón Haroldo[1], Nakojah Chris David[1], Mahdi Madani[2][a] and El-Bay Bourennane[2]
[1]*Faculty of Science and Technology, University of Burgundy, Dijon, France*
[2]*ImViA Laboratory (EA 7535), University of Burgundy, 21000 Dijon, France*

Keywords: Dynamic S-Box, FPGA Implementation, AES Algorithm, Hardware Metrics, Timing Performance, Cryptographic Analysis.

Abstract: In this paper, we present a hardware implementation of an enhanced version of the AES (Advanced Encryption Standard) algorithm, and evaluate its performance. In the proposed design, we replaced the original static S-boxes with a robust dynamic S-box generator mechanism. The principle consists of using the secret key to generate new dynamic S-boxes by applying a bitwise XOR operation with all 256 bytes of the AES standardized S-box. Then, the architecture is implemented on a Xilinx XC7Z020 PYNQ-Z2 FPGA platform to accelerate the calculations, and its robustness is evaluated using many security tests. The experimental results prove the satisfaction of our design for several cryptographic properties, such as nonlinearity, bijectivity, and strict avalanche criterion that confirm its resistance against the main cryptanalysis attacks.

1 INTRODUCTION

Communication or information sharing is a key aspect of life that comes in various forms, such as text format, images, and videos. Images are the most commonly used visual form of communication in today's digital age. Digital or wireless communication comes with its own security challenges, with the transmitted information being vulnerable to attacks; hence, its confidentiality and integrity are of great concern. Data encryption, or simply cryptography, is a method of transforming data (plaintext, either texts or images) into a language (ciphertext) that is not recognizable or cannot be read by unauthorized parties without the permission of the sender (key). There are different cryptography methods. In Symmetric cryptography, the same key is used for encryption and decryption, where the sender and receiver must share the same key.

Symmetric cryptography includes two types of ciphers: block ciphers and stream ciphers. Block ciphers convert plaintext into ciphertext in fixed block sizes. AES (Rijmen and Daemen, 2001) and DES are examples of symmetric cryptography that operate on fixed block sizes of 128 and 64 bits, respectively. The role of block cipher encryption and decryption is to provide diffusion and confusion functions. The confusion is enforced by establishing a nonlinear function between the plaintext and the ciphertext; the diffusion consists of spreading the influence of a small change to the plaintext throughout the whole ciphertext. Stream ciphers, on the other hand, convert plaintext into ciphertext one bit or byte at a time. Asymmetric cryptography uses a pair of keys for encryption. A public key for encryption and a private key for decryption. Diffie Hellman, RSA, and RCC are examples. The Advanced Encryption Standard (AES), developed by Rijndael and adopted by the National Institute of Standards and Technology (NIST) since 2001, has been approved as the standard encryption algorithm (Dubertret, 2023).

Recently, the hardware implementation of encryption algorithms has gained a wave of attention from researchers and developers because of their high performance and efficiency compared to software-based solutions. The AES can be implemented efficiently both in hardware and software. Software implementation takes the smallest resources, but it offers only limited physical security. Because of the growing requirements for high-speed, high-volume secure communication combined with physical security, the hardware implementation of cryptographic algorithms becomes essential (Rahimunnisa et al., 2014). FPGAs are desired because of their low energy consumption and parallel processing capabilities, which offer an ideal platform for implementing

[a] https://orcid.org/0000-0001-9727-8567

Haroldo, C., David, N., Madani, M. and Bourennane, E.
FPGA Implementation of AES-Based on Optimized Dynamic s-Box.
DOI: 10.5220/0012780300003767
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT 2024)*, pages 730-737
ISBN: 978-989-758-709-2; ISSN: 2184-7711
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

cryptographic algorithms like AES. The Implementation on FPGA devices can achieve high throughput rates, making them ideal for confusion and diffusion in real-time tasks in digital communication. It is useful in areas where greater emphasis is placed on the speed of communication.

In this study, we improved the design of the AES block cipher and performed its hardware implementation. Then, we assessed its material metric and explored its parallelism properties to accelerate runtime execution. The main principle consists of replacing the static S-box with a dynamic one. The proposed principle consists of the use of the secret key to generate new dynamic S-boxes by applying bitwise XOR operations with all 256 bytes of the AES-standardized S-box. The dynamically generated S-boxes demonstrate at least the same or greater cryptographic characteristics as the standard AES S-box while increasing the complexity needed to break the algorithm. Also, no additional information should need to be exchanged between the sender and receiver of the encrypted data other than the secret key. The choice of the Dynamic S-box encryption mechanism in this research is due to the security improvement that it can provide, and we apply the proposed technique to the AES algorithm because it is a known and widely used encryption algorithm with weaknesses. The dynamic s-box introduces variability, which strengthens the security of the encryption by disrupting the patterns that can be followed by cryptanalysts to break the encryption. Also, the dynamic s-box uses some other cryptographic properties, such as bijectivity, nonlinearity, strict avalanche criteria, and correlation immunity, to ensure the confidentiality of the encrypted data on the FPGA and resist attacks.

The proposed architecture is designed using the Vitis High-Level Synthesis (HLS) with a C++ language and implemented on a Field Programmable Gate Array (FPGA) technology (Smith and Johnson, 2018), (Chen and Wang, 2019) through the Xilinx XC7Z020 PYNQ-Z2 hardware platform. The robustness of the proposed architecture is tested by evaluating its keystream performance by analyzing three different dynamic s-boxes.

The remainder of this paper is organized as follows. Section 2 presents the whole related work, and Section 3 presents briefly the architecture and the processing steps of the standard AES block cipher. Section 4 describes the proposed architecture, including the dynamic S-box designed to enhance the security of the regular architecture. Section 5 presents the Vitis HLS and FPGA implementation results and the hardware metrics (logic resources, efficiency, throughput, frequency, etc.). Section 6 investigates cryptanalytic analysis, allowing us to prove the robustness of the proposed scheme. Finally, Section 7 summarizes the whole article and gives directions for our future work.

2 PREVIOUS RELATED WORKS

In the field of cryptography, there have been several studies on the implementation of the Advanced Encryption Standard (AES) using hardware-based solutions. The challenging factor has been to increase throughput with low latency, considering low-cost devices. An adaptive and convenient way to achieve that is by optimizing the s-box. Hence, various researchers have introduced several methods to achieve this common goal. A few of these works are discussed in this section.

An FPGA-based AES accelerator's comprehensive design and analysis are presented in Wang et al.'s work (Wang and Ni, 2004). The authors provide a cutting-edge method for employing a dynamic S-box to perform the AES algorithm on an FPGA. Unlike classic fixed-function AES accelerators, this method enables the accelerator to adjust to various key sizes and rounds. Memory-mapped I/O (MMIO) interfaces are used to implement the dynamic S-box, enabling the CPU to read and write data to the S-box. To achieve minimal area, various optimization strategies for AES of 32-bit data channels are shown in (Bui et al., 2017). These techniques involve reducing the amount of control logic and registers by decreasing activity and implementing a clock gating scheme for the data storage register. An FPGA with a high-performance implementation of AES-128 is proposed by Li et al. (Li et al., 2017). To get high throughput. The authors combined pipelining and parallelism. In (Zhang et al., 2016), the authors achieved their objective of AES implementation on FPGA by combining techniques such as pipelining, parallelism, and look-ahead logic to optimize the AES algorithm's critical paths. In (Smith and Johnson, 2018), this paper suggests a dynamic S-Box-based FPGA implementation of AES that is effective and suitable for Internet of Things (IoT) devices. The implementation's major goal is to consume resources as little as possible while still achieving excellent speed and security. In (Chen and Wang, 2019), the authors describe a pipeline architecture-based, fast FPGA implementation of AES with Dynamic S-Box. By parallelizing encryption operations and integrating the dynamic S-Box technique for increased security, the approach improves throughput. The secure FPGA implementation of AES with Dynamic S-Box for embedded devices is the main emphasis of this work (Zhang and Li, 2020). The implementation uses FPGA parallelism for effective encryption while addressing security concerns in contexts with limited resources. The AES introduces a composite field arithmetic CFA-based S-box operation to decrease the area (Priya et al., 2017). To lower the crucial delay and boost clock frequency and throughput, the sub-pipelining concept is used in the process. Guzmán, Nieto, and Bernal in (Guzman et al., 2016) used a pipelined architecture on the Xilinx Virtex 5 FPGA platform to offer a hardware implementation of the AES-128 algorithm in non-feedback modes of operation, namely ECB and CTR. In (Mamun et al., 2017), the authors suggest altering the S-box by adding an extra byte, which they refer to as the 'AES-SBOX+.' The purpose of the improved S-box is to make the AES algorithm more resistant to linear and differential cryptanalysis. The authors also used power analysis to assess the security of the AES-SBOX+ against side-channel attacks and show that it is more resilient to these types of assaults than the traditional AES S-box. To produce the S-Box output, the input data is subjected to an affine transformation and bit-permutation as part of the transformation technique described in (Nandan and Gowri Shankar Rao, 2022). The authors show that this transformation method is resistant to various cryptanalytic attacks, such as differential and linear attacks. Compared to the traditional AES S-Box design, the suggested S-Box design uses less power and takes up less space when implemented on an FPGA platform.

3 STANDARD AES BLOCK CIPHER ARCHITECTURE

The Advanced Encryption Standard (AES) is a symmetric encryption algorithm. This means that both the sender and receiver use the same key for both encryption and decryption of the data. It has a fixed block size of 128 bits and variable key lengths of 128, 192, or 256 bits. The key is used to initialize a series of round keys, which are then used in a complex series of operations to transform the input data (plaintext) into encrypted data (ciphertext). The matrix used to store the ciphertext block in all the intermediate and final steps is referred to as a state matrix, 16×16 bytes.

In the AES architecture, the encryption and decryption transformations are carried out in a series of operations. The algorithm consists of 10, 12, or 14 rounds, depending on the size of the key used. These rounds consist of a series of transformations, which are substitution, permutation, and mixing of data. The core operations of AES are SubBytes (byte substitution), MixColumn (mixing column), ShiftRow (row-wise permutation), and AddRoundKey (see Figure 1). Each round provides a different effect on the final ciphertext. The substitution round (SubBytes) is a nonlinear function, that provides confusion and resistance to linear and differential cryptanalysis, The Sbox should be carefully designed so that this operation may be reversible and provide an unequivocal one-to-one relation between the elements of the Sbox and its inverse (bijectivity). The permutation steps ShiftRows and MixColumns are linear functions included in the different rounds to spread the influence of each plaintext byte over the whole state matrix (diffusion). Finally, the key addition round (AddRoundKeys) makes the ciphertext dependent on the secret key.

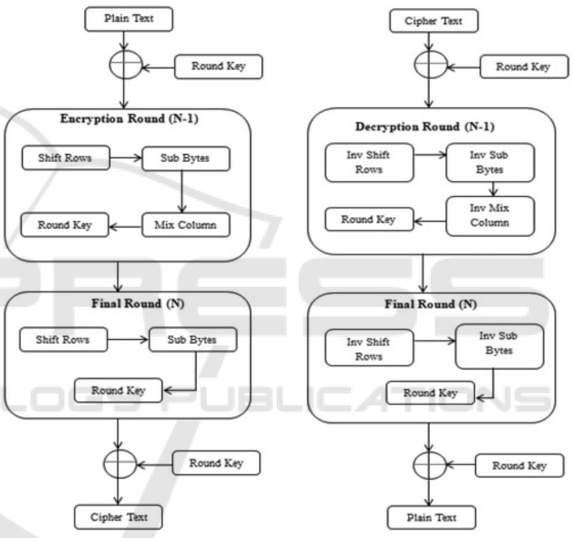


Figure 1: Encryption and decryption architecture of AES.

The cryptographic properties of S-boxes measure how randomly the ciphertext bits are changed, some key figures of merit are shown below (Webster and Tavares, 1986), (Waqas et al., 2015):

- Strict Avalanche Criterion SAC: Whenever a single bit at the input changes, all the output bits should have a 50% probability of changing, it is measured on a scale [0-1], This property is related to diffusion.
- Bit Independence Criterion-BIC: Output bits j and k should change independently when any single input bit i is changed (complemented). Correlation between j and k when i is reversed.
- Nonlinearity-NLM: Complex and non-proportional relationship between the inputs and the outputs. This property is related to confusion [19].

In addition to these requirements, S-boxes should be reversible to guarantee the bijectivity (completeness) of the encryption-decryption process.

4 PROPOSED ARCHITECTURE

The two main objectives set by this work consist of improving the AES algorithm by adding dynamic S-boxes and increasing the runtime performance by implementing the modified algorithm on an FPGA platform.

It is expected that the dynamically generated S-boxes demonstrate at least the same or greater cryptographic characteristics as the standard AES S-box while increasing the complexity needed to break the algorithm. Also, no additional information needs to be exchanged between the sender and receiver of the encrypted data other than the secret key. The FPGA implementation should exploit parallelism within the AES algorithm. The approach introduced in (Arrag et al., 2013) will be employed to generate key-dependent dynamic S-boxes, It consists of the use of one byte of the cipher key to generate new S-boxes by applying the XOR operation with all 256 bytes of the AES S-box. In this work, the 7th byte was used (but any byte could be used), thus the S-box will be referred to as S-BOXkey7. In Equation 1 below, the ith element of the new Dynamic S-box, S-BOXkey7, corresponds to the ith element of the S-BOXAES, XORed with the 7th byte of the cipher key.

$$S-BOXkey7[i] = S-BOXAES[i] \oplus Key7 \qquad (1)$$

To generate the inverse S-box, it is necessary to use the inverse routine according to Equation 2.

$$S-BOXkey7\_INV[i] = S-BOXAES\_INV[i \oplus Key7] \qquad (2)$$

The algorithm implemented performs standard AES encryption/decryption, where the updateKey and SubByte functions have been modified to dynamically calculate the new S-box. In Figure 2, a flow diagram of the implemented AES modified encryption algorithm with dynamic S-box.

In the updateKey and SubByte routines, in Figure 5, the key's 7th byte is XORed with the AES S-box corresponding value to generate the new S-box value. Any other byte may be selected, as mentioned in the S-box implementation section. The rest of the encryption steps are the same. For decryption, the key's 7th byte is XORed with the inverse s-box position being

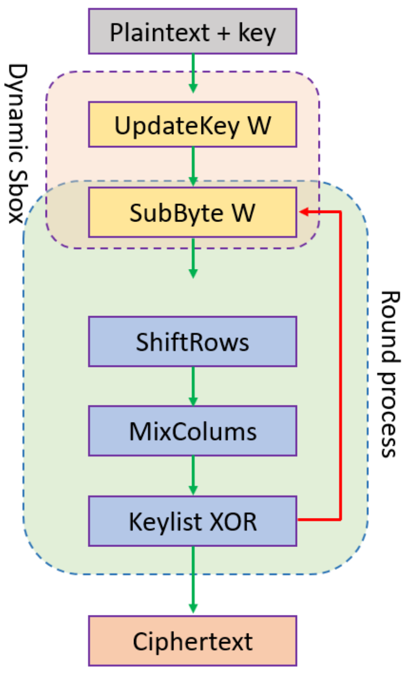
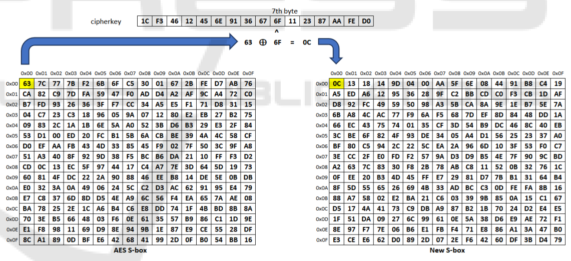


Figure 2: Process of the proposed dynamic s-box.

evaluated and fed to the AES inverse S-box. For better understanding, the approach used is illustrated in Figure 3.


7th byte
cipherkey
1C F3 46 12 45 6E 91 36 67 6F 11 23 87 AA FE D0
63 ⊕ 6F = 0C

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x00	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
0x01	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
0x02	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
0x03	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
0x04	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
0x05	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
0x06	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
0x07	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
0x08	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
0x09	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
0x0A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
0x0B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
0x0C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
0x0D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
0x0E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
0x0F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

AES S-box

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x00	0C	13	18	14	9D	04	00	AA	5F	6E	08	44	91	B8	C4	19
0x01	A5	ED	A6	12	95	36	28	9F	C2	BB	CD	C0	F3	CB	1D	AF
0x02	D8	92	FC	49	59	50	98	A3	5B	CA	8A	9E	1E	B7	5E	7A
0x03	6B	A8	4C	AC	77	F9	6A	F5	68	7D	EF	8D	84	48	DD	1A
0x04	66	EC	43	75	74	01	35	CF	3D	54	B9	DC	46	8C	40	EB
0x05	3C	BE	6F	82	4F	93	DE	34	05	A4	D1	56	25	23	37	A0
0x06	BF	80	C5	94	2C	22	5C	EA	2A	96	6D	10	3F	53	F0	C7
0x07	3E	CC	2F	E0	FD	F2	57	9A	D3	D9	B5	4E	7F	90	9C	BD
0x08	A2	63	7C	83	30	F8	2B	78	AB	C8	11	52	0B	32	76	1C
0x09	0F	EE	20	B3	4D	45	FF	E7	29	81	D7	7B	B1	31	64	B4
0x0A	8F	5D	55	65	26	69	4B	33	AD	BC	C3	0D	FE	FA	8B	16
0x0B	88	A7	58	02	E2	BA	21	C6	03	39	9B	85	0A	15	C1	67
0x0C	D5	17	4A	41	73	C9	DB	A9	87	B2	1B	70	24	D2	E4	E5
0x0D	1F	51	DA	09	27	6C	99	61	0E	5A	38	D6	E9	AE	72	F1
0x0E	8E	97	F7	7E	06	B6	E1	FB	F4	71	E8	86	A1	3A	47	B0
0x0F	E3	CE	E6	62	D0	89	2D	07	2E	F6	42	60	DF	3B	D4	79

New S-box

Figure 3: Example of a dynamic S-box.

According to (Arrag et al., 2013), the resulting S-boxes inherit the cryptographic characteristics of the AES S-box. To corroborate, a non-linearity and average SAC tests (Qu et al., 2009) were performed on 3 S-boxes and compared to the AES S-box value, concluding that the non-linearity and SAC values are equivalent (+/-1%) to those of the AES and satisfy the results of good S-boxes designed for robust block ciphers, as prove the results in Table 1.

Table 1: Dynamic Sbox Non-linearity and SAC.

S-box	NLM	avg_sac
AES	112	0.5048
0x23	112	0.499
0x6F	112	0.4954
0xD7	112	0.4982

Once implemented, the dynamic S-box scheme would enhance the AES algorithm by adding a $2^8 = 256$ complexity to its 2^{128} complexity (Nissar et al., 2019), in the case where the cipher key is changed frequently.

5 VITIS HLS DESIGN WORKFLOW AND FPGA IMPLEMENTATION RESULTS

5.1 Vitis HLS Design

The proposed architecture is designed using the Vitis High-Level Synthesis (HLS) with C++ language. The advantages of this approach are multiple: shorter development time, easier implementation of recursive functions, simple hierarchical design, and the possibility that existing primitive libraries may be re-used to accelerate development time. The design is implemented on the Xilinx XC7Z020 PYNQ-Z2 FPGA platform. The design process for Vitis HLS is as follows:

- C Simulation: The C simulation and the co-simulation allow the designer to compile the code and test the design behavior without programming the target device.
- C Synthesis: On the C Synthesis, the target device resources are allocated based on the design needs (FF, LUT, and SLICEs).
- Co-simulation: The simulation step consists of testing the functional behavior of the implemented module by comparing the generated outputs with the software or theoretical expected outputs.
- Export RTL (Register Transfer Level): The Export RTL stage packs the VHDL or Verilog design into an IP package that may be used in Vivado (VHDL graphical development tool).
- Implementation: The implementation step takes care of the layout and routing of the design.

The implemented design is structured as a C/C++ function, which is denominated as the top-level function. This function may reference other functions within its definition. The input and output arguments to the top-level function establish the interactions with the exported design.

5.2 FPGA Implementation Results

After various simulations and synthesis runs, the design for a CBC mode use of the AES with dynamic S-box was validated at 833 ns. The resource utilization for the design is shown in Table 2.

The maximum frequency was calculated using Equation 3, where T is the target clock period, T=8 ns in the present design, and WNS is the worst negative slack calculated after the implementation place and route in Vivado tools (Mahdi et al., 2023).

$$Max\_Freq = \frac{1}{T - WNS}\ [MHz] \tag{3}$$

To calculate the throughput, Equation 4 is employed, where N is the number of bits per block.

$$Throughput = N \times Max\_Freq\ [Mbps] \tag{4}$$

The Efficiency of the design was evaluated using Equation 5.

$$Efficiency = \frac{Throughput}{Slices}\ [Mbps/Slices] \tag{5}$$

The comparison of hardware metrics of several AES implementations is summarized in Table 3. Despite the comparative study with other FPGA targets with different timing and logic resource characteristics is difficult to carry out, we try to give an objective analysis. Therefore, considering the timing metrics, we remark that the proposed implementation presents one of the better throughput achievements (70 Gb/s) but uses largely low logic resources compared to (Qu et al., 2009), (Liu et al., 2013). In addition, both the timing performances and logic resources are better than those obtained by (Guzman et al., 2016), (Fan and Hwang, 2007). Whereas the (Elsayed et al., 2008) implementation achieved better performances than our architecture. However, the frequency of the used device in (Elsayed et al., 2008) (425 Mhz from the datasheet) is greater than the frequency of the used board in this work (125 Mhz). By considering those remarks, we can consider that the proposed architecture has good hardware metrics and timing performance.

6 CRYPTOGRAPHIC ANALYSIS

In this section, an analysis of the cryptographic properties of the proposed algorithm is performed. A set of 3 test images was encrypted using the AES-128 dynamic S-box algorithm in CBC block mode. The images employed are 256x256 pixel, 8-bit grayscale pictures. The experiments were performed using a Vitis HLS testbench written in C++, and the data was extracted and plotted using MATLAB and Excel on an Intel (R) Core (TM) i5-8250U CPU operating at 1.60 GHz, running Microsoft Windows 11 (64-bit), with 16 GB of RAM.

Table 2: Resource utilization for the Zynq xc7z020.

Mode	Slices	Slice LUTs	Slice Registers	Fmax MHz	Throughput Gbit/s	Efficiency Mbps/Slices
CBC	3287	10102	7795	540.54	69.19	21.04

Table 3: Comparison of results with other AES architecture designs.

Reference	Device	F/MHz	Mode	LUTs	Throughput
[26]	xc5vlx85	576.07	CTR	22994	73.73Gb/s
[27]	xc4vlx200	250	ECB	86806	32.00 Gb/s
[14]	xc5vlx110t	272.59	CTR	11677	34.89 Gb/s
[28]	xc2vp30	405.277	CBC	6361	108.59 Gb/s
[29]	xc7vx690t	516.8	ECB	3436	66.10 Gb/s
This work	xc7z020	540.54	CBC	10102	69.19 Gb/s

6.1 Hamming Distance and Key Sensitivity Analysis

The Hamming Distance (HD) is used in cryptography to measure the difference between two-bit strings (keys) of the same size. Figure 4 shows the hamming distance between plain and encrypted images. To understand how sensitive the algorithm is to the secret key, two tests were performed, by measuring the distance between the same encrypted plaintext twice by slightly changing one bit of the secret key. This process gives two different encrypted texts, which are compared to measure the effect of the key change on the encrypted output (Mahdi et al., 2023). Ideally, half of the bits, or 50%, should change by just changing one bit of the secret key. We repeated this process for 100 different pairs of random keys to understand how a slight change (one bit) in the secret key can cause a change in the encrypted texts. Figure 5 shows the hamming distance between two encrypted images, where only one bit of the secret key was changed. Equation 6 was used to calculate the hamming distance.

$$HD(C_1,C_2) = \frac{1}{|N|}\sum_{k=1}^{N}(C_1[k] \oplus C_2[k]) \times 100\% \quad (6)$$

The results obtained are robust and close to the optimal value of 50% within a small variation of $+0.2/-0.22\%$.

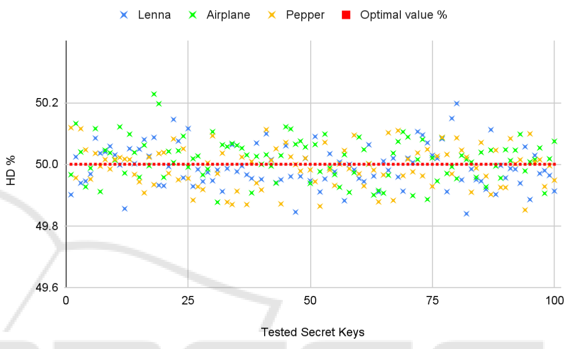


Figure 4: Hamming distance between plain and encrypted images.

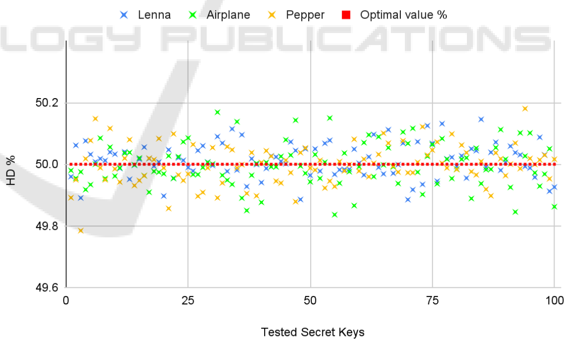


Figure 5: Hamming distance for key sensitivity analysis.

6.2 Analysis of the Uniformity of the Keystream Distribution

Analyzing the uniformity of the keystream distribution gives us a better understanding of how evenly distributed the keystream generated by the stream cipher is across its possible values. The original three plain images (Lena, Airplane, an Pepper) (see Figures 6 (a), (i), and (q)) and their corresponding distributions are shown in Figures 6 (e), (m), and (u), respectively. The encrypted images from Lena using three different s-boxes (0x23, 0x6f, and 0xd7) are shown in Figures 6 (c), (d), and (e) and their corresponding distributions are shown in Figures 6 (g), (h), and (i), respectively. Similarly for images encrypted from Airplane, and Pepper. Analyzing distribution of each of the encrypted images, it is easy to remark that represents a uniform distribution.

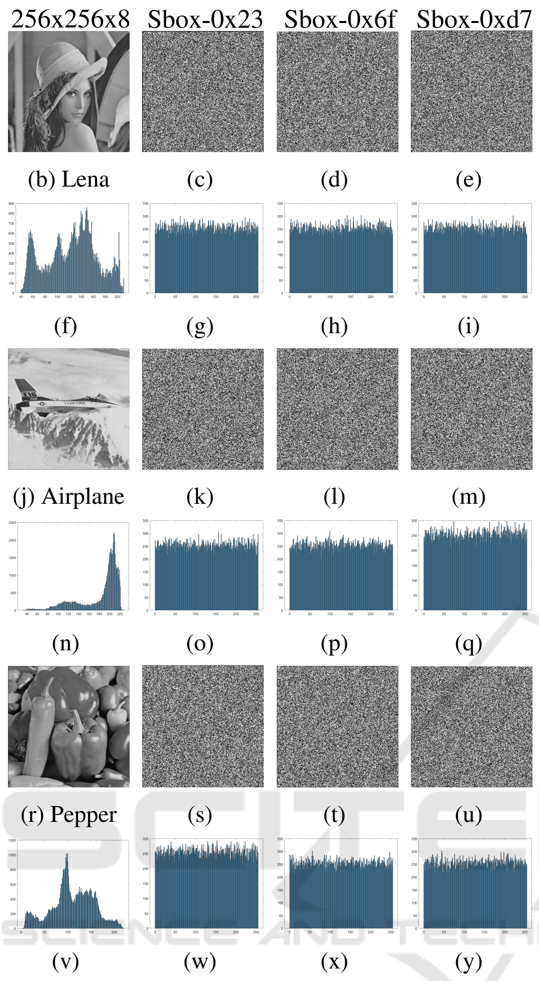


Figure 6: Distribution of plain and encrypted images.

6.3 CHI-Square Analysis

A Chi-square analysis was tested using Equation 7 over 100 different random keys, to confirm the uniformity of encrypted images in our proposed design and the results obtained in the histograms of Figure 10. The Chi-square test, or analysis, is a statistical approach used to compare observed frequencies of occurrence against expected frequencies of occurrence.

$$\chi_{exp}{}^{2} = \sum_{i=1}^{N_c-1} \frac{(O_i - E_i)^2}{E_i} \qquad (7)$$

$\chi_{exp}{}^{2}$: chi squared,O_i: observed value,E_i: expected value, $N_c = 256$ levels, $E_i = n_b/N_c$,$n_b = 65536$ for 256×256 images.

Given that the theoretical value for $\alpha = 0.05$ and $N_c = 256$ is $\chi_{th}{}^{2}(255, 0.05) = 293.24$, and the experimental Chi-Square results from Table 4 are $\chi_{exp}{}^{2} =$ 257.04, 257.29, and 253.25, respectively. We can conclude that the uniformity of the cipher-text is confirmed because the experimental values are smaller than the theoretical ones.

Table 4: Comparison of results with other AES architecture designs.

Image	HD	χ^2_{exp}
Lenna	0.499923	257.04312
Airplane	0.500170	257.2930
Pepper	0.499911	253.2499

6.4 Key Space Analysis

The key space is a method to determine the number of combinations to perform on an algorithm to determine its ability to stand against cryptographic attacks such as brute force. Our proposed dynamic s-box contains 2^{128} different key combinations as in standard 128-bit AES, but this complexity is compounded by 2^{8} with the presented scheme of dynamic S-boxes, for a final complexity of 2^{136}, which makes the relationship between the plain images, secret keys, and the encrypted images, very complex and robust. Making it resistant to brute force attacks even when a pair of plain-encrypted data is known (Mahdi et al., 2023).

7 CONCLUSIONS

In this paper, a method for adding dynamic S-boxes to AES was evaluated and implemented on an FPGA target device. The cryptographic characteristics of the dynamic S-boxes were compared to those of the AES S-box and found to be equivalent. The FPGA design implementation for the modified AES algorithm was evaluated, and the resource utilization and performance were reported and compared with those of previous works. An overall cryptographic analysis was performed on the modified algorithm, evaluating key aspects such as uniformity of the keystream, key sensitivity analysis, and key space analysis, obtaining results that confirmed the robustness of the proposed algorithm.

In the future, we will explore ways to optimize the HLS design and extend the tests performed to the 192-bit and 256-bit variants. It is possible to modularize the exported design to use it as a testbed for different S-box research based on AES. Another additional step that would speed testing and evaluation, would be the design of a PYNQ overlay to perform faster tests by using the Python programming language. Additional functionality, such as padding, may be included, but it was not considered in this work. All the sample data fed to the algorithm has been formatted to multiples of 16 bytes (128 bits).

ACKNOWLEDGEMENTS

We would like to thank Xilin university donation program for the send Pynq-Z2 board, and the IEM Department of Computer Science, Electronics, and Mechanics of the UFR Sciences and Techniques, of the University of Burgundy which opened the doors of its classes and laboratories to AESE (Advanced Electronics Systems Engineering) Master 2 students to successfully complete this project.

REFERENCES

Arrag, S., Hamdoun, A., Tragha, A., and Khamlich, S. (2013). Implementation of stronger aes by using dynamic s-box dependent of master key. *Journal of Theoretical and Applied Information Technology*, 53:196–204.

Bui, D.-H., Puschini, D., Bacles-Min, S., Beigné, E., and Tran, X.-T. (2017). Aes datapath optimization strategies for low-power low-energy multisecurity-level internet-of-things applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(12):3281–3290.

Chen, W. and Wang, L. (2019). High-speed fpga implementation of aes with dynamic s-box using pipeline architecture. *Journal of Cryptographic Engineering*, 9(4):289–302.

Dubertret, G. (2023). *Initiation à la cryptographie avec Python*. De Boeck supérieur.

Elsayed, G., Elramly, S., Hasan, B., and Shehata, K. (2008). An efficient implementation of cbc mode rijndeal aes on an fpga. pages 1–8. IEEE Xplore.

Fan, C.-P. and Hwang, J.-K. (2007). Implementations of high throughput sequential and fully pipelined aes processors on fpga. pages 353 – 356. IEEE Xplore.

Guzman, I., Nieto, R., and Norena, A. (2016). Fpga implementation of the aes-128 algorithm in non-feedback modes of operation. *Dyna (Medellin, Colombia)*, 83:37–43.

Li, Y., Zhang, Y., and Tian, Y. (2017). A high-performance fpga implementation of aes-128. In *In 2017 IEEE 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0228–0232.

Liu, Q., Xu, Z., and Yuan, Y. (2013). A 66.1 gbps single-pipeline aes on fpga. pages 378–381. IEEE Xplore.

Mahdi, M., Safwan, E. A., Camel, T., Mark Joseph, V., El-Bay, B., and Olivier, D. (2023). Fpga-based implementation of enhanced zuc stream cipher based on dynamic s-box. In *Proc. of the 9th International Conference on Engineering and Emerging Technologies (ICEET)*. IEEE Xplore.

Mamun, A., Rahman, S., Shaon, T., and Hossain, M. A. (2017). Security analysis of aes and enhancing its security by modifying s-box with an additional byte. *International journal of Computer Networks & Communications*, 9:69–88.

Nandan, V. and Gowri Shankar Rao, R. (2022). Low-power and area-efficient design of aes s-box using enhanced transformation method for security application. *International Journal of Communication Systems*, 35(2).

Nissar, G., Garg, D., and Khan, B. (2019). Implementation of security enhancement in aes by inducting dynamicity in aes s-box. *International Journal of Innovative Technology and Exploring Engineering*, 8.

Priya, S. S., Karthigaikumar, P., Siva Mangai, N. M., and Kirti Gaurav Das, P. (2017). An efficient hardware architecture for high throughput aes encryptor using mux based sub pipelined s-box. In *Wireless personal communications*, volume 94, pages 2259–2273.

Qu, S., Shou, G., Hu, Y., Guo, Z., and Qian, Z. (2009). High throughput, pipelined implementation of aes on fpga. *2009 International Symposium on Information Engineering and Electronic Commerce*, pages 542–545.

Rahimunnisa, K., Karthigaikumar, P., Soumiya, R., Jayakumar, J., and Kumar, S. S. (2014). Fpga implementation of aes algorithm for high throughput using folded parallel architecture. In *Security and communication networks*, volume 7, pages 2225–2236.

Rijmen, V. and Daemen, J. (2001). Advanced encryption standard. In *Proceedings of federal information processing standards publications, national institute of standards and technology*.

Smith, J. and Johnson, R. (2018). Efficient fpga implementation of aes with dynamic s-box for iot devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(6):1123–1135.

Wang, S.-S. and Ni, W.-S. (2004). An efficient fpga implementation of advanced encryption standard algorithm. In *2004 IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 2, pages 11–597.

Waqas, U., Afzal, S., Mir, M., and Yousaf, M. (2015). Generation of aes-like s-boxes by replacing affine matrix. pages 159–164.

Webster, A. F. and Tavares, S. E. (1986). On the design of s-boxes. In Williams, H. C., editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 523–534, Berlin, Heidelberg. Springer Berlin Heidelberg.

Zhang, Q. and Li, H. (2020). Secure fpga implementation of aes with dynamic s-box for embedded systems. *ACM Transactions on Embedded Computing Systems*, 19(3):Article 15.

Zhang, Y., Wang, Z., and Yang, J. (2016). A high-speed and low-area aes implementation on fpga. In *In 2016 10th International Conference on Computer Science and Network Technology (ICCSNT)*, pages 1–4.