Received 26 January 2023, accepted 28 February 2023, date of publication 10 March 2023, date of current version 5 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3255779

RESEARCH ARTICLE

Design and Implementation of a Configurable Encryption System for Power-Constrained Devices

J. D. J. DE LA ROSA-DE LA ROSA [1,2], J. S. MURGUÍA [2], M. MEJÍA-CARLOS [1,2], AND MIGUEL ÁNGEL LASTRAS-MONTAÑO [1,2]
[1]Instituto de Investigación en Comunicación Óptica, Universidad Autónoma de San Luis Potosí, San Luis Potosí 78210, México
[2]Facultad de Ciencias, Universidad Autónoma de San Luis Potosí, San Luis Potosí 78295, México

Corresponding authors: M. Mejía-Carlos (marcela.mejia@uaslp.mx) and Miguel Ángel Lastras-Montaño (miguel.lastras@uaslp.mx)

This work was partially supported by the Consejo Nacional de Ciencia y Tecnología (CONACYT) under Grant CB2017-2018-A1-S-45697. J. D. J. De La Rosa-De La Rosa was supported by a CONACYT Scholarship under Grant CVU 931148. The authors thankfully acknowledge the computer resources, technical expertise and support provided by the Laboratorio Nacional de Supercómputo del Sureste de México, CONACYT member of the network of national laboratories, under project 202201019N.

ABSTRACT In this work, we present a configurable encryption system based on the Encryption by Synchronization in a Cellular Automata (ESCA) system, which is a symmetric key algorithm based on the synchronization phenomenon of Cellular Automata with rule-90. With the aim of producing a flexible system to trade-off power consumption and security level, we implemented a pseudo-random number generator (PRNG) that can be configured with three different key sizes. This variable-length PRNG, together with the capability of bypassing specific modules in the rest of the system, allow us to operate under a wide range of applications. In particular, it would enable online adjustments in IoT and power-constrained devices to fine-tune them between a low-power consumption and a maximum-security level. The system can be implemented with 5956 gates, and it is designed to provide in a 0.5 μm CMOS process a throughput of 50 Mbps @ 37 mW, at the maximum-security level, and an energy consumption of less than 7 mW @ 30 Mbps at the lowest-security level, while still providing a satisfactory perceptual security metric.

INDEX TERMS Cipher, cellular automata, symmetric cryptosystem, CMOS, ASIC.

I. INTRODUCTION

Nowadays, embedded systems are used in a wide variety of applications such as wearable devices, medical implants, Internet of Things (IoT) devices, Radio-frequency identification (RFID) tags, and Wireless Sensor Networks (WSN). An important property of these systems is their capacity to securely store, access, and transmit information, and the process responsible for safeguarding this information does not have a low energy cost [1], [2]. To achieve this, different types of encryption systems have been developed to safeguard the confidentiality and integrity of information.

A problem to overcome is that embedded systems are typically battery-operated, and the time and energy costs of using an encryption system are high [3], [4], [5]. Such energy-constrained embedded systems require a low-cost specialized encryption process with low energy consumption and a small footprint area [6], [7]. Instead of implementing these encryption systems in software or general purpose hardware (e.g. single-board computers, microcontrollers, or FPGAs), an Application-Specific Integrated Circuit (ASIC) approach is preferred.

An encryption process is a tool to protect information, transforming the data into illegible text in such way that only the recipient is able to return to the original form of the information. Some of the most common encryption systems are AES [8], DES [9], and 3DES [10]. The main problem with the DES system is that, with the current computing power, it can be broken with using a brute force attack. To solve this problem, the 3DES system was implemented, which

The associate editor coordinating the review of this manuscript and approving it for publication was Jun Wang.

basically uses triple DES systems with 3 different keys, but with the disadvantage that it increases the execution time and consumption of different resources such as energy or silicon area for its implementation. The AES system supports 128, 192, and 256 bit keys. The process consists of four stages, and 10, 12, or 14 rounds, depending on the key size, which can be compute-intensive, and prohibitive for some energy-aware applications. Currently, there are other types of encryption algorithms that have been proven to be compared and sometimes better than the AES system, much of which depends on the type of information being encrypted and the means by which it is transmitted.

Cellular Automata (CA) systems provide simple algorithms owing to their high parallelism, homogeneity, and easy implementation in software and hardware systems [11], [12], making them good candidates for designing cryptosystems. CAs are discrete dynamical systems that evolve in discrete time steps and have been used to simulate biological systems [13], [14], road traffic [15], epidemics [16], surface water flow [17], and cryptography [18], [19], among others. In recent years, many encryption systems based on CA have been developed, such as those presented in [20], [21], [22], [23], [24], and [12], which are based on the evolution of cellular automata states. In [25], the synchronization phenomenon of CA was applied to develop an encryption system, now known as ESCA, composed of an indexed family of permutations and a pseudo-random number generator (PRNG) based on the forward and backward evolution of the CA rule-90. The indexed family permutations are the block cipher functions of the proposed scheme and are applied to plain-text and cipher-text by employing a key produced by the PRNG.

In this paper, we propose a configurable ASIC implementation of the ESCA system. The fundamental transformations are built around additive CA rules [26], [27], which makes them suitable for hardware implementations. The implemented system allows a trade-off between power consumption and security level, depending on the needs, by disabling selected modules and reusing hardware. We applied statistical and sensitivity tests to the proposed system to corroborate its high level of security, and validated its functionality in software and on an FPGA.

The rest of the paper is organized as follows: Section II presents an overview of the ESCA system and in Section III we review different tests applied to the encryption system to corroborate its security. Section IV describes the hardware implementation in ASIC technology, and Section V details its performance and power consumption. Section VI presents the implementation results and a functional validation of the system with FPGA and software implementations. Section VII presents conclusions and a brief discussion of the study.

II. ESCA SYSTEM OVERVIEW

The ESCA system is a symmetric block cipher of key- and data-variable lengths, based on the synchronization phenomenon of CA that evolves according to the local rule-90 [25], [28]. The system is conformed by an asymptotically perfect pseudo-random number generator [26] employed to generate keys for two indexed families of permutations, Ψ and Φ, called encryption and decryption functions, respectively. The system can be implemented using a Basic Unit Cipher (BUC), as shown in Fig. 1a, which consists of a space-pattern conformed by the words $x = (x_0^0, x_0^1, \ldots, x_0^{N-1})$, $t = (x_2^0, x_3^0, \ldots, x_{N+1}^0)$, $m = (x_1^N, x_2^N, \ldots, x_N^N)$ and $c = (x_{N+1}^0, x_{N+1}^1, \ldots, x_{N+1}^{N-1})$. These words can be initialized, and the automata can then be evolved, according to rule-90, in the forward or backward directions to find the desired word, depending on the task required. The word c is obtained by iterating backward in time using as initial conditions x and m, and correspond to the indexed family permutation Ψ_t. The word m is obtained iterating forward in time employing as inputs x and c, and corresponds to the indexed family Φ_t. On the other hand, word t is generated iteratively backward in time using x and y, and correspond to the function $t = h(x, y)$ [25], [27]. A simpler version to implement a BUC is the one-time algorithm, in which these words are calculated using the XOR sum of only the required elements.

In addition to the indexed family permutations, a pre-processing stage based on the structure of the PRNG was added to obtain a secure and flexible cryptosystem [29]. In addition, a substitution box (S-box) based on the cellular automata rule-90 [30], was added to confer greater security to the system. In [27], an ESCA system matrix approach was proposed to have a simple implementation when larger block sizes are required, and is the notation used in this work. The encryption procedure starts with the pre-processing stage, where the plain-text is transformed to an unintelligible form; afterwards, the S-box applies a non-linear transformation; finally, the data are encrypted with permutation Ψ, and the key is generated to be employed in the permutation. The decryption process is carried out in reverse order.

The proposed implementation operates on an 8-bit block of data, and the supported key sizes are listed in Table 1. The cryptosystem can be configured in 15 different modes, bypassing one, two or three modules, as well as three key sizes. The notations used to denote the modes are listed in Table 1. It consists of five bits divided into two groups. The first group of three bits represents the enabled modules, where the first bit enables the permutation, the second bit the S-box, and the third bit the pre-processing module. The remaining group of 2 bits sets the key size of the PRNG: 00 for 15 bits, 01 for 31 bits, and 10 for 63 bits. For instance, the 100-01 mode enables the Ψ module only and set the key size to 31 bits, and mode 111-10 enables all modules and is set to 63 bits of the key size. The sub-blocks of the encryption system based on the matrix approach are described below.

A. PSEUDO-RANDOM NUMBER GENERATOR

We define a PRNG of size $N = 2^n - 1$, for $n = 1, 2, 3 \ldots$, as a module that takes an input of size $2N + 1$, and produces a sequence of outputs of size N, with the property that larger modules, that is, those with larger N values, produces higher-quality sequences [11].

The keys employed in Ψ and Φ permutations, are generated by the PRNG, and can be calculated using

$$U^{i+1} = H_N U^i, \quad \text{for } i = 1, 2, 3, \ldots, \tag{1}$$

where, $H_N = (H_{top}; H_{bot})$ is a square matrix of size $(2N+1) \times (2N+1)$, where $N = 2^n - 1$ for $n = 1, 2, 3, \ldots$, and $U^i = (x; y)^T$ is a vector of $2N+1$ elements. The matrix H_{top} of size $N \times (2N+1)$, is produced by applying an addition modulo 2 operation of the two previous rows, with previous row shifted to the right by one position, given two initial vectors, and correspond to $v = [v_1, 0, \ldots, v_{N+2}, \ldots, 0]$ and $w = [0, w_2, 0, \ldots, w_{N+1}, 0, w_{N+3}, \ldots, 0]$, with $v_1, v_{N+2}, w_2, w_{N+1}, w_{N+3} = 1$. The matrix H_{bot} of size $(N+1) \times (2N+1)$, is formed by a identity matrix in the first $(N+1)$ columns and the remaining N columns are filled with zeros. In the first iteration where $i = 1$, the vector U^0 is composed by the initial conditions (seeds) x and y, of N and $N+1$ elements respectively, and the vector U^1 resulting from (1), is conformed by the first key generated in the first N elements (also denoted as t^1 at time $i = 1$), and the remaining $N+1$ elements, correspond to the feedback of the first N elements of U^0. The missing element is taken from the LSB of U^0 to be the new MSB of the remaining $N+1$ elements of U^1. In other words, H_{top} computes the key and H_{bot} feeds back the previous key for the next operation.

In this work, three different PRNG sizes were implemented, with $n = 4, 5$ and 6, in such way that $N = 15, 31$ and 63 for the key length. For instance, for $N = 7$, the equation (1) can be rewritten as

$$U_{i+1} = \left(\begin{array}{ccccccccccccccc}
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&1&0&0&0&0&0&1&0&1&0&0&0&0&0\\
1&0&1&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&1&0&0&0&1&0&1&0&1&0&0&0\\
1&0&1&0&1&0&0&0&1&0&0&0&1&0&0\\
0&1&0&0&0&1&0&1&0&0&0&1&0&1&0\\
1&0&0&0&1&0&1&0&0&0&0&0&0&0&1\\
\hline
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0
\end{array}\right)
\begin{pmatrix} x_1\\ x_2\\ x_3\\ x_4\\ x_5\\ x_6\\ x_7\\ y_1\\ y_2\\ y_3\\ y_4\\ y_5\\ y_6\\ y_7\\ y_8 \end{pmatrix} \tag{2}$$

From (2) is possible obtain boolean functions, as in (3), for an easy and suitable hardware implementation instead of evaluating the BUC or calculating the matrix-vector product, which requires implementing a more complex circuit and a greater number of operations. The binary operation $+$ is the logic XOR and the word t correspond to the number (key) generated.

$$\begin{aligned}
t_1^{i+1} &= x_1^i + y_2^i\\
t_2^{i+1} &= x_2^i + y_1^i + y_3^i\\
t_3^{i+1} &= x_1^i + x_3^i + y_4^i\\
t_4^{i+1} &= x_4^i + y_1^i + y_3^i + y_5^i\\
t_5^{i+1} &= x_1^i + x_3^i + x_5^i + y_2^i + y_6^i\\
t_6^{i+1} &= x_2^i + x_6^i + y_1^i + y_5^i + y_7^i\\
t_7^{i+1} &= x_1^i + x_5^i + x_7^i + y_8^i
\end{aligned} \tag{3}$$

By taking advantage of the CA-based PRNG structure, an efficient implementation is possible in which a larger PRNG is formed by growing the BUC as shown in Fig. 1a, allowing the generation of different key sizes using the same PRNG. For a module of size $2^n - 1$, the first $2^{n-1} - 1$ boolean functions are those of a module of size $2^{n-1} - 1$. This allows an efficient hardware implementation of smaller PRNG modules out of a larger module. We exploit this property to derive smaller but faster and more energy-efficient PRNG modules out of larger, power-hungry modules, at the expense of producing lower quality sequences.

TABLE 1. Encryption system operation modes and their seed distribution.

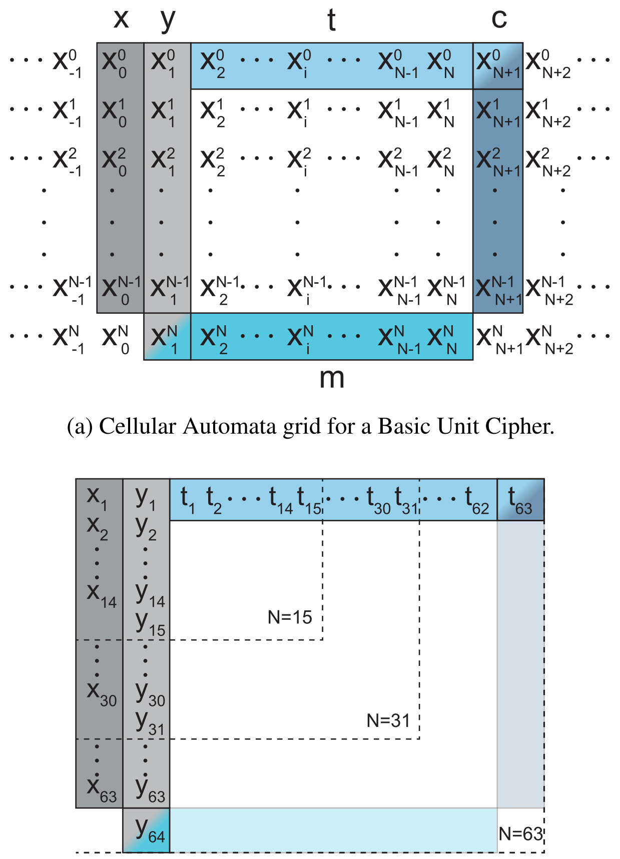
Mode	Permutation	S-box	Pre-processing	Seed size
111-10	127 bits	4 bits	9 bits	140 bits
110-10	127 bits	4 bits	n/a	131 bits
101-10	127 bits	n/a	9 bits	136 bits
100-10	127 bits	n/a	n/a	127 bits
111-01	63 bits	4 bits	9 bits	76 bits
110-01	63 bits	4 bits	n/a	67 bits
101-01	63 bits	n/a	9 bits	72 bits
100-01	63 bits	n/a	n/a	63 bits
111-00	31 bits	4 bits	9 bits	44 bits
110-00	31 bits	4 bits	n/a	35 bits
101-00	31 bits	n/a	9 bits	40 bits
100-00	31 bits	n/a	n/a	31 bits
011-11	n/a	4 bits	9 bits	13 bits
010-11	n/a	4 bits	n/a	4 bits
001-11	n/a	n/a	9 bits	9 bits

B. TRANSFORMATION PERMUTATIONS

The transformation from plain-text m to cipher-text c is achieved by the permutation $c = \psi_t(m)$ and is calculated using

$$c = \Psi_t(m) = [(P_N \times t) + (Q_N \times m)] \mod 2. \tag{4}$$

The key t of N elements and plain-text m of J elements, are multiplied by the matrices P_N and Q_N respectively, and then added to complete the transformation. Matrix P_N has dimensions of $J \times N$ and is generated by applying a zero-fill right shift of one position to the previous row, from the first row whose components in position $j = (2^n + 1) - 2^{i+1}$ for $i = 0, 1, 2, \ldots, (n-1)$ are equal to 1 and the rest are equal to 0 [27]. The matrix Q_N has dimensions of $J \times J$ and is generated as if it were a grid of CA by applying the 90 rule, with zeros as boundary conditions and taking the first row as $q = [q_1, 0, \ldots, 0]$, where $q_1 = 1$ [27].



(a) Cellular Automata grid for a Basic Unit Cipher.

(b) Coupling of 3 BUCs of different sizes.

FIGURE 1. **A BUC allows us to generate the families of permutations, as well as the PRNG, evolving forward or backward in time.**

To recover the plain-text, the inverse process corresponds to the permutation $m = \Phi_t(c)$, and is calculated by using

$$m = \Phi_t(c) = [(R_N \times t) + (T_N \times c)] \mod 2, \qquad (5)$$

where $R_N = [-Q_N^{-1}P_N] \bmod 2$ of dimensions $J \times N$, and $T_N = Q_N^{-1} \bmod 2$ of dimensions $J \times J$. In a symmetric encryption system, the same key must be used for both encryption and decryption processes. In contrast to the implementation of the PRNG, which is implemented as a single block, the permutation Ψ have to be implemented as three different blocks for each key size, since the single-time algorithms for each key size are different.

C. PRE-PROCESSING

The pre-processing transformation converts the plain-text into an unintelligible form by employing a seed z of $J + 1$ elements and the matrix H_t used in the PRNG [29]. The pre-processing transformation is calculated by

$$\hat{m} = H_t \binom{m}{z} \mod 2. \qquad (6)$$

For the first transformation, z is employed, and for the remaining calculations, the last computed $\hat{m}$, becomes the first elements of the next z and for the last one element, the first element of the previous z takes place.

$$m = M_N \binom{\hat{m}}{z} \mod 2 \qquad (7)$$

Matrix M_N has the same dimensions as H_t and is made of matrices Q_N and D_N, such that $M_N = (Q_N|D_N)$. Matrix D_N contains a value of 1 on the elements of the first superdiagonal line, and on diagonals $k_a = 2^a - 1$, for $a = 1, 2, \ldots, n-1$. The Q_N matrix is described in detail in Section II-B. The same key is used for the inverse process.

D. SUBSTITUTION BOX

Since the CA rule 90 has presented attractive properties in cryptography such as flexibility to adapt in an algorithm of massively parallel computations in computing architectures, randomness, multifractal features, among others [26], [27], [31], [32], previous works [27], [31], [32] have considered a matrix approach based on this CA rule to implement in an adjustable way the main elements of an encryption system. To assess this matrix approach, its performance was evaluated and analyzed by several tests, and the results in security issues have demonstrated good performance required for many cryptographic applications [27], [31], [32]. Following this matrix approach and taking advantage of the appealing properties of the CA rule, in [30] a square generating matrix K was implemented to generate a strong S-box, where the cryptographic strength was evaluated by standard tests, such as strict avalanche criterion and nonlinearity, among others, showing a high performance of this S-box compared with other existing S-boxes [30]. Of course, there exists different manners to implement the matrix K to generate a strong S-box, but we considered this. Since, we are considering to carry out an extensive examination with different initial condition vectors to generate matrices, and evaluate the performance in a complete encryption system, a thorough investigation of these issues is left for a future work.

Basically, an S-box is a non-linear substitution applied to a word of eight elements of a state matrix to be transformed to a different value. The transformation is obtained using

$$s = \begin{cases} (K \times x^T)^{-1} & \text{if } x \neq 0 \\ 0 & \text{if } x = 0, \end{cases} \qquad (8)$$

where x is the word to transform and K is a matrix based on the CA rule-90 with dimensions of 8×8 elements, and is formed by the matrices K_L and K_R, with dimensions of 8×4 each. Similar to the matrix Q_N, the matrix K_L can be generated by applying the CA rule-90 with zeros as boundary conditions for the next three rows, taking as the initial sequence, the vector $a = [0, 0, 0, 1, 0, 0, 0, 0]$, then, the matrix is rotated 90 degrees in the clockwise direction. Matrix K_R is generated by multiplying K_L with matrix P. Matrix K is formed as $K = (K_L|K_R)$ [30].

The S-box detailed above does not satisfy the condition of not having fixed points or opposite fixed points; thus, the implemented S-box was generated from the initial vector $a = [0, 1, 0, 0, 1, 1, 0, 0]$ to get only one fixed point. Equation (9) shows the matrix generated from the vector previously mentioned.

$$K = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{9}$$

Similar to the AES S-box [33], to remove fixed points, the sum of a constant was added, modifying the transformation (8) to $(K \times x^T)^{-1} + C$, for $x \neq 0$ and C for $x = 0$. Valid constants were extracted as follows:

1,	3,	20,	30,	31,	35,	36,	37,
64,	68,	71,	85,	87,	109,	118,	126,
129,	137,	146,	168,	170,	184,	191,	220,
224,	225,	235,	252,	254.			

The initial condition used for the generation of the S-box was selected from 2^8 possible combinations based on the number of fixed-points and a greater number of valid constants for its sum. In addition, the S-boxes that met the above conditions were synthesized to compare the area in silicon they occupied to select the one with the smallest size.

To recover the input data, the inverse function is employed using (10), which reverses the S-box operation by adding the same constant. Then, the multiplicative inverse is calculated, and the inverse affine transformation is applied.

$$x^T = \begin{cases} K^{-1} \times (s + C)^{-1} & \text{if } x \neq 0, \\ s + C & \text{if } x = 0. \end{cases} \tag{10}$$

III. SECURITY ANALYSIS

To measure the efficiency of the encryption system, the most common tests applied in cryptography [31], [34], [35], [36] were carried out, such as statistical tests including histogram analysis, correlation analysis, entropy analysis and PSNR analysis, as well as sensitivity tests such as differential attacks analysis and key sensitivity analysis. In addition to these tests, detrended fluctuation analysis (DFA) was performed to evaluate the performance of the system [37]. The tests were performed on a database of 60 images[1] and their corresponding cipher-images for the 15 modes. The keys employed for the PRNG, pre-processing and S-box are as follows: $x = 4131211101987654321$ and $y = 9223372036854775807$ for a 63-bit PRNG, $x = 1987654321$ and $y = 2147483647$ for a 31-bit PRNG and, $x = 14321$ and $y = 32767$ for a 15-bit PRNG, $c = 87$ and $z = 171$. Fig 2 shows the cipher-images in the different operating modes of the system using Lena's image.

[1]Available at http://sipi.usc.edu/database

A. STATISTICAL TESTS

DFA is a technique used to analyze singular or fractal behaviors that can be present in images. This is done by extracting the Hurst indices of sequences in the plane of the image, and then, the average of the scaling exponent can be estimated [37]. To resist a common attack, a good cryptosystem should map the plain-image to a random cipher-image without exhibiting any pattern, and the image pixels should take as many random values as possible [38]. This can be measured in terms of the fractal dimension of the cipher-image and therefore by its scaling fluctuation exponent [39]. In [40] DFA and multifractal-DFA were generalized for higher dimensions, and 2D-DFA was subsequently employed to distinguish the fractal and multifractal properties of synthetic surfaces. The scaling fluctuation exponent $\alpha = 0.5$ means that the pixels are uncorrelated or short-range correlations (e.g., white noise). For $0 < \alpha < 0.5$, the correlation in the signal is said to be anti-persistent, where an increment is highly probable to be followed by a decrement and vice versa. In the case of $0.5 < \alpha < 1$, the signal is said to be persistent, where an increment is highly probable to be followed by an increment and vice versa. The values $\alpha = 1$ and $\alpha = 1.5$, correspond to $1/f$ noise and Brownian motion, respectively [41]. For secure encryption, the scaling exponent should be close to 0.5 and 1 for one-dimensional DFA and 2D-DFA, respectively [37], [39].

The results of the 2D-DFA tests show that in all modes, except mode 010-11 (S-box only), the scaling exponents α are close to 1 (Table 2), which suggests that there is no correlation between pixels and that there is no visually comprehensible information, whose pixels appear totally random. As pointed out in [39], this metric seeks to evaluate the level of perceptual security, so we can conclude that except for the 010-11 mode, the rest of the encryption modes are secure from the point of view of perceptual security. Figure 2 shows a plain-image and the cipher-images in the different operation modes, which do not reveal any information that would allow us to distinguish the original image.

In addition to the DFA, histogram and entropy analyses were performed to verify the level of randomness. In the case of a cipher-image histogram, a uniform distribution means that each possible pixel value has the same probability to appear, which is ideal for a cipher-image because it can hide the redundancy of plain-images [31], [36]. The chi-square test was used to check whether the given histogram fit a normal distribution, and using a level of significance $\alpha = 0.05$, we obtained $\chi^2(255, 0.05) = 293$. Comparing the chi-square values obtained from the ciphered images, as shown in Table 2, our null hypothesis is accepted; therefore, the histograms fit a uniform distribution, except for the 010-00 mode. Additionally, the entropy analysis is a measure of the degree of randomness in the image and shows the distribution of pixel values without considering spatial information. For grayscale images, the ideal value for a cipher-image is 8. Table 2 lists the entropy values obtained for all configurations. We can see that all results are close

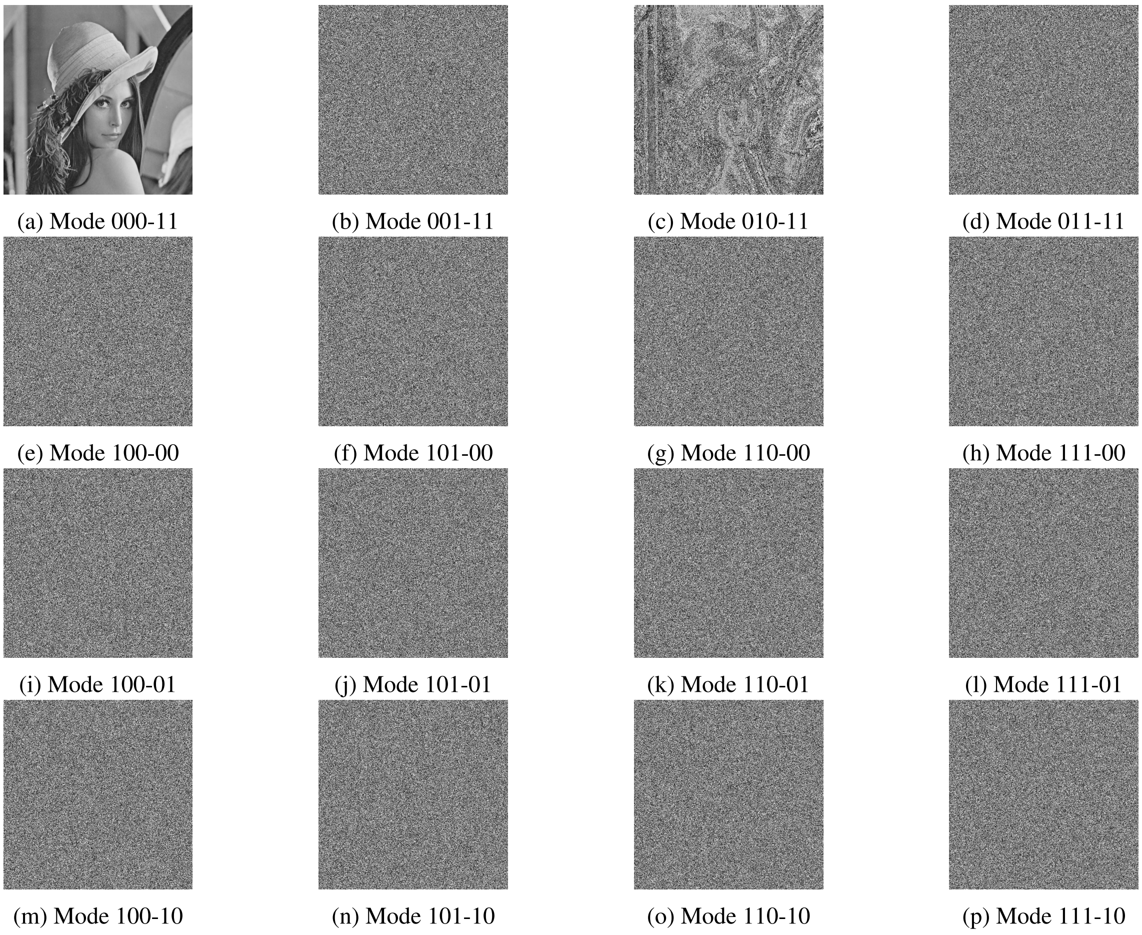


FIGURE 2. Image of Lena ciphered in the different operation modes.

to the theoretical value except for the 010-11 mode (S-box only), obtaining the same result for both, because the S-box maintains the frequency of the histogram, and the entropy is insensitive to spatial information.

The Peak Signal-to-Noise Ratio (PSNR) represents the ratio between the maximum possible value (power) of a signal and the power of the distorting noise that affects the quality of its representation. The PSNR value approaches infinity when the mean squared error (MSE) value between the plain and cipher-image approaches zero, where a higher PSNR value indicates higher image quality, and a smaller PSNR value implies large differences between the images [42]. The results in Table 2 show values close to 8.7, except for the S-box configuration, which confirms the good encryption results for the remaining modes.

The Pearson correlation coefficient is a measure of linear correlation between two sets of data. In images, the correlation coefficient of adjacent pixels is very high due to the amount of redundant information; therefore, a good encryption system should present a weak correlation close to 0 [43]. As shown in Table 3, the correlation coefficients were calculated for the adjacent pixels in the horizontal, vertical, and diagonal directions, and between the cipher and plain-images. The results showed a low correlation between the cipher and plain-images, as well as a low correlation between adjacent pixels in the horizontal, vertical, and diagonal directions in all operation modes except for the 010-11 mode (S-box only), with a slight increment.

TABLE 2. Average DFA α, χ^2, entropy and PSNR values, from the complete 60 image database.

Mode	DFA	χ^2	Entropy	PSNR (dB)
plain	1.8460	1.0304E6	6.4673	-
001-11	1.0004	1.3936	7.9993	8.7269
010-11	1.3035	7.0218E2	6.4673	8.4378
011-11	1.0025	1.1513	7.9993	8.7287
100-00	1.0076	1.2036	7.9993	8.7347
100-01	1.0014	0.8354	7.9993	8.7259
100-10	0.9959	0.9694	7.9993	8.7282
101-00	1.0030	1.2268	7.9993	8.7287
101-01	1.0010	1.2187	7.9993	8.7275
101-10	1.0011	1.0210	7.9993	8.7291
110-00	1.0060	0.9380	7.9993	8.7281
110-01	0.9995	0.8923	7.9993	8.7254
110-10	1.0074	0.9703	7.9993	8.7291
111-00	0.9999	0.8697	7.9993	8.7267
111-01	1.0034	1.2210	7.9993	8.7289
111-10	1.0033	0.9951	7.9993	8.7260

TABLE 3. Average correlation coefficients between the plain-image and the cipher-image, and between adjacent pixels in horizontal, vertical, and diagonal directions, in the different operation modes.

Mode	Corr.	Corr. H.	Corr. V.	Corr. D.
plain	1	0.8437	0.8682	0.7522
001-11	-0.0001	-0.0101	0.0005	-0.0001
010-11	-0.0554	0.0989	0.0988	0.0719
011-11	0.0000	-0.0004	-0.0000	-0.0000
100-00	0.0015	0.0007	0.0002	0.0009
100-01	-0.0002	-0.0006	0.0007	-0.0001
100-10	0.0002	-0.0010	-0.0015	-0.0013
101-00	0.0001	-0.0006	0.0001	0.0002
101-01	0.0001	0.0001	-0.0006	0.0004
101-10	0.0003	-0.0000	-0.0001	0.0000
110-00	0.0002	0.0003	0.0002	0.0000
110-01	0.0001	0.0001	0.0000	0.0002
110-10	0.0003	0.0000	0.0004	0.0000
111-00	-0.0000	-0.0002	0.0001	0.0007
111-01	0.0003	0.0000	-0.0001	0.0003
111-10	-0.0001	0.0001	0.0001	-0.0002

While the above tests show poor performance with the S-box module, this does not imply that the S-box does not fulfill its purpose. The appropriate and most commonly used tests for S-boxes are the strict avalanche criterion (SAC), bit independence criterion (BIC), nonlinearity (NL), linear approximation probability (LP), and differential approximation probability (DP). These tests were performed on the S-box implemented in [30], showing that the performance is comparable to that of other S-boxes, including those used in the AES system.

B. SENSITIVITY TESTS

Differential attack analysis consists of measuring the difference between cipher-images with a slight change (i.e., a pixel) in one of the plain-images. The most common tests used to assess resistance against differential attacks on a cryptosystem, are Number of Pixel Change Rate (NPCR) and Unified Average Changing Intensity (UACI) [44], [45]. As shown in Table 4, the modes without the pre-processing module obtained NPCR and UACI values (marked with *) that can be interpreted as low resistance because they do not have the influence of the previous pixels, whereas the remaining modes demonstrated resistance against differential attack; thus, we recommend employing the pre-processing stage to guarantee a more secure encryption process.

Key sensitivity can be measured by two cipher-texts generated by the same plain-text, but with a slightly different key, and it should result in two significantly different cipher-texts. To corroborate the expected performance, the set of images was encrypted with two different keys, differing in only one bit. Table 4 shows the correlation, NPCR, and UACI parameters (marked with †), between the cipher-image sets. The results suggest that the groups of cipher-images are significantly different, with the exception of the operation mode using only the S-box.

In addition, a good cryptosystem should not reveal information when the cipher-text is decrypted with a slightly different key. As we can see in Table 4, all operation modes result in significantly different plain-text (marked with ‡), except for mode 001-11, which recover much of the original information; therefore, it is not recommended to use the pre-processing module alone. To solve this problem when using the pre-processing mode, it is possible to use the inverse pre-processing module as the pre-processing module, simply by changing the mode from encrypt to decrypt and vice versa. The results of comparing decrypted images with a slightly different key were a 0.0043 correlation coefficient, 99.6127 % NPCR and 30.5535 % UACI.

C. BRUTE-FORCE ATTACK

A brute-force attack is a method of defeating a cryptosystem by trying all the possible keys. The size of the key space should be greater than 2^{100} to be considered a secure system [38]; the greater the computational power and the smaller the key space size, the more feasible is the decoding of the message. The implemented cryptosystem uses the x and y keys in the PRNG module to generates keys for Ψ and Φ permutations, which can be configured to three different sizes. Therefore, the sizes of x and y are 15-16, 31-32 and 63-64 bits. In addition, z and c, of 9, and 4 bits, respectively, are employed in the pre-processing and S-box modules. The total key length may vary from 4 to 140 bits, making the encryption system sufficiently secure to overcome a brute-force attack, when permutation Ψ with a 63 bits key length is used.

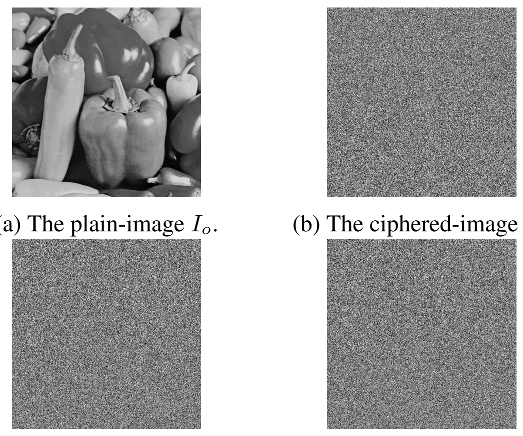
D. CRYPTOANALYSIS ATTACKS

The implemented system is shown to be secure against the above statistical tests. In [29], an analysis with the Chosen-Plain image attack was performed to an older version of the implemented system (without the S-box module), and proved to be resistant to such attacks. We performed this analysis attack for all operation modes as follows: a mask image I_m is generated by applying the bitwise XOR operation between the chosen plain-image (completely black image) and its cipher-image. Then, a cipher-image I_c is obtained from another plain-image I_o (peppers test image). Finally, we try to recover the plain-image I_o, by performing the bitwise XOR operation between the mask image I_m and the cipher-image I_c. If the obtained image reveals information from the plain image, it means that the encryption system is insecure, otherwise, the system resisted the attack.

Figure 3 shows the needed images to perform the chosen-plain image attack. The encryption mode used was all-module and 63-bit key length (mode 111-10). As can be seen in Figure 3, it does not reveal plain-image information, suggesting that the system is resistant to this type of attack. In addition, as mentioned in [29], if the encryption system is secure against Chosen-Plain image attack, it is also secure against cipher image-only attack or known-plain image attack.

TABLE 4. Average NPCR and UACI values between cipher-images with a single pixel change on one of the plain-images (marked with $*$). Average NPCR and UACI values, as well as correlation coefficients between images ciphered with slightly different keys (marked with $\dagger$), and between plain-images decrypted with slightly different keys (marked with $\ddagger$). In both cases, the keys differ in their least significant bit.

Mode	NPCR (%)*	UACI (%)*	Corr.†	NPCR (%)†	UACI (%)†	Corr.‡	NPCR (%)‡	UACI (%)‡
plain	0.0004	0.0002	1	0	0	1	0	0
001-11	99.80	33.53	-0.0017	99.80	33.53	0.9999	0.0008	30.56
010-11	0.0004	0.0001	0.9996	100	0.7843	0.0003	100	34.40
011-11	99.80	33.53	-0.0022	99.61	33.53	0.0016	99.60	33.43
100-00	0.0004	0.0002	-0.0011	99.62	33.49	-0.0011	99.62	33.40
100-01	0.0004	0.0002	0.0034	99.62	33.38	0.0027	99.62	33.46
100-10	0.0004	0.0002	-0.0027	99.62	33.52	-0.0022	99.62	33.45
101-00	99.80	33.53	0.0021	99.59	33.43	0.0001	99.60	33.47
101-01	99.80	33.54	-0.0002	99.61	33.48	0.0016	99.62	33.48
101-10	99.80	33.53	0.0005	99.61	33.46	0.0006	99.60	33.46
110-00	0.0004	0.0001	-0.0011	99.59	33.49	-0.0002	99.61	33.46
110-01	0.0004	0.0001	0.0032	99.61	33.39	0.0001	99.61	33.47
110-10	0.0004	0.0001	-0.0028	99.62	33.53	-0.0001	99.61	33.45
111-00	99.80	33.54	-0.0002	99.61	33.47	0.0004	99.61	33.47
111-01	99.80	33.53	0.0001	99.61	33.46	-0.0001	99.61	33.45
111-10	99.80	33.54	0.0004	99.61	33.46	-0.0003	99.61	33.46



(a) The plain-image I_o. (b) The ciphered-image I_c. (c) The mask image I_m. (d) The recovered image.

FIGURE 3. The chosen-plain image attack for the gray-scale peppers test image.

E. POWER ANALYSIS ATTACK

In the power analysis attack, the attacker monitors the power consumption of the encryption system, so that he can obtain information about the data being manipulated from the small movements of electrical charges on the device [46]. To corroborate the reliability of this type of analysis, we performed simulations of the entire synthesized system by varying the seed while monitoring the gates switching activity. We performed toggle counting on 100 decryption processes with the correct and incorrect key and obtained two normal-like distributions. To verify that both distributions were similar, we performed the Kolmogorov-Smirnov goodness-of-fit test [47], for which the null hypothesis is that the data from the incorrect-key decryption process were drawn from a population with a distribution of data belonging to the correct-key decryption group, and the alternative hypothesis is that they were not. We obtained that the null hypothesis is not rejected with a 0.05 significance level and p-value of 0.4688. Once configured, the implemented system processes the data in the same way regardless of the seeds, therefore the toggle count between with correct and incorrect seeds shows insignificant variations that do not reveal information about the process being carried out.

TABLE 5. Results of NIST statistical test suite of 63-bit, 31-bit, and 15-bit PRNG.

Test	63-bit	31-bit	15-bit
Frequency	0.6579	0.4373	0.1816
Block-frequency	0.3041	0.9114	0.8165
Runs	0.1816	0.5749	0.9114
Long runs of ones	0.6163	0.5749	0.5749
Rank	0.4012	0.6163	0.1917
Spectral DFT	0.8832	0.5141	0.0966
No overlapping templates	0.6579	0.2368	0.6993
Overlapping templates	0.9835	0.2368	0.5749
Universal	0.9781	0.4750	0.1626
Linear complexity	0.0004	0.7981	0.7598
Serial	0.4750	0.8832	0.2493
Approximate entropy	0.2368	0.5956	0.3838
Cumulative sums forward	0.8514	0.5956	0.7981
Cumulative sums reverse	0.4750	0.0167	0.2023
Random excursions	0.1783	0.1816	0.0705

F. NIST TEST

The NIST Statistical Test Suite (STS) is a testing suite for randomness analysis, based on statistical hypothesis testing. It comprises 15 tests that evaluate the null hypothesis over a binary sequence. From each test, we obtained a p-value that represents the probability that a perfect random number generator would have produced a sequence that was less random than the tested sequence [48]. As shown in Table 5, for $N = 31$ and $N = 63$, all tests were successful, suggesting that the PRNG produces high-quality pseudo-random numbers. For $N = 15$ bits, some statistical problems are reflected in the *Runs* test, so they are considered to be of lower quality. One thousand blocks of one million bits were used to perform the tests.

TABLE 6. Results of DIEHARD statistical test suite of 63-bit, 31-bit, and 15-bit PRNG.

Test	63-bit	31-bit	15-bit
Birthday spacings	0.0584	0.0394	0.5029
Overlapping 5-permutation	0.8585	0.7449	0.7563
Binary rank for 31x31	0.4392	0.5486	0.7704
Binary rank for 32x32	0.9151	0.9896	0.3468
Binary rank for 6x8	0.1099	0.8061	0.3147
Bitstream	0.6787	0.9322	0.1453
OPSO	0.4830	0.5092	0.2603
OQSO	0.7288	0.7208	0.1257
DNA	0.8052	0.0526	0.4305
Count-the-1's on a stream	0.4347	0.3680	0.0578
Count-the-1's for specific bits	0.5821	0.8244	0.4937
Parking-lot	0.5661	0.8194	0.6677
Minimum distance	0.0741	0.3524	0.4935
3DSpheres	0.2402	0.9560	0.5136
Squeeze	0.777	0.0211	0.8978
Overlapping sums	0.4044	0.1407	0.1040
Runs	0.9967	0.3957	0.3737
Craps	0.6299	0.3373	0.99135

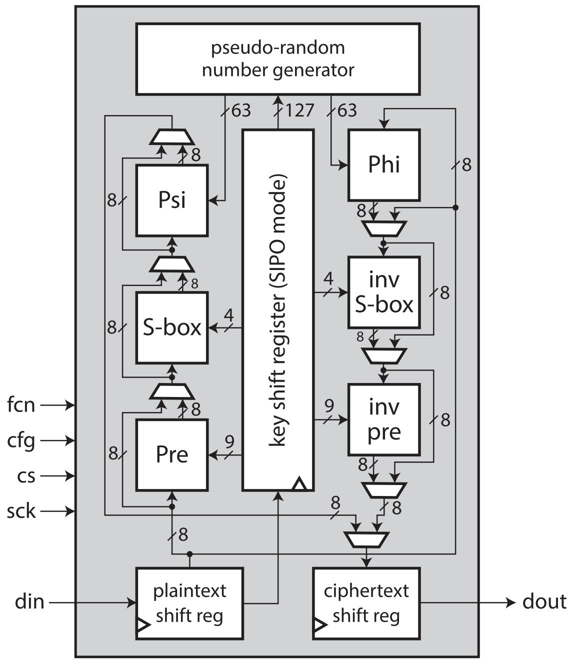
G. DIEHARD TESTS

DIEHARD tests[2] are a set of statistical tests that measure the quality of randomness in a sequence of numbers. It consists of 18 tests from which a p-value with an acceptable range of [0,1) is obtained. Table 6 shows the p-values obtained from sequences of numbers generated by the implemented PRNG of 63, 31 and 15 bits. As can be seen, all values are under acceptable range, thus "pass" the tests and we can conclude that the PRNG produces highly random keys.

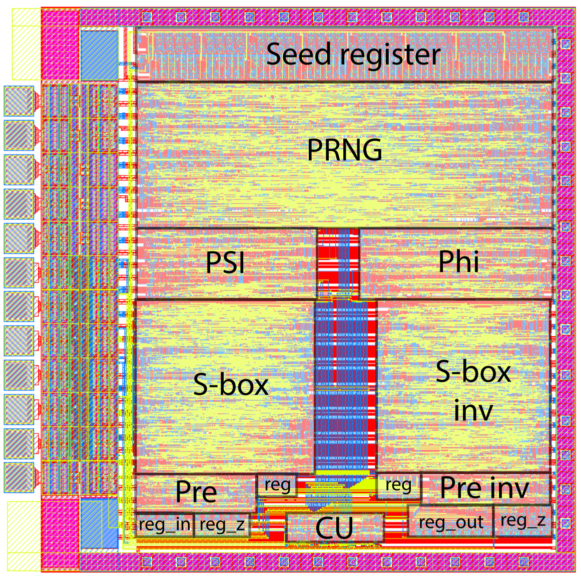
IV. HARDWARE IMPLEMENTATION

The top level architecture of the ESCA system core is illustrated in Fig. 4a. As mentioned above, the system can be configured in several modes (as shown in Table 1), with the configuration byte stored in an 8-bit shift register. This initial configuration process also includes an encryption key and the first byte to encrypt (or decrypt). The order in which they are streamed to the input is as follows: configuration byte, encryption key and the first byte to encrypt. The data stream is carried out by employing the SPI protocol, and the configuration is loaded when the chip is in the *reset mode* (active low *cfg* signal). The encryption key is stored in a 140-bit shift register, 127 bits of which are employed for x and y, 9 bits for z, and 4 bits for c. When the initial configuration stream is completed, a pulse of *cs* and *cfg* signal change, are needed to start transmitting the data to be processed. The data to be processed is entered into 8-bit blocks and stored in a shift register. When a new byte is received, it is processed, and the previous data are stored in a shift register to be sent. In other words, new data are processed every 8 cycles (every new valid 8-bit data); therefore, no clock cycles are wasted. The *data_in* and z registers were instantiated twice to improve routing area efficiency. The contents of the registers for the unused modules, are masked to zero to prevent glitches.

[2]Available at https://web.archive.org/web/20160125103112/http://stat.fsu.edu/pub/diehard



(a) High-level architecture of the ESCA encryption core.

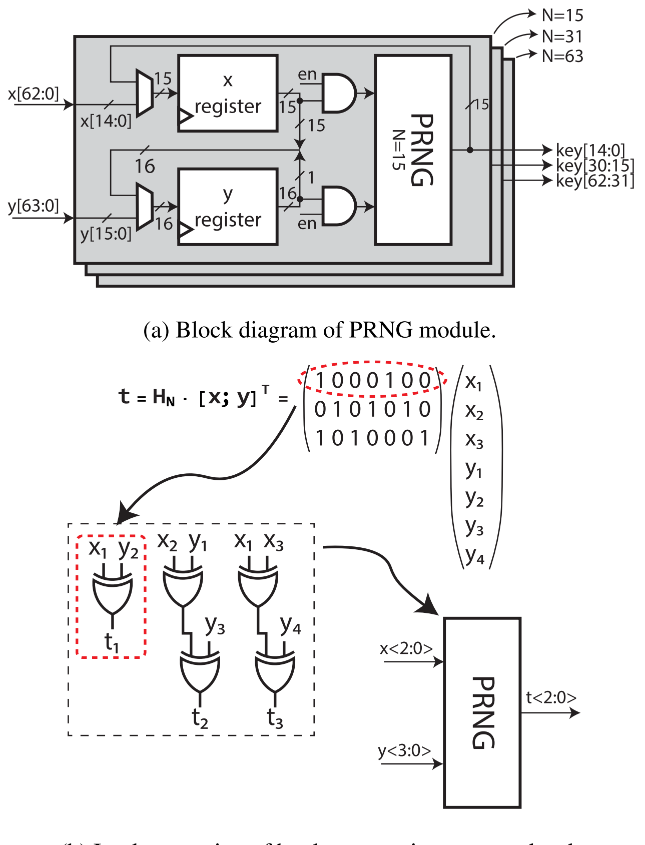


(b) Layout view of the ESCA system implemented.

FIGURE 4. The block diagram defined for the ESCA system and its final physical implementation.

A. PRNG MODULE

The PRNG module shown in Fig. 5a is composed by three sub-modules, which are enabled depending on the selected mode and each generates a key that can be concatenated to form a larger key. The smaller sub-module generates a 15-bit length key, the second generates the next 16 bits to form a 31-bit length key and the largest generates the last 32 bits to form a 63-bit length key. In each sub-module, two registers are used to store the x and y seeds. The input data of these registers are multiplexed between the initial key



(a) Block diagram of PRNG module.

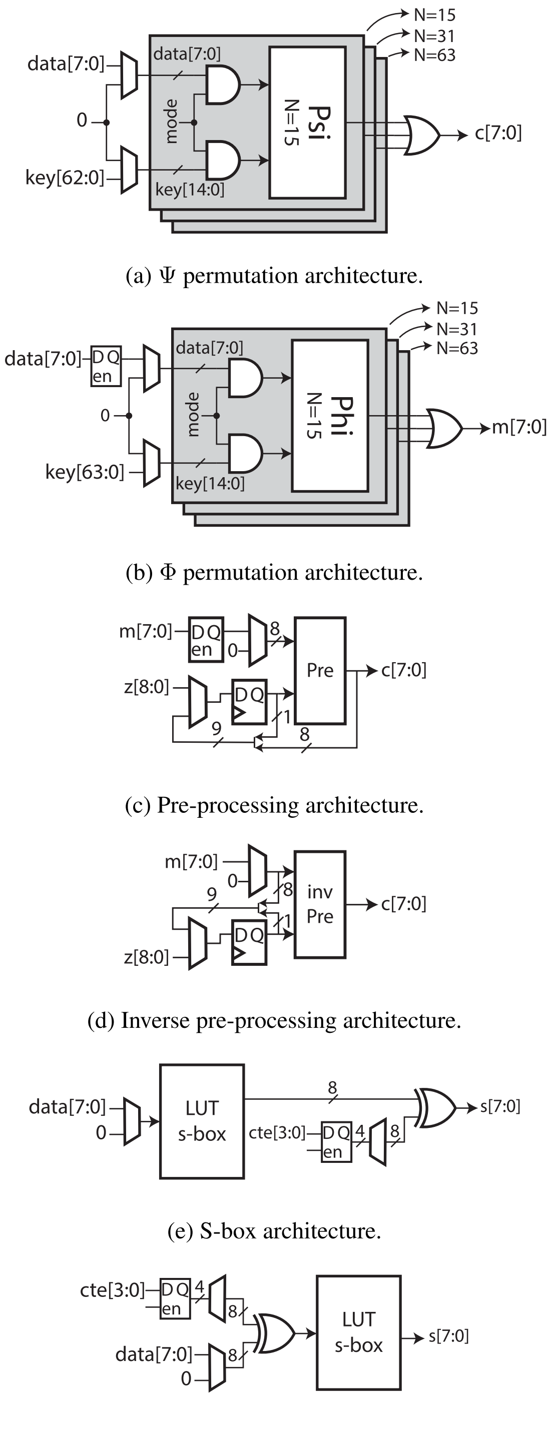
(b) Implementation of boolean equations to gate-level.

FIGURE 5. **Implemented PRNG architecture.**

and the values described in Section II-A. The outputs of the registers are gated to prevent glitches when a sub-module is not in use. These gated signals feed the block composed by the Boolean equations to calculate the key, so that a key is generated each time the clock is triggered in the registers. Fig. 5b shows the structure of the block containing the Boolean equations and XOR gates for the calculation of the key.

B. Ψ AND Φ PERMUTATIONS MODULES

The implemented modules of the Ψ and Φ permutations are shown in Fig. 6a and Fig. 6b, respectively. Each permutation consists of three different sub-modules for each key size. The datapath of the Ψ permutation starts with multiplexers for the plain-text and key, that allows masking the signals to zero values when the module is disabled. If the signals are propagated, they are fed to each sub-module, which may or may not be gated depending on the configured mode. If the module is enabled, only one sub-module can be enabled, and in it, the plain-text and the key will propagate to the block containing the combinational logic that generates the cipher-text. The sub-module outputs are ORed to generate the permutation module output. The structure of the Φ module only differs with the latches to store the plain-text, to prevent glitches while inputting the keys.



(a) Ψ permutation architecture.

(b) Φ permutation architecture.

(c) Pre-processing architecture.

(d) Inverse pre-processing architecture.

(e) S-box architecture.

(f) Inverse S-box architecture.

FIGURE 6. **Implementations of the modules that conform the ESCA system.**

C. PRE-PROCESSING MODULE

As shown in Fig. 6c, the input data is fed to latches and multiplexed to prevent glitches inside the block, which contains a combinational circuit (equivalent to the matrix-vector multiplication detailed in II-C) to the pre-processing data. The key is fed to a 9-bit register through multiplexers that select between the initial condition or the last word of the previous calculus to perform the feedback process. When the input propagates through the latches, and the key through the registers, the data is transformed by the pre-processing block. The inverse process in Fig. 6d feeds the input data to the boolean equations block through multiplexers and, simultaneously, in a 9-bit register, a key is stored through a multiplexer that selects between the initial condition and the previous input data. The data and key are processed by an inverse pre-processing block and the output data is generated.

D. SUBSTITUTION BOX MODULE

Fig. 6e shows a schematic of the S-box implemented, made mainly from a lookup table described in Verilog and synthesized to convert the description into a combinational circuit. The same process was performed for the inverse S-box. The datapath consists of a multiplexer that masks the input data which later feeds the combinational logic block that transforms the data. The transformed data is XORed with a constant that is provided from another lookup table of valid constants, which is controlled by a key stored in the initial configuration. The sum corresponds to the output of the module.

In the inverse process, the input data is multiplexed to prevent glitches and the constant is decoded from the lookup table of valid constants. The input data and the constant are XORed and the result is transformed by the lookup table that performs the byte transformation.

V. IMPLEMENTATION RESULTS AND DISCUSSION

The design was described in Verilog and synthesized using a Standard Cell library. Pre- and post-synthesis simulations were performed to corroborate its performance and the physical design was also checked with Design Rule Check (DRC) and Layout Versus Schematic (LVS) checks. A maximum operation frequency of 50 MHz, and throughput of 50 Mbps, was obtained and with all modules enabled, and a 63-bit key length. The latency is equal to the cycles needed to input the initial stream conformed by the configuration, seeds and the first data (8 bits), and may vary from 164 to 25 cycles. Figure 4b shows the layout of the chip, with an area of 4 mm^2, including the IO pads.

Table 8 lists the gate count per module and its percentage occupation. The PRNG module occupies the largest amount of area with 27%, despite the fact that the modular design to generate three different key sizes is focused on reducing area. The second largest blocks are the S-boxes, with 18% each. A possible way to obtain a more compact implementation is described in [49], where the computation of the GF(2^8) inverse is broken up into computations in smaller subfields.

The maximum operating frequency depends on the chosen operation mode. When all modules are enabled and the key size is set to 63 bits (111-10 mode), the lowest operating frequency is obtained (50 MHz); otherwise, when only the pre-processing module is enabled, the highest frequency is reached (110 MHz). The datapath to encrypt/decrypt flows through up to 3 modules (pre-processing, S-box and permutation) without any memory element between them, forming a single combinational logic path. Therefore, the propagation time is the sum of the propagation time of the 3 modules. The system can be improved by using a faster communication protocol and pipelined structure, adding memory elements such as registers or latches, at the output of each module, decreasing the final propagation time, dominated by the largest of the 3 modules, and thus increasing the performance of the system. The implemented system was designed to implement the SPI communication protocol, with a maximum standard frequency of 50 MHz. The mode of operation with all the modules enabled and 63-bit key theme (111-10 mode) can already operate at such frequency, so it was not necessary to use a pipeline structure. The higher cost of using a pipelined approach is the increment in power consumption, silicon footprint, and latency.

TABLE 7. Power consumption of all operation modes.

Mode	11	X10	X01	X00
63-bit	22 mW	21 mW	21 mW	20 mW
31-bit	14 mW	13 mW	13 mW	12 mW
15-bit	10 mW	8.8 mW	8.4 mW	7.8 mW
No PRNG	6.9 mW	5.4 mW	5.5 mW	4.5 mW

Table 7 shows the simulation results of the power consumption in all configurations with a clock frequency of 30 MHz. When the maximum security is set (111-10 mode), the chip consumes the most energy owing to the number of gate switchings of all modules in use. When the PRNG key size is set to 31 and 15 bits, the power consumption is reduced by a factor of 63% and 45% respectively. On the other hand, if the S-box module, pre-processing module, or both are disabled, consumption is reduced to 93%, 93%, and 89% respectively.

Compared to the 111-10 mode (all modules enabled and 63-bit key size), the modes 010-11 (S-box only) and 000-11 (no module activated), consume 24% and 20%, respectively and this is because the largest amount of energy is consumed by the data transmission. To achieve the difference between the modes mentioned, the transistor size of the flip-flops that compose the shift registers involved were optimized by optimizing the setup time, hold time, and C-Q delay values, to achieve minimal power consumption without affecting the previously reached performance. To corroborate this, simulations were performed with smaller register transistors, and the power consumption of *data_in* and *data_out* registers were reduced to 30% and 26%, respectively.

Table 8 shows the energy consumption of each module in the simulation configured with the 111-10 mode. The PRNG module consumes 56% of the power owing to its high-switching-activity gates. The next highest consumption comes from the *data_in* and *data_out* registers, which consume 13% of the total power. Next was the Ψ module with 12% due to the glitches caused by the propagation delay of the data in the previous modules and PRNG. The modules that are not in use, such as the inverse modules and the input

TABLE 8. Core power consumption comparison in 111-10† and 010-11‡ modes, and area distribution of the core (area in terms of equivalent 2-input NAND gates).

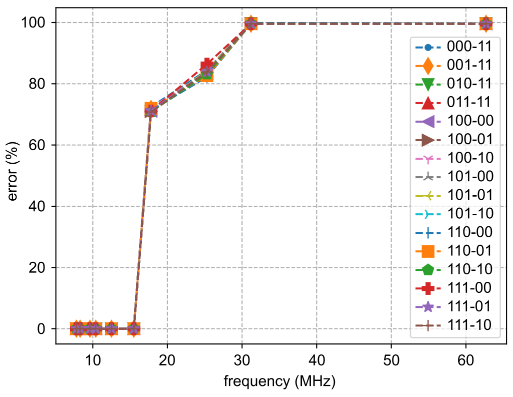
Module	Power† (W)	Power‡ (W)	Area (GE)
Pre-proc	9.1E-04 (4.1%)	2.7E-04 (4.9%)	152.7 (2.7%)
S-box	1.4E-03 (6.3%)	7.1E-04 (13%)	1047 (18%)
Ψ	2.6E-03 (12%)	5.9E-05 (1.1%)	462.7(7.8%)
Φ	2.7E-04 (1.2%)	5.6E-08 (0.0%)	496.2 (8.3%)
S-box^{-1}	8.4E-05 (0.4%)	8.4E-05 (1.6%)	1080 (18%)
Pre-proc^{-1}	8.4E-07 (0.0%)	5.4E-07 (0.0%)	158.3 (2.7%)
PRNG	1.2E-02 (56%)	7.9E-06 (0.1%)	1615 (27%)
Data_in reg.	1.2E-03 (5.5%)	1.2E-03 (23%)	39.57 (0.7%)
Data_in reg.(2)	3.0E-05 (0.1%)	3.0E-05 (0.6%)	39.57 (0.7%)
Data_out reg.	1.7E-03 (7.8%)	1.4E-03 (26%)	83.10 (1.4%)
X-Y reg.	2.5E-06 (0.0%)	2.5E-06 (0.0%)	543.4 (9.1%)
Z reg.	4.9E-06 (0.0%)	4.9E-06 (0.1%)	43.79 (0.7%)
Z reg. (2)	4.9E-06 (0.0%)	4.9E-06 (0.1%)	43.79 (0.7%)
C reg.	7.7E-09 (0.0%)	7.7E-09 (0.0%)	22.69 (0.4%)
Control unit	1.6E-03 (7.2%)	1.6E-03 (30%)	126.1 (2.1%)

register of the decryption process, have lower consumption compared to the others.

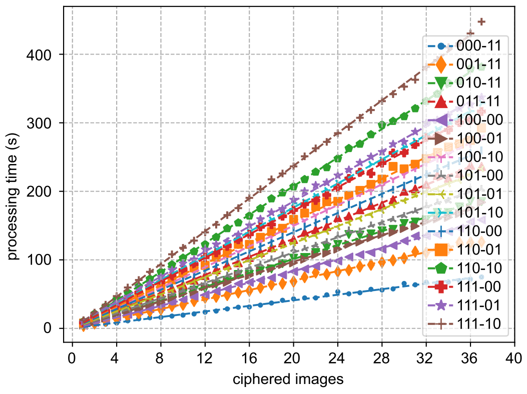
The consumption of all modules in the 010-11 mode (S-box only) is also shown in Table 8, which is the mode with the lowest energy consumption. The *data_in* and *data_out* registers use 50% of the total power. The second highest power consumption is the control unit with 30%, whereas the S-box module consumes 13%, making data transmission the most energy-consuming part. Based on the above, we recommend three operation modes (highlighted in Table 7): one with the highest level of security (highlighted in blue), one with the lowest energy consumption (highlighted in green), and one that offers a fair compromise between security and energy consumption (highlighted in red).

Mode 111-01 (all modules enabled with 31 bits key length) consumes 63% compared to 111-10 mode (all modules enabled with 63 bits key length), and in terms of security both satisfy the statistical and sensitivity tests applied; however, it has a shorter key length, therefore is more susceptible to brute force attack. Another option to save more energy is to disable pre-processing and S-box modules; however, the trade-off between energy savings and reduced security, it is not worth it. The third option has the lowest power consumption in exchange for a lower level of security, however, S-boxes are used in some applications such as watermarking [50], image encryption [51], and embedded systems [52].

To make a fair comparison between our system with other cryptosystem implementations, it is necessary to consider several factors such as: security level, round-based or serial-based ciphers, CMOS technology used, among others. Since it is difficult to apply all the above criteria, we decided to compare the ciphers using the power/frequency (μW/MHz) metric. This is in order not to favor specific metrics for evaluation. Because the 0.5 μm process consumes significantly more power than smaller processes, we do not rule out the possibility of implementing our design in newer technologies to improve the implementation.



(a) Error rate of encrypted data of a FPGA implementation through different frequencies.



(b) The elapsed time encrypting grayscale images of size 256x256 pixels, in the different encryption modes. The dashed lines indicate the linear fits.

FIGURE 7. Software and FPGA implementation performance.

In comparison to the lightweight cipher implementations in [49], [53], [54], and [55], where powers in the range of 37 to 278 μW/MHz are obtained using a 130 nm node and ranging from 3.1k to 18k gate equivalent (GE), we obtained a range of 62 to 733 μW/MHz with the 0.5 μm node. However, by implementing our system on a 130 nm node, considering the quadratic effect of decreasing voltage and capacitance [56], we estimate to reduce the system up to 10% of the current consumption, i.e., a range of 6.2 to 73 μW/MHz. In comparison with AES implementations, which is one of the most important security benchmarks, the power consumption of implementations presented in [46], [57], [58], and [59] varies in the range of 60 to 2461 μW/MHz, in which the amount of GE varies from 60k to 200k.

VI. SYSTEM VALIDATION

The proposed system was implemented in an FPGA Cyclone IV E, and a Raspberry Pi 3 Model b+ was used to test the embedded design. Figure 7a shows the error obtained by encrypting in all the modes at different frequencies. The difference in the error between the modes across different frequencies is very low, because the difference between

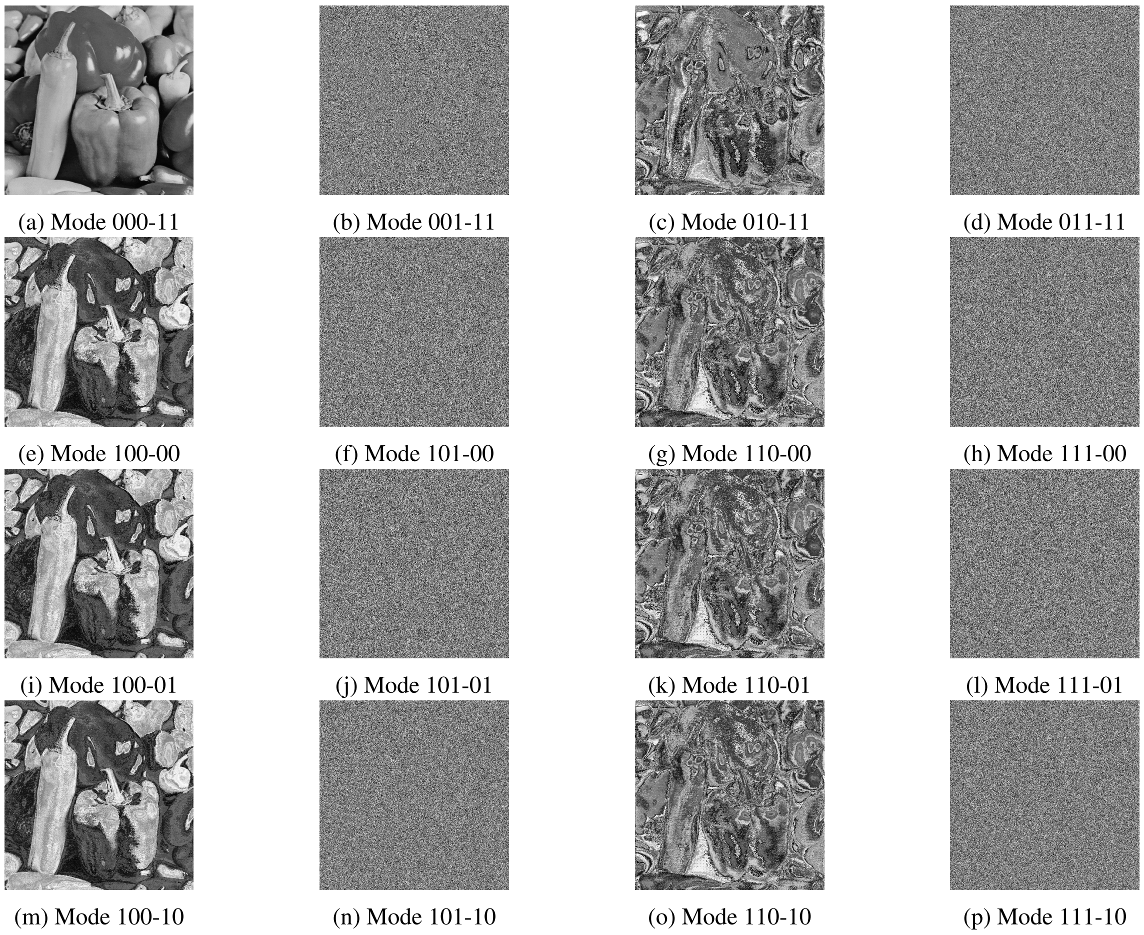


FIGURE 8. **The chosen-plain image attack for the gray-scale peppers test image.**

frequencies is a significant cause of the limitations of Raspberry Pi in generating more frequency in clock signals. The system works correctly with a maximum operating frequency of 16 MHz and uses 1341 logic elements. Despite the FPGA technology node (60 nm), the implementation in 0.5 μm CMOS technology is approximately three times faster in addition to the saving energy.

The system was also implemented in software using Python and the Numpy library on a computer with an Intel Core i7-8550U processor and 8 GB of memory RAM. Tests were performed to measure the frequency and the results are shown in Fig 7b. When all modules are disabled, that is, the cost of moving the information only, it achieves a throughput of 1 Mbps, and when all modules are enabled and 63-bit PRNG is set, it achieves 176 kbps.

VII. CONCLUDING REMARKS

In this work, we report on the ASIC implementation of a configurable ESCA system with encryption and decryption processes embedded on the same chip. The design is implemented on a 5V, 0.5 μm CMOS process. In its highest security configuration, it can achieve a maximum operating frequency of 50 MHz with a throughput of 50 Mbps (at 37 mW), which is higher than that of FPGA and software implementations of the same design. Furthermore, since the system is implemented using a Standard Cell based approach, it can be readily migrated to more advanced processes, thus improving its performance per watt metric. The unique nesting property of the PRNG implementation, in which smaller PRNG modules are embedded into larger ones, allows us to reduce by 25% the area of the PRNG module, thus resulting in a total of 5956 gates for the implementation of the entire system. Of these, 16% were used for the SPI protocol, and can be omitted in designs where it is not required. In addition, it has been demonstrated through different tests, such as histograms, DFA, correlation, entropy, PSNR, and differential attack, the high level of security of the implemented encryption system, which is especially suitable for encrypting images, as it provides a high level of perceptual security in all but one mode of operation.

We believe that this type of configurable encryption systems will be a key tool for its use in a wide range of applications, owing to the trade-off between power consumption and security level. This allows us to use the system in applications that require a higher level of security, where the power consumption of the encryption procedure is insignificant, as well as on resource-constrained devices, such as ASICs for IoT devices or IPs for system-on-chip.

APPENDIX
CHOSEN PLAIN-IMAGE ATTACKS

Figure 8 shows the result of the chosen plain-image attack in the different system operation modes. As mentioned in Section III-D, the mask images were generated using a completely black image and their corresponding cipher-image. The plain-image is the peppers test image shown in 3a. As mentioned in Section III-B, the operation modes without the pre-processing module, presented a low resistance to the differential attack analysis since they do not have the influence of the previous pixels, and also presented a weak performance against the chosen plain-image attacks, thus our recommendation of employing the pre-processing module.

REFERENCES

[1] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos, "Lightweight cryptography for embedded systems—A comparative analysis," in *Data Privacy Management and Autonomous Spontaneous Security*. Berlin, Germany: Springer, 2013, pp. 333–349.

[2] K. Yang, D. Blaauw, and D. Sylvester, "Hardware designs for security in ultra-low-power IoT systems: An overview and survey," *IEEE Micro*, vol. 37, no. 6, pp. 72–89, Nov. 2017.

[3] M. Masoud, I. Jannoud, A. Ahmad, and H. Al-Shobaky, "The power consumption cost of data encryption in smartphones," in *Proc. Int. Conf. Open Source Softw. Comput. (OSSCOM)*, 2015, pp. 1–6.

[4] J. C. Pry and R. K. Lomotey, "Energy consumption cost analysis of mobile data encryption and decryption," in *Proc. IEEE Int. Conf. Mobile Services (MS)*, Jun. 2016, pp. 178–181.

[5] T. Nie, L. Zhou, and Z.-M. Lu, "Power evaluation methods for data encryption algorithms," *IET Softw.*, vol. 8, no. 1, pp. 12–18, Feb. 2014.

[6] S. B. Sadkhan and A. O. Salman, "A survey on lightweight-cryptography status and future challenges," in *Proc. Int. Conf. Advance Sustain. Eng. Appl. (ICASEA)*, Mar. 2018, pp. 105–108.

[7] T. K. Goyal, V. Sahula, and D. Kumawat, "Energy efficient lightweight cryptography algorithms for IoT devices," *IETE J. Res.*, vol. 68, no. 3, pp. 1722–1735, pp. 2019.

[8] J. Daemen and V. Rijmen, *The Design of Rijndael: AES—The Advanced Encryption Standard*. Springer-Verlag, 2002, pp. 1–238.

[9] A. Biryukov and C. De Cannière, "Data encryption standard (DES)," in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg, Ed. Boston, MA, USA: Springer, 2005, pp. 129–135, doi: 10.1007/0-387-23483-7_94.

[10] D. Coppersmith, D. B. Johnson, and S. M. Matyas, "A proposed mode for triple-DES encryption," *IBM J. Res. Develop.*, vol. 40, no. 2, pp. 253–262, Mar. 1996.

[11] J. S. Murguía, M. M. Carlos, H. C. Rosu, and G. Flores-Eraña, "Improvement and analysis of a pseudo-random bit generator by means of cellular automata," *Int. J. Modern Phys. C*, vol. 21, no. 6, pp. 741–756, Jun. 2010.

[12] S. Roy, U. Rawat, and J. Karjee, "A lightweight cellular automata based encryption technique for IoT applications," *IEEE Access*, vol. 7, pp. 39782–39793, 2019.

[13] G. B. Ermentrout and L. Edelstein-Keshet, "Cellular automata approaches to biological modeling," *J. Theor. Biol.*, vol. 160, no. 1, pp. 97–133, Jan. 1993.

[14] D. G. Mallet and L. G. De Pillis, "A cellular automata model of tumor–immune system interactions," *J. Theor. Biol.*, vol. 239, no. 3, pp. 334–350, Apr. 2006.

[15] S. Maerivoet and B. De Moor, "Cellular automata models of road traffic," *Phys. Rep.*, vol. 419, no. 1, pp. 1–64, 2005.

[16] S. H. White, A. M. Del Rey, and G. R. Sanchez, "Modeling epidemics using cellular automata," *Appl. Math. Comput.*, vol. 186, no. 1, pp. 193–202, 2007.

[17] J. A. Parsons and M. A. Fonstad, "A cellular automata model of surface water flow," *Hydrological Processes*, vol. 21, no. 16, pp. 2189–2195, 2007.

[18] S. Roy, M. Shrivastava, C. V. Pandey, S. K. Nayak, and U. Rawat, "IEVCA: An efficient image encryption technique for IoT applications using 2-D von-Neumann cellular automata," *Multimedia Tools Appl.*, vol. 80, no. 21, pp. 31529–31567, 2021.

[19] S. Roy, M. Shrivastava, U. Rawat, C. V. Pandey, and S. K. Nayak, "IESCA: An efficient image encryption scheme using 2-D cellular automata," *J. Inf. Secur. Appl.*, vol. 61, Sep. 2021, Art. no. 102919.

[20] M. Seredynski and P. Bouvry, "Block encryption using reversible cellular automata," in *Proc. Int. Conf. Cellular Automata* Cham, Switzerland: Springer, 2004, pp. 785–792.

[21] J. Jin, "An image encryption based on elementary cellular automata," *Opt. Lasers Eng.*, vol. 50, no. 12, pp. 1836–1843, 2012.

[22] Y. Wang, Y. Zhao, Q. Zhou, and Z. Lin, "Image encryption using partitioned cellular automata," *Neurocomputing*, vol. 275, pp. 1318–1332, Jan. 2017.

[23] S. Roy, J. Karjee, U. S. Rawat, N. D. Pratik, and N. Dey, "Symmetric key encryption technique: A cellular automata based approach in wireless sensor networks," *Proc. Comput. Sci.*, vol. 78, pp. 408–414, Jan. 2016.

[24] M. Seredynski, K. Pienkosz, and P. Bouvry, "Reversible cellular automata based encryption," in *Proc. IFIP Int. Conf. Netw. Parallel Comput.* Cham, Switzerland: Springer, 2004, pp. 411–418.

[25] J. Urias, E. Ugalde, and G. Salazar, "A cryptosystem based on cellular automata," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 8, no. 4, pp. 819–822, Dec. 1998.

[26] M. Mejia-Carlos and J. Urias-Hermosillo, "An asymptotically perfect pseudorandom generator," *Discrete Continuous Dyn. Syst.*, vol. 7, no. 1, pp. 115–126, 2001.

[27] J. S. Murguía, G. Flores-Eraña, M. M. Carlos, and H. C. Rosu, "Matrix approach of an encryption system based on cellular automata and its numerical implementation," *Int. J. Modern Phys. C*, vol. 23, no. 11, Nov. 2012, Art. no. 1250078.

[28] J. Urias, G. Salazar, and E. Ugalde, "Synchronization of cellular automaton pairs," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 8, no. 4, pp. 814–818, Dec. 1998.

[29] M. T. Ramírez-Torres, J. S. Murguía, and M. M. Carlos, "Image encryption with an improved cryptosystem based on a matrix approach," *Int. J. Modern Phys. C*, vol. 25, no. 10, Oct. 2014, Art. no. 1450054.

[30] J. A. Aboytes-González, J. S. Murguía, M. Mejía-Carlos, H. González-Aguilar, and M. T. Ramírez-Torres, "Design of a strong S-box based on a matrix approach," *Nonlinear Dyn.*, vol. 94, no. 3, pp. 2003–2012, Nov. 2018.

[31] L. E. Reyes-López, J. S. Murguía, H. González-Aguilar, M. T. Ramírez-Torres, M. Mejía-Carlos, and J. O. Armijo-Correa, "Scaling analysis of an image encryption scheme based on chaotic dynamical systems," *Entropy*, vol. 23, no. 6, p. 672, May 2021.

[32] J. S. Murguía, M. Mejía Carlos, C. Vargas-Olmos, M. T. Ramìrez-Torres, and H. C. Rosu, "Wavelet multifractal detrended fluctuation analysis of encryption and decryption matrices," *Int. J. Modern Phys. C*, vol. 24, no. 9, Sep. 2013, Art. no. 1350069.

[33] S. Das, J. K. M. S. U. Zaman, and R. Ghosh, "Generation of AES S-boxes with various modulus and additive constant polynomials and testing their randomization," *Proc. Technol.*, vol. 10, pp. 957–962, 2013.

[34] P. Ramasamy, V. Ranganathan, S. Kadry, R. Damasevicius, and T. Blažauskas, "An image encryption scheme based on block scrambling, modified zigzag transformation and key generation using enhanced logistic—Tent map," *Entropy*, vol. 21, no. 7, p. 656, Jul. 2019.

[35] N. Ben Slimane, N. Aouf, K. Bouallegue, and M. Machhout, "A novel chaotic image cryptosystem based on DNA sequence operations and single neuron model," *Multimedia Tools Appl.*, vol. 77, no. 23, pp. 30993–31019, Dec. 2018.

[36] S. Askar, A. Karawia, A. Al-Khedhairi, and F. Al-Ammar, "An algorithm of image encryption using logistic and two-dimensional chaotic economic maps," *Entropy*, vol. 21, no. 1, p. 44, Jan. 2019.

[37] C. Vargas-Olmos, J. S. Murguía, M. T. Ramírez-Torres, M. M. Carlos, H. C. Rosu, and H. González-Aguilar, "Two-dimensional DFA scaling analysis applied to encrypted images," *Int. J. Modern Phys. C*, vol. 26, no. 8, Aug. 2015, Art. no. 1550093.

[38] G. Alvarez and L. I. Shujun, "Some basic cryptographic requirements for chaos-based cryptosystems," *Int. J. Bifurcation Chaos*, vol. 16, no. 8, pp. 2129–2151, Aug. 2006.

[39] C. Vargas-Olmos, J. S. Murguía, M. T. Ramírez-Torres, M. M. Carlos, H. C. Rosu, and H. González-Aguilar, "Perceptual security of encrypted images based on wavelet scaling analysis," *Phys. A, Stat. Mech. Appl.*, vol. 456, pp. 22–30, Aug. 2016.

[40] G.-F. Gu and W.-X. Zhou, "Detrended fluctuation analysis for fractals and multifractals in higher dimensions," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 74, no. 6, Dec. 2006, Art. no. 061104.

[41] J. Alvarez-Ramirez, E. Rodriguez, I. Cervantes, and J. Carlos Echeverria, "Scaling properties of image textures: A detrending fluctuation analysis approach," *Phys. A, Stat. Mech. Appl.*, vol. 361, no. 2, pp. 677–698, Mar. 2006.
[42] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 2366–2369.
[43] A.-V. Diaconu and A. C. Dascalescu, "Correlation distribution of adjacent pixels randomness test for image encryption," *Proc. Rom. Acad. Ser. A*, vol. 18, pp. 351–360, Jan. 2017.
[44] Y. Wu, J. P. Noonan, and S. Agaian, "NPCR and UACI randomness tests for image encryption," *Cyber J., Multidisciplinary J. Sci. Technol., J. Select. Areas Telecommu.*, vol. 1, pp. 31–38, Apr. 2011.
[45] A. Kassem, H. A. H. Hassan, Y. Harkouss, and R. Assaf, "Efficient neural chaotic generator for image encryption," *Digit. Signal Process.*, vol. 25, pp. 266–274, Feb. 2014.
[46] K. Tiri, D. D. Hwang, A. Hodjat, B.-C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede, "AES-based cryptographic and biometric security coprocessor IC in 0.18-μm CMOS resistant to side-channel power analysis attacks," in *Dig. Tech. Papers. Symp. VLSI Circuits*, 2005, pp. 216–219.
[47] F. J. Massey Jr., "The Kolmogorov–Smirnov test for goodness of fit," *J. Amer. Stat. Assoc.*, vol. 46, no. 253, pp. 68–78, Mar. 1951.
[48] L. Bassham, A. Rukhin, J. Soto, J. Nechvatal, M. Smid, S. Leigh, M. Levenson, M. Vangel, N. Heckert, and D. Banks, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Special Publication (NIST SP), Sep. 2010. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762
[49] P. Hamalainen, T. Alho, M. Hannikainen, and T. D. Hamalainen, "Design and implementation of low-area and low-power AES encryption hardware core," in *Proc. 9th EUROMICRO Conf. Digit. Syst. Design (DSD)*, 2006, pp. 577–583.
[50] S. Jamal, M. Khan, and T. Shah, "A watermarking technique with chaotic fractional S-box transformation," *Wireless Pers. Commun.*, vol. 90, no. 4, pp. 2033–2049, 2016.
[51] Ü. Cavusoglu, S. Kaçar, I. Pehlivan, and A. Zengin, "Secure image encryption algorithm design using a novel chaos based S-box," *Chaos, Solitons Fractals*, vol. 95, pp. 92–101, Feb. 2017.
[52] R. Sornalatha, N. Janakiraman, K. Balamurugan, A. K. Sivaraman, R. Vincent, and A. Muralidhar, "FPGA implementation of protected compact AES S-box using CQCG for embedded applications," *Smart Intell. Comput. Commun. Technol.*, vol. 38, pp. 396–401, Oct. 2021.
[53] T. Guneysu and H. Handschuh, *Cryptographic Hardware and Embedded Systems—CHES 2015*, vol. 9293. Saint-Malo, France: Springer, Sep. 2015.
[54] B. Rashidi, "Flexible structures of lightweight block ciphers PRESENT, Simon and LED," *IET Circuits, Devices Syst.*, vol. 14, no. 3, pp. 369–380, May 2020.
[55] R. Nithya and D. S. Kumar, "Where AES is for internet, Simon could be for IoT," *Proc. Technol.*, vol. 25, pp. 302–309, Jan. 2016.
[56] N. H. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. London, U.K.: Pearson, 2015.
[57] S. K. Mathew, F. Sheikh, M. Kounavis, S. Gueron, A. Agarwal, S. K. Hsu, H. Kaul, M. A. Anders, and R. K. Krishnamurthy, "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, no. 4, pp. 767–776, Apr. 2011.
[58] P. V. S. Shastry, A. Kulkarni, and M. S. Sutaone, "ASIC implementation of AES," in *Proc. Annu. IEEE India Conf. (INDICON)*, Dec. 2012, pp. 1255–1259.
[59] S. Morioka and A. Satoh, "A 10-Gbps full-AES crypto design with a twisted BDD S-box architecture," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 7, pp. 686–691, Jul. 2004.

J. D. J. DE LA ROSA-DE LA ROSA received the B.S. and M.S. degrees in engineering physics and applied sciences from Universidad Autónoma de San Luis Potosí, México, in 2018 and 2020, respectively, where he is currently pursuing the Ph.D. degree. In 2020, he was a Verification Engineer with Intel, where he was supporting basic debugging issues and functional validation testing. His research interests include the development of VLSI ICs, resistive crossbar-based memories, and ANNs.

J. S. MURGUÍA received the B.Sc., M.Sc., and Ph.D. degrees from the Universidad Autónoma de San Luis Potoí (UASLP), México, in 1998, 1999, and 2003, respectively. He is a Professor Researcher in electronics with the Sciences Faculty, UASLP. In 2010 and 2011, he made sabbatical stays as a Visiting Researcher with the Institute of Bio-Circuits, University of California San Diego, USA, and the Potosino Institute of Scientific and Technological Research, México, respectively. He has been involved in signal processing and scaling analysis research for the last 20 years, with regard to encryption systems and the analysis and implementation of dynamical chaotic systems. He is a member of the National System for Researchers (SNI-CONACyT-México) and a Regular Member of the Mexican Academy of Sciences. In 2018, he had participated in the management of the National Laboratory for Research, Instrumentation and Medical Imaging (CI3M), UASLP.

M. MEJÍA-CARLOS received the Ph.D. degree in applied sciences from Universidad Autónoma de San Luis Potosí, México, in 2001. She is currently a Professor with Universidad Autónoma de San Luis Potosí. Her research interests include cryptography, multimedia signal processing, and information security.

MIGUEL ÁNGEL LASTRAS-MONTAÑO received the B.S. and M.S. degrees in engineering physics and applied sciences from Universidad Autónoma de San Luis Potosí, México, and the Ph.D. degree in electrical and computer engineering from the University of California at Santa Barbara, in 2017. His research interests include the architectural development and organization of novel resistive crossbar-based memories and their use in 3-D-IC systems.