UNIVERSIDAD AUTONÓMA DE SAN LUIS POTOSÍ

FACULTAD DE CIENCIAS

UNIVERSIDAD AUTÓNOMA
DE SAN LUIS POTOSÍ

Application and implementation of an improved encryption system

Tesis para obtener el grado de doctor

Presenta:

M. en C. Marco Tulio Ramírez Torres

Asesores:

Dr. José Salomé Murguía Ibarra

Dra. Marcela Mejía Carlos

13 de julio 2015

Acknowledgements

I would first like to thank my entire family, for their support and specially to my father, my first math teacher. Thanks to my advisors, Dr. José Salomé Murguía I. and Dra. Marcela Mejía Carlos, for their guidance and trust through my entire research. All scientific production and this thesis could not be done without their support and guidance. Thanks to Dr. Alfonso Lastras, director of IICO. Thanks to all the people that work at IICO. Thanks to CONACYT (México) and UASLP for the support given during this research.

Abstract

In this work an improved version of the encryption system denominated ESCA is presented. This symmetric encryption system is based on the synchronization phenomena of cellular automata that uses the evolution local rule-90. In this new proposed version it has been established some preprocessing stages to have a flexible, reconfigurable and secure encryption system, where the numerical and experimental implementations have been carried out. The ESCA system was applied to encrypt different kind of signals, but in particular to images. In the implementations and applications, different platforms were considered: LabVIEW and a Virtex-5 FPGA. The main results obtained from this work are a compression-encryption scheme for real-time video, and an FPGA implementation of a reconfigurable image encryption system, based on the soft-processor MicroBlaze and the matrix approach of ESCA system. In addition, different tests and attacks were applied to the encrypted images, showing that the improved version of the ESCA system can be considered secure against statistical, cryptanalysis and differential cryptanalysis attacks. On the other hand, some multifractal characteristics of the density of ones of the main matrices of the ESCA system were determined using the scaling method known as the wavelet transform multifractal detrended fluctuation analysis.

Resumen

En este trabajo se presenta una versión mejorada del sistema de cifrado denominado ESCA. Este sistema de cifrado simétrico se basa en el fenómeno de sincronización de autómatas celulares, que utiliza la regla de evolución local 90. En la nueva versión propuesta se ha establecido algunas etapas de preprocesamiento para tener un sistema de encriptación flexible, reconfigurable y seguro, donde se han llevado a cabo las implementaciones numéricas y experimentales. El sistema ESCA se aplicó para cifrar diferentes tipo de señales, pero en particular a imágenes. En las implementaciones y aplicaciones se consideraron diferentes plataformas: LabVIEW y una FPGA Virtex-5. Los principales resultados obtenidos a partir de este trabajo es un esquema de compresión-encriptación para vídeo en tiempo real, y una implementación en FPGA de un sistema de encriptación de imagenes reconfigurable, basado en el procesador MicroBlaze y el enfoque de matricial de sistema ESCA. Además, las diferentes pruebas y ataques se aplicaron a las imágenes cifradas, que muestran que la versión mejorada del sistema ESCA puede ser considerado seguro contra ataques estadísticos, de criptoanálisis y criptoanálisis diferencial. Por otra parte, algunas características multifractales de la densidad de los de los principales matrices del sistema ESCA se determinaron utilizando el método de escalamiento conocido como WT-MFDFA (wavelet transform multifractal detrended fluctuation analysis).

Contents

List of Figures

1

Introduction

Cryptography is defined as the study of hidden writing, or the science of encrypting and decrypting messages. This science provides a set of different techniques, mechanisms, and tools aimed at protecting and authenticate communications and transactions. Since ancient times, the search of algorithms or procedures to protect different kind of information has been a complex issue. In the beginning, the cryptography was considered as an art, and it was not until the last century when cryptography became a science. The modern cryptography has an approach based on mathematical analysis, the developed algorithms try to satisfy certain definition of security. Nowadays, with the great advances in technology, there has been an increasing interest to have better and more efficient algorithms to maintain, as secure as possible, the different information that is related to us.

The data encryption allows to protect the confidentiality of information, this kind of algorithms make indistinguishable the data, avoiding from third parties without permission can see the information. The image encryption is an specialized area, because it requires different considerations. The classical algorithms like AES, DES and IDEA can not be

used in ECB mode for image encryption, therefore, there are numerous image encryption methods that have been proposed through the last few decades. For instance many encryption systems have been implemented with a chaotic approach [1, 2, 3, 7, 4]. Due of their chaotic properties such as ergodicity and sensitive dependence on initial conditions and on system parameters. To some of these [4, 5] have been found security weaknesses by applying the Chosen/Known plaintext attacks and the differential Chosen-plaintext attack, respectively in Ref. [7, 8].Therefore it is necessary that an extensive security analysis be applied to novel encryption systems before to be presented, there are many kinds of attacks that can be applied even when we can exclude brute-force attacks because the scheme has a large initial condition.

Some cryptography techniques are based on the application of cellular automata (CA); such cryptosystems have used CA in the encryption algorithm, as a pseudo-random generator, or as a combination of them[9, 10, 11, 12, 13, 14]. Numerous patents of methods of encryption data using CA, have been granted [15, 16]. The encryption system based on CA use to have the following characteristics: tamperproof, quick operation, flexibility to accept data of different sizes and no errors in the encryption and decryption [17].

On the other hand, the importance of implement cryptographic algorithms is to prove the security of the schemes in the real-word, and try to find new possible weaknesses not considered before. Besides, the hardware implementation of cryptographic algorithms allows to improve its efficiency using dedicated architectures. For this case the reconfigurable logic devices are a suitable solution to implement cryptographic algorithms for embedded systems or high speed applications. In these days, the capacity and features of FPGA devices have increased so that it is possible to implement on them an entire cryptographic system based on a soft processor core.

In this work we improve and implement the ESCA system, which is based on the synchronization phenomenon of CA using the rule-90 cellular automaton. However, the ESCA system presented some security weaknesses. In this direction, a modified and flexible matrix approach is presented here for improving its security. We will see that this matrix encryption proposal resist some common attacks, such as the Chosen-plaintext attack, and reducing the dimensions of the main matrices will help us to implement effectively both encryption and decryption processes. In other words, this alternative way will allow us to have a secure, flexible and reconfigurable encryption system that fits in the present digital technology. Despite this system has been implemented and considered in some applications, it has been rarely used to carry out image encryption operations, in this platform we performed a security analysis.

With this improved version, we developed two applications, a numerical implementation of a compression-encryption scheme, an appealing option in real-time video applications such as communications, surveillance among others. And the second one, an FPGA implementation of the ESCA system for image encryption, in a reconfigurable approach. The complete system includes a personal computer with a Host program that sends and receives the images, and an FPGA with the ESCA system programed (patent pending).

The thesis is organized as follows. Chapter 2 gives a general background. In Chapter 3 is described the encryption system ESCA, whereas Chapter 4 contains the implementation, analysis and applications of this cryptosystem. In Chapter 5 are discussed the conclusions and future work. In addition, there is an appendix about multifractal features of the main matrices of ESCA system.

2

Background

2.1 Cellular Automata

The concept of cellular automata (CA) was introduced in the 1940's by the mathematicians John von Neumann and Stanislaw Ulam [18]. CA are used to model complex behavior of natural systems, where local interactions are involved. In fact, CA represent a class of dynamical systems that enable to describe the evolution of complex systems with simple rules, without using differential equations. Besides, they can be easily implemented in hardware.

Cellular automata consist in an organized lattice of cells, where each cell has a finite number of states. The CA form a two dimensional lattice whose cells evolve in discrete steps, according to a local update rule applied uniformly over all the cells. At the beginning, a state is assigned to the cells at time $t = 0$, where the new states of a cell will depend on its own previous state and states of its neighborhood, as is shown in Fig. 2.1.

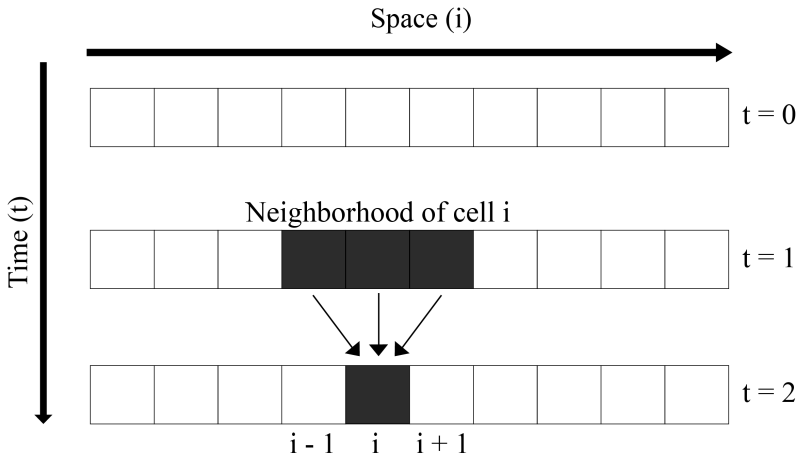

Figure 2.1: Space-time in 1-D CA.

Elementary cellular atomata (ECA) are 1-D CA with two states $\mathbb{Z}_2 = \{0, 1\}$ and radius-1 neighborhood. A CA local rule $\mathcal{A}$ is the algorithm used to compute the next cell state, in ECA can be referred as: $x_i^{t+1} = \mathcal{A}(x_{i-1}^t, x_i^t, x_{i+1}^t)$. The ECA differ from each other only in the choice of the local rule $\mathcal{A}$, they have three variables and each can get two state values, therefore there are eight posible combinations, resulting $2^8 = 256$ different local rules and ECA. For example, the local rule 90 is described by $x_i^{t+1} = \mathcal{A}(x_{i-1}^t + x_{i+1}^t) \bmod 2$. The name of the rule is obtained from the decimal equivalent of the binary resulting expressions of the eight combinations in the neighborhood, as we can see in the following lookup table:

Number	7	6	5	4	3	2	1	0
Neighborhood	111	110	101	100	011	010	001	000
Rule result	0	1	0	1	1	0	1	0

In 1984, Stephen Wolfram [19] proposed a heuristic classification of cellular automata that

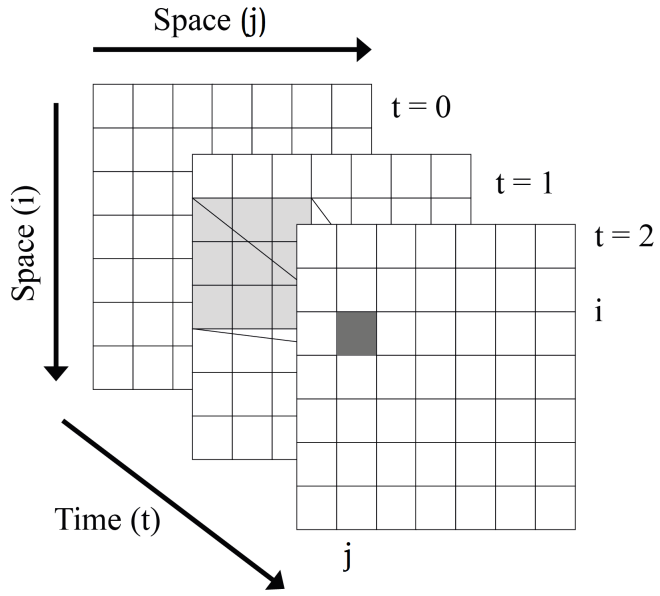

Figure 2.2: Space-time in a 2-D CA

comprises four classes of CA:

1. Class I – the CA evolution reaches a certain final state and stays there.

2. Class II – the CA encounters simple or cyclic structures.

3. Class III – the majority of initial states lead to arbitrary patterns.

4. Class IV – generates global complex structures.

This classification mainly refers to 1-D CAs, but similar ones can be made for 2-D or 3-D cases; an example of a 2-D CA is shown in Fig. 2.2. The very large phenomenology of the CA model offers a good basis for applications in cryptography due its apparently big complexity and massive parallelism.

2.2 Synchronization phenomenon

The synchronization phenomenon occurs when after a time period the behaviors of two dynamical systems get arbitrarily close. In the case of CA, after a certain number of steps at time t the difference between the two state vectors $\underline{x}^t$, $\underline{y}^t$ corresponding to the driver and the replica cellular automaton, eventually equals the null vector $\mathbf{0} = (0, 0, ..., 0)$. For this is necessary that, in every step both vectors should evolve using the same local rule. In Ref. [20] has been shown that a pair of linear elementary CA, where the local rule $\mathcal{A}$ corresponds to the automaton rule 90, synchronizes if every pair of consecutive coupled coordinates are separated by a block of $N = 2^n - 1$ sites uncoupled. For cellular automata, the synchronization exists as the result of a non-trivial coupling. Cellular automata coupling occurs when a given set of coordinates (coupled coordinates) are copied from one of the systems which is the driver cellular automaton, to the system that will be the replica cellular automaton. In this way, at each time step, both systems evolve according to the same local rule and using the same coupled coordinates. As an example of synchronization phenomenon, the Fig. 2.3 shows a case considering $n = 3$, therefore the coupled coordinates are seperated by $N = 2^3 - 1 = 7$ sites. In the same Fig. we can see the evolution according to the automaton rule 90 of driver and replica CA with the same the coupled coordinates. After $2^3 - 1 = 7$ steps, the evolution of both cellular automaton is the same.

2.3 Two-dimensional wavelet transform

The first mention of wavelets appeared in an appendix to the thesis of the mathematician Alfred Haar in 1909 [21]. However, the concept of the wavelet was proposed by the

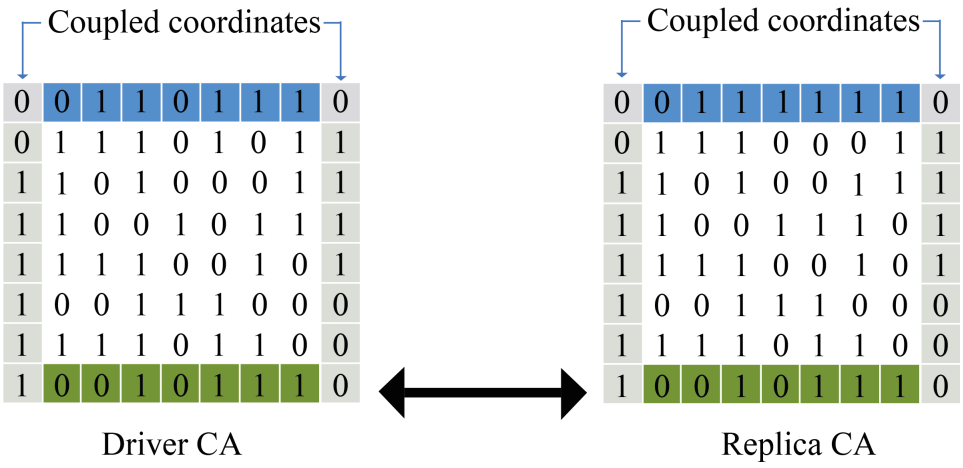

Figure 2.3: Synchronization phenomenon. Left: driver CA. Right: replica CA.

geophysicist Jean Morlet and the physicist Alex Grossman in 1984. The Haar transform is known as a compact, dyadic, orthonormal wavelet transform, and for many years was the only orthogonal wavelet known. In 1985, the mathematician Yves Meyer constructed the second orthogonal wavelet called Meyer wavelet. The popularity of this area increased in the following years, more and more scholars joined in this field, achieving the first international conference in France in 1987.

The discrete wavelet transform has a huge number of applications in different areas, but many signals have a bi-dimensional nature, like images. For this case, the wavelet transform has also a discrete version to process them. The wavelet transform in two dimensions considers a two-dimensional scaling function, $\Phi(x, y)$, and three two-dimensional wavelets, $\Psi^H(x, y)$, $\Psi^V(x, y)$ and $\Psi^D(x, y)$ where the superscript index indicates the information of the signal at the horizontal (H), vertical (V), and diagonal (D) directions. Each function corresponds to the product of a one-dimensional scaling function ϕ and a corresponding wavelet ψ such that the product does not produce a one-

dimensional result, i. e.,

$$\Phi(x,y) = \phi(x)\phi(y), \tag{2.1}$$

$$\Psi^H(x,y) = \psi(x)\phi(y), \tag{2.2}$$

$$\Psi^V(x,y) = \phi(x)\psi(y), \tag{2.3}$$

$$\Psi^D(x,y) = \psi(x)\psi(y). \tag{2.4}$$

With these functions, we define the scale and translation versions as

$$\Phi_{j,m,n}(x,y) = 2^{j/2}\Phi(2^j x - m, 2^j y - n), \tag{2.5}$$

$$\Psi^d_{j,m,n}(x,y) = 2^{j/2}\Psi^d(2^j x - m, 2^j y - n) \tag{2.6}$$

where $j, m, n \epsilon \mathbb{Z}$ and the superscript index d assumes the values H, V and D to identify the directional wavelets given in (2.2)-(2.4). In the same spirit as in the case of the DWT in one dimension, and considering that (2.5)-(2.6) constitute an orthonormal basis for $L^2(\mathbb{R}^2)$, the expansion of a function $f(x,y)$ of finite energy is then

$$f(x,y) = \frac{1}{\sqrt{MN}}\sum_m\sum_n \mathbf{a}_{j0;m,n}\Phi_{j0;m,n}(x,y) + \frac{1}{\sqrt{MN}}\sum_{d=H,V,D}\sum_m\sum_n \mathbf{d}^d_{j;m,n}\Psi^d_{j;m,n}(x,y), \tag{2.7}$$

where the scaling $\mathbf{a}_{j;m,n}$ and wavelet $\mathbf{d}^d_{j;m,n}$ coefficients are defined as

$$\mathbf{a}_{j;m,n} = \iint f(x,y), \Phi_{j;m,n}(x,y)dxdy; \tag{2.8}$$

$$\mathbf{d}^d_{j;m,n} = \iint f(x,y), \Psi^d_{j;m,n}(x,y)dxdy. \tag{2.9}$$

Equation (2.7) represents the synthesis equation, whereas (2.8) and (2.9) are the analysis equations. Both equations constitute the two-dimensional discrete wavelet transform (2D-DWT). From now on, unless otherwise stated, we refer a two-dimensional function or signal $f(x,y)$ as an image function $I(x,y)$, since the 2D-DWT is generally used to image analysis. To compute numerically the two-dimensional wavelet transform, we follow the Mallat's algorithm for two-dimensional functions [22]. With this algorithm, the multiresolution decomposition of a two-dimensional function or an image is represented by a series of approximations and details of sub-images, which become increasingly coarse. In general, a 2D-DWT can be considered as a separable filter bank of row and column directions that decomposes one resolution level of an image into four sub-images. One stage of this procedure is shown in 2.4, where h and g correspond to a lowpass and highpass filter, respectively, and they are followed for the operation of downsampling by two. We can observe that after applying the first wavelet level transform, we have four sub-images, and if the original image function $I(x,y)$ has dimensions $M \times N$, then each sub-image have $M/2$ rows and $N/2$ columns. The approximation sub-image is obtained by computing approximations along rows of the signal $I(x,y)$ followed by computing approximations along columns. This sub-image is an averaged version of the image $I(x,y)$ with half resolution and with statistical properties that are similar to those of the original signal $I(x,y)$. In the same way, in the horizontal sub-image, we first

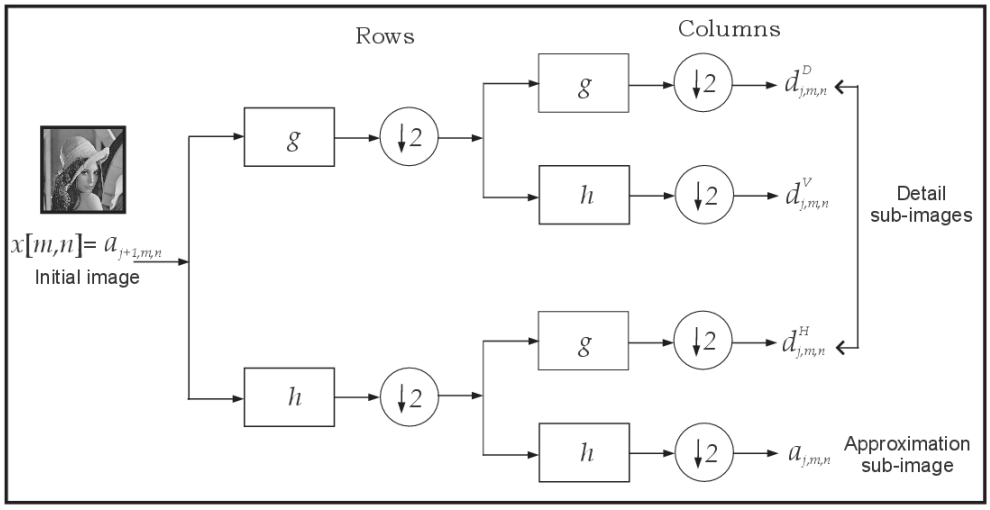

Figure 2.4: One stage in a multiresolution image decomposition.

compute the approximations along the rows of the image $I(x, y)$ followed by computing the details along the columns. As a result, the horizontal edges of $I(x, y)$ will be always detected by the details along the columns. Since this subimage analyzes the horizontal information, it is clear why it is denoted as the horizontal sub-image. In the multiresolution decomposition the same wavelet transformation is applied but only to the approximation subimage obtaining again four sub-images, but now with dimensions of $M/2^k$ rows and $N/2^k$columns, where $k = 1, \ldots, \min(\log_2(M), \log_2(N))$ is the wavelet level. In Fig. 2.5 is shown an example of this numerical calculation, using the two-dimensional Haar wavelet transform. This transform is considered in this thesis, because its algorithm is memory efficient and reversible.

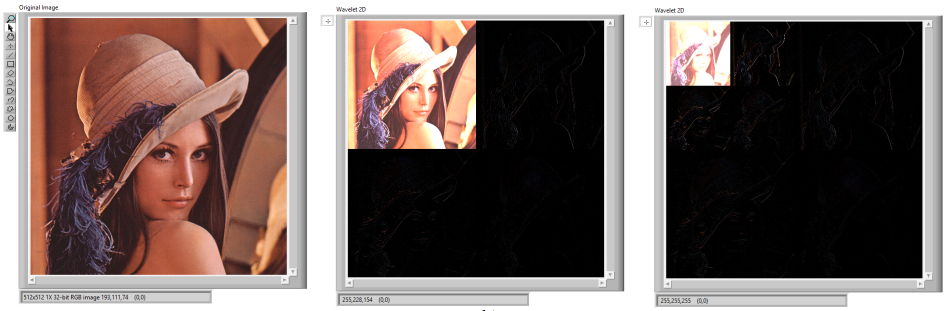

Figure 2.5: Multilevel wavelet decomposition. a) Original Lena image, b) First level Decomposition of Lena Image, c) Second level Decomposition of Lena Image.

2.4 Cryptography

Cryptography is the art and science of keeping information secure from unintended audiences. The study of the procedures, processes and methods of making and using secret writing, as codes or ciphers. The security of information is an issue as old as the writing, therefore it has been adapted to different communication media (paper, telegraph, telephone and computer networks) through the time.

The cryptographic algorithms are split into two main branches: symmetric (or private-key) algorithms, and asymmetric (or public-key) algorithms. In the symmetric algorithms, the encryption key K is the same as the decryption key, as is shown in Fig. 2.6. Hence, the sender and receiver should keep private the secret key K with the aim to prevent a third party may decrypt the messages. These algorithms have some advantages as: speed, strength and availability. The disadvantages of the symmetric algorithms are: distribution and management of keys, and scalability [23]. The algorithms AES, DES and IDEA are examples of symmetric ciphers.

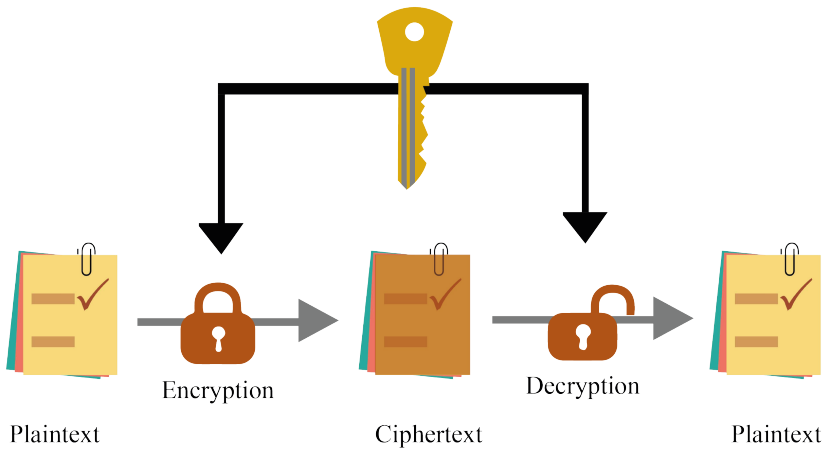

Figure 2.6: Symmetric cryptography algorithm.

Otherwise, the asymmetric algorithms use two different keys, a public key K_E and a private key K_D, see Fig. 2.7. In this case, the encryption key K_E cannot decrypt the ciphertext. The main advantages of asymmetric algorithms are: key management and distribution, scalability, access control and authentication. On the other hand, the disadvantages of the asymmetric ones are: computationally intensive and slow. Some examples of these algorithms are RSA, Elliptic Curve Cryptography and ElGamal.

The modern cryptography is based on three cores:

- Formal definitions,
- assumptions,
- proofs.

The formal definitions using mathematical models, specify what security means and

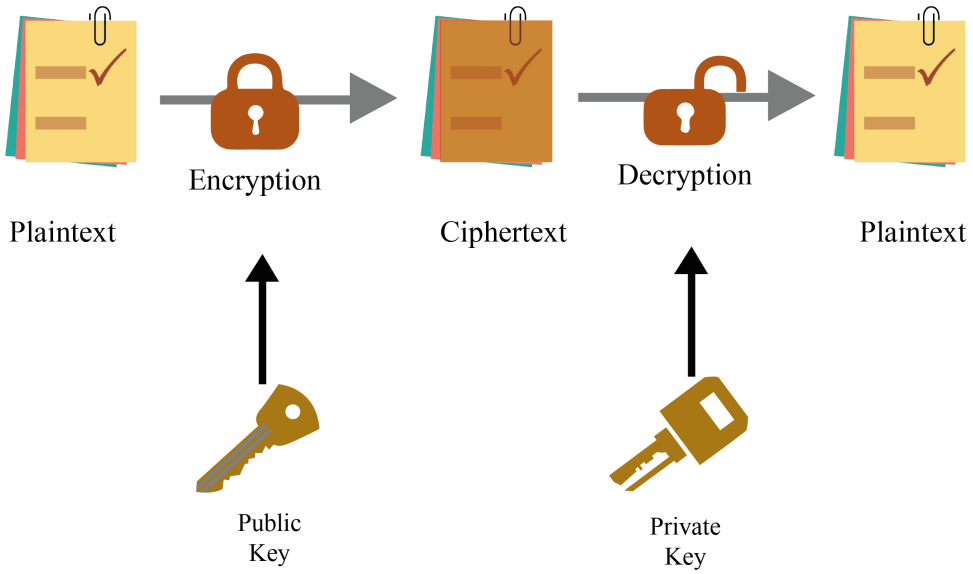

Figure 2.7: Asymmetric cryptography algorithm.

the objectives to achieve. In this way, in a scheme design we can discern what is important and what is not, and also the formal definitions allow others to understand the security guarantees provided by some scheme. The cryptography requires computational assumptions to enable proofs of security. The assumptions allow meaningful comparison between schemes based on different assumptions. The assumptions should be clear and unambiguous, so that the researchers can attempt to validate them. The proofs simulate a scenario where an attacker try to break a scheme. The proofs give an overview of the security provided by a scheme, relative to specified definitions and assumptions.

In the real world, secure schemes have been broken due two possible reasons:

- the definition does not correspond to reality
- the assumption is invalid

Therefore, the cryptography works in a continuing search for definitions and proofs that allow to developers and users be sure that the real world environment where some scheme will be implemented, fits with the definition.

3

ESCA system

The encryption based on the synchronization phenomenon of cellular automata (named ESCA for short) is a symmetric key algorithm that uses the rule-90 cellular automaton [42]. In Ref. [42] the ESCA system is considered as a class of block cryptosystems that comprises two indexed families of permutations and an asymptotically perfect pseudo-random number generator (PRNG). Urías *et al.* in Ref.[20] described the synchronization phenomenon of cellular automata (CA), by pointing out that a pair of coupled CA with local rule 90 can synchronize if every pair of consecutive coordinates is separated by a block of $2^k - 1$ uncoupled sites, for k a positive integer.

Moreover, in order to have a simple implementation, Murguía *et al*.[25] have used a matrix approach to implement all of the components of the ESCA system. With the aim to make the cryptosystem secure against cryptanalysis attacks, in Ref. [26] the ESCA system was modified, as we will observe.

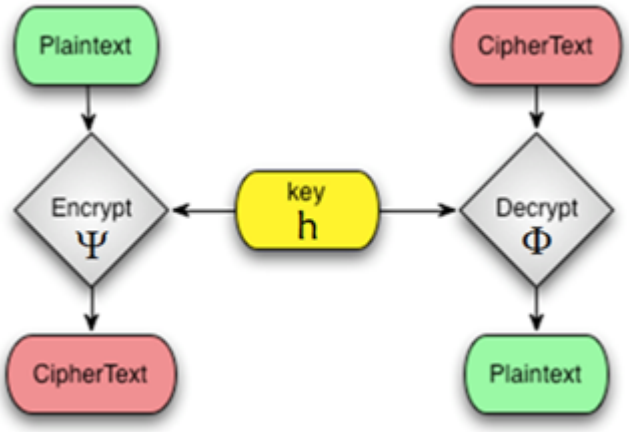

Figure 3.1: The main components of the ESCA system: the indexed families of permutations, Ψ and Φ, and the pseudo-random generator of keys.

3.1 The encryption sytem model

The ESCA system is a cryptosystem that comprises the sets M, C and K of binary words of length N, i.e. $\mathbb{Z}_2^N$, where $\mathbb{Z}_2 = \{0, 1\}$, and two indexed families of permutations, $\Psi = \{\psi_{\mathbf{k}} : \mathbf{k} \,\epsilon\, K\}$ and $\Phi = \{\phi_{\mathbf{k}} : \mathbf{k} \,\epsilon\, K\}$. The words in M and C are called the plaintexts and ciphertexts, respectively, whereas the words in the set of indices K are the enciphering keys. In addition, the functions $\psi_{\mathbf{k}} : M \to C$, and $\phi_{\mathbf{k}} : C \to M$, are called the encryption and decryption functions, respectively. Basically, the cryptosystem transforms a plaintext sequence $\mathbf{m}$ to a ciphertext sequence $\mathbf{c}$, i.e. for every $\mathbf{k} \,\epsilon\, K$ one has $\mathbf{c} = \psi_{\mathbf{k}}(\mathbf{m})$, whereas to disclose from the sequence of cipher-blocks, one uses the decryption functions $\phi_{\mathbf{k}}$, i.e. $\mathbf{m} = \phi_{\mathbf{k}}(\psi_{\mathbf{k}}(\mathbf{m}))$. Since the complete encryption scheme is private, the encryption and decryption processes use the same enciphering key $\mathbf{k}$. In Fig. 3.1 is illustrated in a general way the ESCA system.

3.2 The Basic Unit Cipher

A general implementation and definition of the main elements of the ESCA system is possible with the help of the basic unit cipher (BUC) [42]. A space-pattern of the BUC is shown in Fig. 3.2, which is defined as the $N \times N$ square pattern in the lattice that consists of N time-running words $(x_1^0, x_1^1, ..., x_1^{N-1}), ..., (x_N^0, x_N^1, ..., x_N^{N-1})$. The space-pattern of the BUC is initially calculated with the infinite sequence

$$x^0 = (..., x_{-1}^0, x_0^0, x_1^0, ..., x_{N-1}^0, x_N^0, ...), \tag{3.1}$$

that evolves according to the local rule $\mathcal{A}_L(x_{i-1}^t, x_i^t, x_{i+1}^t) = [x_{i-1}^t + x_{i+1}^t] \bmod 2$, from $t = 0$ to $t = N = 2^n - 1$, where $i \neq 0$ and $i \neq N + 1$ since x_0^t and x_{N+1}^t are externally assigned at each time t. This local rule corresponds to the rule 90 of a CA. In the space-pattern of the BUC, five main words are identified $\mathbf{x}$, $\mathbf{y}$, $\mathbf{m}$, $\mathbf{c}$ and $\mathbf{k} = (\mathbf{x}, \mathbf{y})$. The word $\mathbf{c} = (x_{N+1}^1, x_{N+1}^2, ..., x_{N+1}^N)$, located on the right side of the BUC, is a cipherblock word. It is obtained using the indexed family permutation $\Psi_{\mathbf{k}}(\mathbf{m})$, with input words $\mathbf{k}$ and $\mathbf{m}$; this permutation determines the cipherblocks when the cellular automata is iterated backward in time, i.e., $x_{i+1}^t = (x_i^{t+1}, x_{i-1}^t) \bmod 2$. On the other hand, the permutation Φ allows us to bring the word $\mathbf{c}$ back to the plaintext sequence $\mathbf{m} = (x_2^N, x_3^N, ..., x_N^N)$, that is, $\mathbf{m} = \phi_{\mathbf{k}}(\mathbf{c})$. This word is located at the bottom of the BUC. This inverse permutation, with input words $\mathbf{c}$ and $\mathbf{k}$, is calculated when the cellular automaton is made to run forward in time, i.e., $x_i^{t+1} = (x_{i-1}^t, x_{i+1}^t) \bmod 2$. Finally, the word located at the top of the BUC, $\mathbf{k} = h(\mathbf{x}, \mathbf{y}) = (x_2^0, x_3^0, ..., x_N^0)$ is obtained using the function $h(\mathbf{x}, \mathbf{y})$, where the automaton is also iterated backward in time, with the initial seeds $\mathbf{x} = (x_0^0, x_0^1, ..., x_0^N)$ and $\mathbf{y} = (x_1^0, x_1^1, ..., x_1^{N+1})$ which are located on the left side of the BUC.

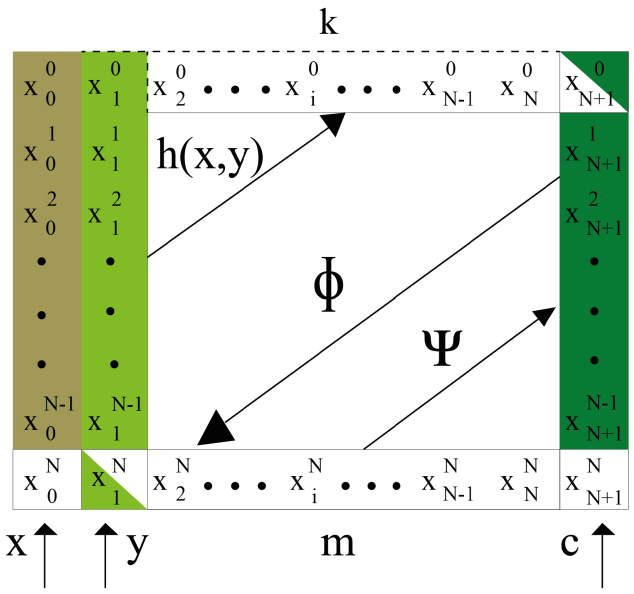

Figure 3.2: Primitives defined by the basic unit cipher. The functions h and Ψ are determined by iterating the cellular automata backward in time, whereas the function Φ is computed by running the cellular automata forward in time.

3.3 Pseudo-Random Sequences Generator

To implement numerically the pseudo-random number generator (PRNG) in its basic form, it is followed the algorithm shown in Fig. 3.3(a). At first, the key generator requires two seeds, $\mathbf{x} = \{x_1, x_2, x_3, ..., x_N\}$ and $\mathbf{y} = \{y_1, y_2, y_3, ..., y_{N+1}\}$, and the first number of N bits generated with these seeds is the sequence output of function h, $\mathbf{k} = \{k_1, k_2, k_3, ..., k_N\}$. Now, this sequence is feeding back to the input, and it becomes the next value of $\mathbf{x}$, whereas the previous value of $\mathbf{x}$ becomes the initial bits of the new $\mathbf{y}$. The missing bit corresponds to the least significant bit (LSB) of the previous $\mathbf{y}$, which becomes now the most significant bit (MSB) of this sequence, and the same procedure is iterated repeatedly. However, with the aim to attain an asymptotically unpredictable

generator under a random search attack, a generating scheme consisting of three coupled transformations h is considered. This proposal is shown in Fig. 3.3(b) [27], and it is explained briefly. Inside the new generator two copies of the basic h transformation are iterated autonomously from their initial words generating two sequences of N bits, $\{p_k\}_{k\geq 0}$ and $\{q_k\}_{k\geq 0}$. The third copy, called the x-map, is iterated in a slightly different manner, the function h in the x-map is driven by the autonomous p-map and q-map according to $x_k = h(p_k, q_k)$. Since the sequences p_k and q_k have a length of N bits each and the required inputs of the transformation h must be one of N bits, and the other of $(N + 1)$ bits, the missing bit is obtained by applying an addition modulo 2 operation between the two respective LSB's that become the MSB's of their respective previous inputs of the maps.

To evaluate the randomness of the generated sequences by the previous approaches, in Ref. [28] used a statistical package denominated the NIST suite to evaluate the generated sequences for $N = 7$, $N = 15$ and $N = 31$ bits, where one and three transformations were considered. In the analysis was observed that the generated pseudo-random sequences with three coupled transformations passes all tests except for $N = 7$, whereas for one transformation the PRNG fails for $N = 7$ and $N = 15$, but it is uniformly distributed.

3.4 The ESCA system with a matrix approach

In previous works [25, 28], the authors were able to implement the ESCA cryptosystem with a matrix approach. In fact, they implemented the family of permutations and the PRNG with some basic matrix operations or transformations on the main sequence matrix $\mathbf{Q}_N$ as we will see.

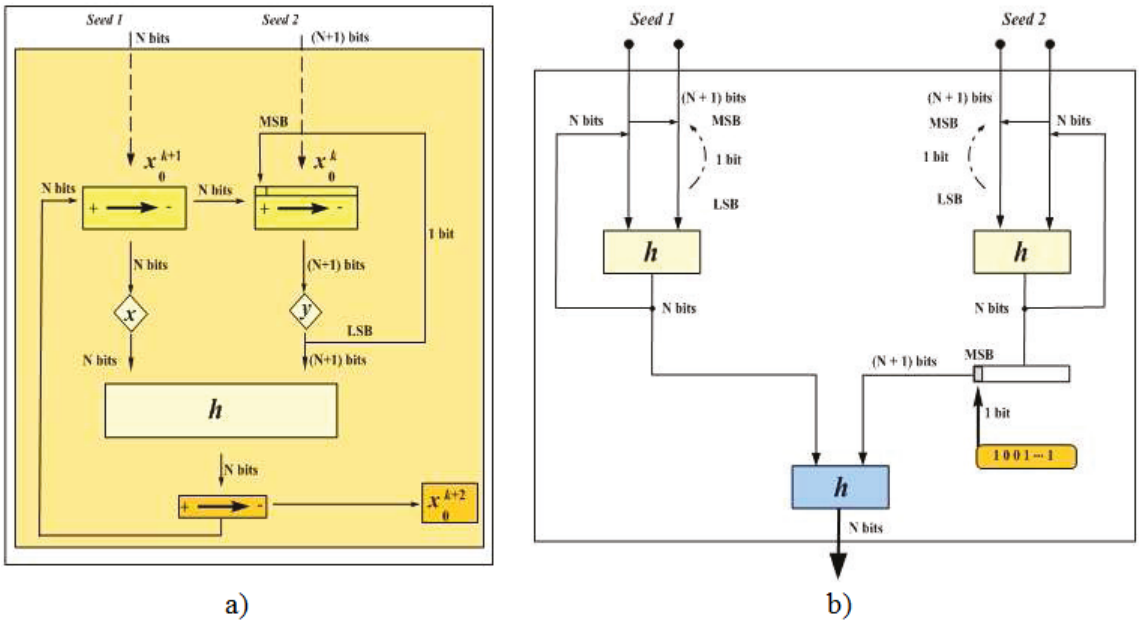

Figure 3.3: (a) Basic form of the pseudo-random number generator. (b) Modified generator consisting of three coupled transformations. Where MSB and LSB correspond to the most significant bit and the least significant bit, respectively.

3.4.1 The indexed family of permutations Ψ

The encryption process can be represented in terms of matrices as

$$\mathbf{c} = \Psi_{\mathbf{k}}(\mathbf{m}) = (\mathbf{P}_N \times \mathbf{x}) + (\mathbf{Q}_N \times \mathbf{m}) \bmod 2, \tag{3.2}$$

where the sequences $\mathbf{c}$, $\mathbf{m}$ and $\mathbf{x}$ have dimensions $N \times 1$, and the two matrices $\mathbf{P}_N$ and $\mathbf{Q}_N$, are square matrices of order N, with $N = 2^n - 1$, for $n = 1, 2, 3....$ The matrix $\mathbf{P}_N$ is an upper triangular that is initially generated from the vector $p = [p_1, p_2, ..., p_N]$, which constitutes the first row, and the components with position index $i = (2^n+1)-2^{j+1}$,

$j = 0, 1, 2, ...(n-1)$, have a value of 1, and 0 otherwise. The next $N-1$ rows are generated by applying a right shift of one position of the previous row with a zero as its first value. Likewise, $\mathbf{Q}_N$ is a lower triangular matrix that can be generated initially from the vector $\mathbf{q} = [q_1, 0, 0..., 0]$, where the component q_1 has a value of 1, and N is the number of bits. Hence, $\mathbf{q}$ is a vector with N components, and it constitutes the first row of the matrix $\mathbf{Q}_N$. The $N-1$ rows are generated by applying the CA rule 90 of the previous row with fixed boundary conditions of zero to the left and right sides. For instance, considering $N = 7$, the triangular matrices $\mathbf{P}_7$ and $\mathbf{Q}_7$ have the form

$$\mathbf{P}_7 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{Q}_7 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{3.3}$$

3.4.2 The indexed family of permutations Φ

In this case the matrix implementation of the inverse permutation has a similar structure of (3.2), that is,

$$\mathbf{m} = \phi_{\mathbf{k}}(\mathbf{c}) = [(\mathbf{R}_N \times \mathbf{x}) + (\mathbf{T}_N \times \mathbf{c})] \bmod 2, \tag{3.4}$$

where the sequences and matrices have the same dimensions as the encryption process.

From Eq. (3.2) and comparing with (3.4), we have that $\mathbf{R}_N = (-\mathbf{Q}_N^{-1}\mathbf{P}_N) \bmod 2$, whereas the $\mathbf{T}_N$ matrix is just the inverse of $\mathbf{Q}_N$, i.e. $\mathbf{T}_N = \mathbf{Q}_N^{-1} \bmod 2$. As an example, and considering again $N = 7$, we have the matrices $\mathbf{R}_7$ and $\mathbf{T}_7$ as

$$\mathbf{R}_7 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{T}_7 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{3.5}$$

3.4.3 The pseudo-random generator

Murguía and collaborators [25, 28] introduced two similar matrix approaches to compute feasibly the pseudo-random sequences of N bits. In both cases, we computed a square sequence matrix $\mathbf{H}_N$ of order $(2N + 1)$. In this thesis, we only describe the format of the matrix $\mathbf{H}_N$ of Ref. [28]. The square matrix $\mathbf{H}_N$ can be partitioned as

$$\mathbf{H}_N = \left(\frac{\mathbf{H}_{Nt}}{\mathbf{H}_{Nb}} \right), \tag{3.6}$$

where the matrices $\mathbf{H}_{Nt}$ and $\mathbf{H}_{Nb}$ constitute the top and bottom parts of $\mathbf{H}_N$. The matrix $\mathbf{H}_{Nt}$ has dimensions of $N \times (2N + 1)$ elements, and it is generated initially from two row vectors, $\mathbf{v} = [v_1, 0, ..., 0, v_{N+2}, ..., 0]$ and, $\mathbf{w} = [0, w_2, 0, ..., w_{N+1}, 0, w_{N+3}, ..., 0]$ where the components v_1, v_{N+2}, w_2, w_{N+1} and w_{N+3} have a value of 1, and N is the number of bits. The vectors $\mathbf{v}$ and $\mathbf{w}$ constitute the two first rows of the matrix $\mathbf{H}_{Nt}$ and the $(N - 2)$

rows are generated by applying an addition modulo 2 operation of the two previous rows with the elements of the previous row shifted to the right by one position. On the other hand, the matrix $\mathbf{H}_{Nb}$, of dimensions $(N+1)\times(2N+1)$, is the identity matrix in the first $(N+1)$ columns and zeros in the rest. For instance, for $N=7$ we have that the top and bottom matrices are

$$\mathbf{H}_{7t} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.7}$$

$$\mathbf{H}_{7b} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{3.8}$$

then the matrix $\mathbf{H}_7$ is

$$\mathbf{H}_7 = \left(\frac{\mathbf{H}_{7t}}{\mathbf{H}_{7b}}\right) = \left(\begin{array}{cccccccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right). \tag{3.9}$$

Basically, with the $\mathbf{H}_{Nt}$ matrix we are able to compute the pseudo-random key sequence, whereas with $\mathbf{H}_{Nb}$ the feedback sequence, as was explained in section 3.3. Therefore, once the number N of bits of sequences are selected we can generate the pseudo-random sequences of N bits with the help of the matrix $\mathbf{H}_N$ using the expression:

$$\mathbf{U}_{k+1} = \mathbf{H}_N \mathbf{U}_k,\ k = 1, 2, ..., \tag{3.10}$$

where $\mathbf{U}_k = [\mathbf{x}_k\, \mathbf{y}_k]^T$corresponds to the first inputs of the PRNG, and $\mathbf{U}_{k+1}$ is composed by the next inputs of the PRNG; note that $\mathbf{U}_{k+1}$ is formed by the generated pseudo-random key $\mathbf{x}_{k+1}$ and the feedback sequence $\mathbf{y}_{k+1}$.

We have used the wavelet transform multifractal detrended fluctuation analysis (WT-MFDFA) as scaling method, to reveal the multifractal features of some matrices that carry out the main functions in ESCA system.

3.5 Improvement of the ESCA system

With the aim to have a secure and flexible cryptosystem, in this Section we propose a modified version of the ESCA system. In fact, we use the structure of the PRNG to attain the ESCA system secure under different model threats.

At first, we transform the plaintext before to encrypt it into an unintelligible form with the help of the top matrix $\mathbf{H}_{Nt}$ of the PRNG, see (3.6). Basically, we require the plaintext sequence $\mathbf{m} = \{m_1, m_2, m_3, ..., m_N\}$, of N bits, and a random binary sequence $z = \{z_1, z_2, z_3, ..., z_{N+1}\}$, of $N + 1$ bits, considering $N = 2^n$ for $n = 1, 2, ...$ Next, the modified plaintext sequence is computed as

$$\widehat{\mathbf{m}} = \mathbf{H}_{Nt} \begin{pmatrix} \mathbf{m} \\ \mathbf{z} \end{pmatrix} \bmod 2, \tag{3.11}$$

where $\widehat{\mathbf{m}} = \{\widehat{m}_1, \widehat{m}_2, \widehat{m}_3..., \widehat{m}_N\}$ is a sequence of N bits. To compute the following modified plaintext by means of (3.11), we require again two sequences, a new plaintext sequence, and a binary random sequence. At this time, the latter sequence comprises the $\widehat{\mathbf{m}}$ sequence, which becomes the initial bits of the new $\mathbf{z}$, and the first bit of the previous $\mathbf{z}$ becomes the last bit of this sequence. For the following sequences, the same procedure is iterated repeatedly as we can see in the Fig. 3.4. This process, including the feedback, can be expressed as:

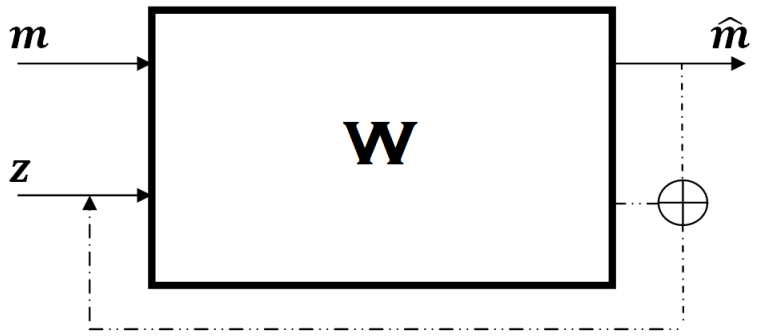

Figure 3.4: Basic form of process function to obtain $\widehat{\mathbf{m}}$.

$$\begin{pmatrix} \widehat{\mathbf{m}}_{k+1} \\ \mathbf{z}_{k+1} \end{pmatrix} = \mathbf{W} \begin{pmatrix} \mathbf{m}_k \\ \mathbf{z}_k \end{pmatrix} \bmod 2, \quad \text{with } \mathbf{W} = \begin{pmatrix} \mathbf{H}_{Nt} \\ \mathbf{H}_{Nt} \\ \mathbf{b} \end{pmatrix}, \text{ and } k = 1, 2, ..., \tag{3.12}$$

where the feedback is computed by the second half $\mathbf{W}$, which is composed by the matrix $\mathbf{H}_{Nt}$, and the row vector $\mathbf{b}$ of $(2N + 1)$ elements with a value of 1 in its position $N + 1$ and 0 otherwise, i.e. $\mathbf{b} = [0, 0, 0, ..., 1, 0, ...0]$.

If we consider the action of Eq. (3.12) in the ESCA system, any encrypted block does not reveal any bits of the secret key $\mathbf{k}$ for any plaintext, even when $\mathbf{m} = \mathbf{0} = \{0, 0, ..., 0\}$. In fact, with this implementation, the ESCA system is resistant to some cryptanalysis attacks like Chosen/Known-Plaintext attacks (CPA and KPA), since it does not encrypt directly the plaintext. This will be illustrated in advance, when the ESCA system is applied to images.

On the other hand, to recover the original plaintext sequences we use the expression

$$\mathbf{m} = \mathbf{M}_N \begin{pmatrix} \widehat{\mathbf{m}} \\ \mathbf{z} \end{pmatrix} \bmod 2, \tag{3.13}$$

where the matrix $\mathbf{M}_N$ has dimensions of $N \times (2N+1)$, and is formed by two matrices, $\mathbf{Q}_N$ and $\mathbf{D}_N$, which constitute the left and right parts of $\mathbf{M}_N$, respectively, i.e. $\mathbf{M}_N = (\mathbf{Q}_N \mid \mathbf{D}_N)$. The matrix $\mathbf{D}_N$ has dimensions of $N \times (N+1)$ elements, and it is a kind of band matrix, where the nonzero elements of $\mathbf{D}_N$ are lied in some different diagonal lines around the main diagonal. In particular, $\mathbf{D}_N$ is a matrix with 1's on the first superdiagonal line, and on the k_a-th subdiagonals, with $k_a = 2^a - 1$, for $a = 1, 2, ..., n$. As an example, we have the following matrix for $N = 8$ where $n = 3$.

$$\mathbf{M}_8 = (\mathbf{Q}_8 \mid \mathbf{D}_8) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{3.14}$$

The matrix approach of the inverse process to recover the plaintext from the processed version is defined by:

$$\begin{pmatrix} \mathbf{m}_{k+1} \\ \mathbf{z}_{k+1} \end{pmatrix} = \mathbf{V} \begin{pmatrix} \widehat{\mathbf{m}}_k \\ \mathbf{z}_k \end{pmatrix} \bmod 2 \quad \text{with } \mathbf{V} = \begin{pmatrix} \mathbf{M}_N \\ \mathbf{H}_{Nb} \\ \mathbf{b} \end{pmatrix}, \text{ and } k = 1, 2, ... \tag{3.15}$$

Besides, the ESCA system is designed to encrypt plaintexts or to decrypt ciphertexts of length $N = 2^n - 1$, for $n = 1, 2, ...$ This bit length is not adequate in most nowadays applications. If we want to encrypt sequences of bit length $J = 2^j$, for $j = 1, 2, ...$, we have to choose the value of $n > j$ such that $N = 2^n - 1$ must be larger than J. With this constraint, we can ensure that the ESCA system is perfectly secrecy, and considering the parameters N and J for the dimensions of the matrices there is no necessity of padding the plaintext sequences. In this way, the number of operations to encrypt/decrypt a block is reduced to less than half, because the original matrices (3.3) and (3.5) are partitioned. Hence, to compute feasibly the encryption process for a bit length of powers of two, the ciphertext sequence can be calculated using the following equation

$$\mathbf{c} = \Psi_{\mathbf{k}} = [(\widehat{\mathbf{P}} \times \mathbf{x}) + (\widehat{\mathbf{Q}} \times \widehat{\mathbf{m}})] \bmod 2, \tag{3.16}$$

where the sequences $\mathbf{c}$ and $\widehat{\mathbf{m}}$ have dimensions $J \times 1$, with $J = 2^j$, for $j = 1, 2, 3, ...$ and the sequence $\mathbf{x}$ has dimensions $N \times 1$, with $N = 2^n - 1$, for $n = 1, 2, 3, ...$ The matrices $\widehat{\mathbf{P}}$, of dimensions $J \times N$, and $\widehat{\mathbf{Q}}$, of order J, conform partitioned matrices of different dimensions of the matrices $\mathbf{P}_N$ and $\mathbf{Q}_N$, respectively. The latter relationship is illustrated in the following equation:

$$\mathbf{P}_N = \left(\begin{array}{c} \widehat{\mathbf{P}} \\ \hline \cdots \end{array} \right), \quad \mathbf{Q}_N = \begin{pmatrix} \widehat{\mathbf{Q}} & \cdots \\ \cdots & \cdots \end{pmatrix}. \tag{3.17}$$

For instance, for $J = 4$ and $N = 7$, and considering Eq. (3.16), we have that

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} \widehat{m}_1 \\ \widehat{m}_2 \\ \widehat{m}_3 \\ \widehat{m}_4 \end{pmatrix} \bmod 2. \tag{3.18}$$

To recover the processed plaintext, the implementation of the respective matrix implementation has a similar structure of (3.16), i.e..

$$\widehat{\mathbf{m}} = \phi_k(\mathbf{c}) = [(\widehat{\mathbf{R}} \times \mathbf{x}) \times (\widehat{\mathbf{T}} \times \widehat{\mathbf{m}})] \bmod 2, \tag{3.19}$$

where the sequences $\mathbf{c}$, $\widehat{\mathbf{m}}$ and $\mathbf{x}$ have the same dimensions as in (3.16). In this case, the matrices $\widehat{\mathbf{R}}$ of dimensions $J \times N$, and $\widehat{\mathbf{T}}$ of order J, correspond to the partitioned matrices of $\mathbf{R}_N$, and $\mathbf{T}_N$, respectively. Considering again $J = 4$ and $N = 7$, with Eq. (3.19), we have that

$$\begin{pmatrix} \widehat{m}_1 \\ \widehat{m}_2 \\ \widehat{m}_3 \\ \widehat{m}_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} \bmod 2. \tag{3.20}$$

The complete, modified and improved encryption system is shown in Fig. 3.5, where we use matrices of small dimensions during both encryption and decryption processes, but the sequence key $\mathbf{k}$ maintains its selected length N.

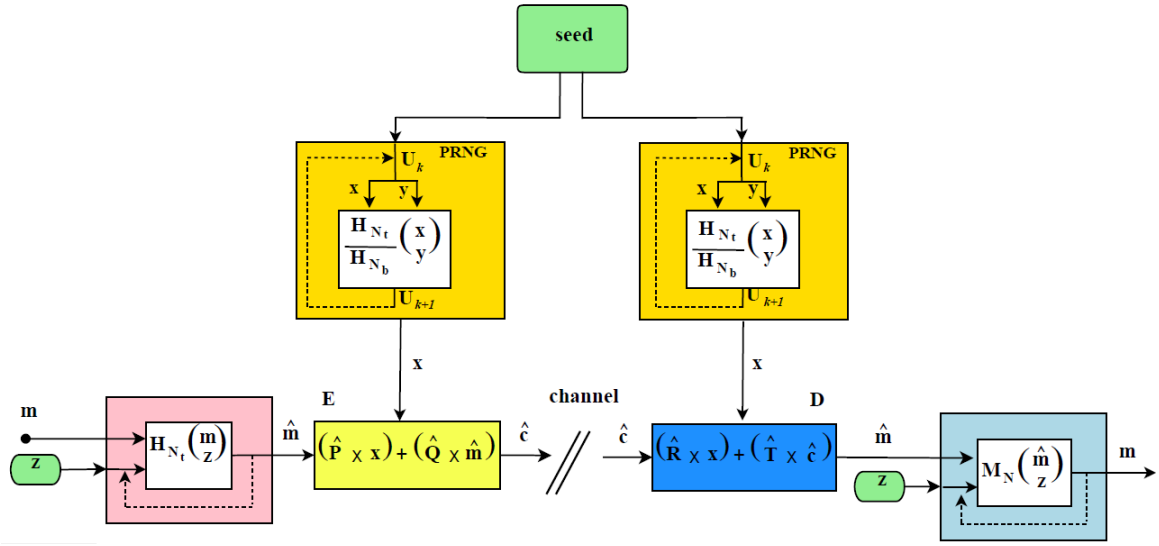

Figure 3.5: The actual ESCA system with a matrix approach.

4

Implementation, analysis and application of the ESCA system

In this chapter is described how the encryption system ESCA is implemented, improved and validated for image encryption [26, 29, 30]. In addition, we will discuss two main applications achieved on this thesis with the ESCA system: a compression-encryption scheme for real-time video and an FPGA implementation of a reconfigurable image encryption [29]. In the hardware implementation is used the matrix approach of the ESCA system [25] and the scheme for real-time video uses the one time equations [31].

4.1 Joint compression and encryption scheme for digital images

With the aim to have an application for real-time secure video transmissions, we implement an encryption-compression scheme for digital images. At first, we used the ESCA system for image encryption, and we carried out a security analysis of the

encryption system. Despite that the enhanced version of the ESCA system exhibited good encryption results and security, the latter system also showed a high latency, thus this processing time is not viable to real-time video transmissions. To overcome this issue we introduced a compression stage, where an energy criterion based on the two-dimensional Haar wavelet transform is considered. This compression scheme has become a useful and flexible procedure to compress different kind of information.

The combination of the encryption and compression procedures presents a system with a good performance to encrypt information with a low latency. In addition, this version can be reconfigurable, thus we have a system that could be an appealing option in real time applications such as video communications, among others. In this order, we present the steps of this implementation. In section 4.1.1 is described the algorithm for the image encryption, whereas in section 4.1.2 is evaluated the encryption system. The wavelet compression scheme is presented in section 4.1.3, and finally the encryption-compression scheme is described in section 4.1.4.

4.1.1 Image encryption

The digital image encryption is an area of interest, because there are digital visual data stored on different media and exchanged over different networks nowadays. This task is required in many applications, such as pay-TV, medical imaging systems, military image communications, surveillance and confidential video conferences. Classic encryption algorithms like Data Encryption Standard (DES), Advanced Encryption Standard (AES), International Data Encryption Algorithm (IDEA)[32] among others present a poor performance in ECB mode for image encryption, due to some of their intrinsic features such as high redundancy, bulk data capacity, high correlation among pixels and others[33,

34].

The ESCA system is proposed to encrypt RGB and grayscale images. The encryption process takes an image as input and gives a ciphered-image whose size is the same as that of input image as output. The procedure to carry out the complete encryption to images proceeds as follows.

1. Load the plain-image $\mathbf{I}$ of size $N \times M$.

2. By scanning the image $\mathbf{I}$ row by row, arrange its respective pixels as a sequence or a vector, and convert each pixel value to their corresponding binary value.

3. Establish the length of the block of bits to encrypt, $J = 2^j$, for $j = 1, 2, \ldots$.

4. Provide the value of n such that $N = 2^n - 1$ must be larger than the length of J bits, and calculate the initial seeds of size N and $(N+1)$ bits to generate the pseudo random sequences with the corresponding matrix to the PRNG.

5. Compute the modified plaintext sequence using (3.11), considering the appropriate size of the matrix $\mathbf{H}_{Nt}$.

6. Calculate the matrices $\widehat{\mathbf{P}}$ and $\widehat{\mathbf{Q}}$ to encrypt the data by means of (3.2).

7. By reshaping the set of ciphered sequences of the previous step into an $N \times M$ image, obtain the ciphered-image $\mathbf{I}_c$.

The decryption process is similar to the encryption scheme.

1. Load the encrypted-image $\mathbf{I}_c$ of size $N \times M$.

2. By scanning the image $\mathbf{I}_c$ row by row, arrange its respective pixels as a sequence or a vector, and convert each pixel value to their corresponding binary value.

3. Establish the length of the block of bits to decrypt, $J = 2^j$, for $j = 1, 2, ...$

4. Provide the value of n such that $N = 2^n - 1$ used to encrypt the image, and with the same initial seeds of size N and $(N+1)$ bits, generate the pseudo random sequences with the corresponding matrix to the PRNG.

5. Calculate the matrices $\widehat{\mathbf{R}}$ and $\widehat{\mathbf{T}}$ to decrypt the data by means of (3.4).

6. To obtain the plain-text sequence, the modified plain-text sequence is processed with the inverse fuction considering the appropriate size of the matrix $\mathbf{M}_N$.

7. By reshaping the set of ciphered sequences of the previous step into an $N \times M$ image, obtain again the plain-image $\mathbf{I}$.

4.1.2 Security analysis

To measure the quality and the robustness of the ESCA scheme, it has been evaluated with different common tests. This is done by testing the distribution of pixels of the ciphered-images, studying the correlation between the plain and ciphered images, the correlation among the adjacent pixels in the ciphered-image, the information entropy of the encrypted images, the key sensitivity analysis, the chosen-plain image attack and for the differential cryptanalysis we calculate the NPCR and the UACI parameters. We will describe them in a general way.

4.1.2.1 Histogram

An image histogram shows how pixels in an image are distributed by plotting the number of pixels at each color intensity level. If the histogram of an encrypted image (or ciphered-image) has a uniform distribution, then the cipher is able to hide the redundancy of original image. We calculate the histograms for three different 256 gray-level images, the Lena, peppers and mandrill images and also for their color version. All of them have dimensions of 512×512, and we consider these images because they are widely used as standard test images in the field of image processing. The histograms for the grayscale Lena test image (plain-image) and its encrypted version are shown in Fig. 4.1, whereas in Fig. 4.2 are illustrated only the histograms for each color intensity level of the color Lena test image with their respective encrypted versions. From the figures, one can see that the histograms of the ciphered-images are uniformly distributed and significantly different from the respective histograms of the plain-images.

4.1.2.2 Correlation between original and encrypted images

To show that the ciphered-image is independent from the plain-image, we calculate the correlation coefficient between both images. If the coefficient is close to 0, it suggests that there is no linear correlation or a weak linear correlation. The correlation coefficients for the three grayscale test images are 0.0007, 0.0006 and 0.0009, respectively, and the results for the color test images are listed in Table 4.1. The term C_{PC} is the correlation coefficient obtained from the P channel image of the plain-image and the C channel image of the ciphered-image. We can observe that the coefficients are close to zero, showing that the plain-images are independent from the respective ciphered versions.

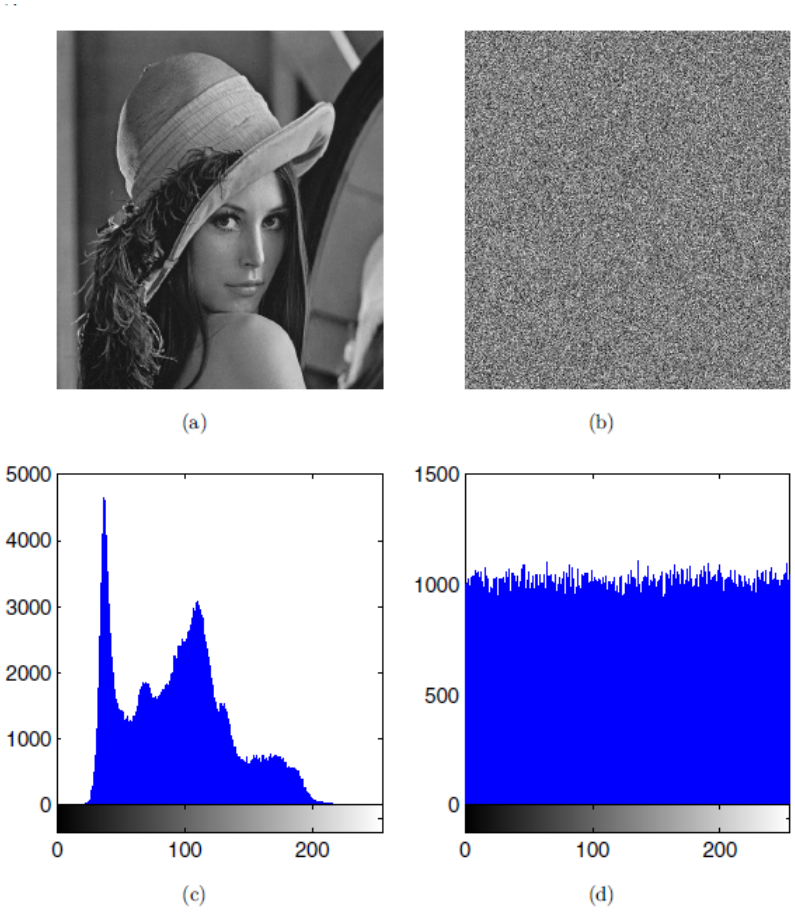

Figure 4.1: Histogram analysis for the gray-scale Lena test image. Top row: (a) The plain-image and (b) the ciphered-image. Bottom row: (c) The histogram of (a) and (d) the histogram of (b).

4.1.2.3 Correlation analysis of adjacent pixels

In addition, we have also analyzed the correlation between two adjacent pixels at the horizontal, vertical and diagonal directions of the plain-images and their corresponding encrypted versions. At first, we randomly select 3000 pairs of adjacent pixels in each

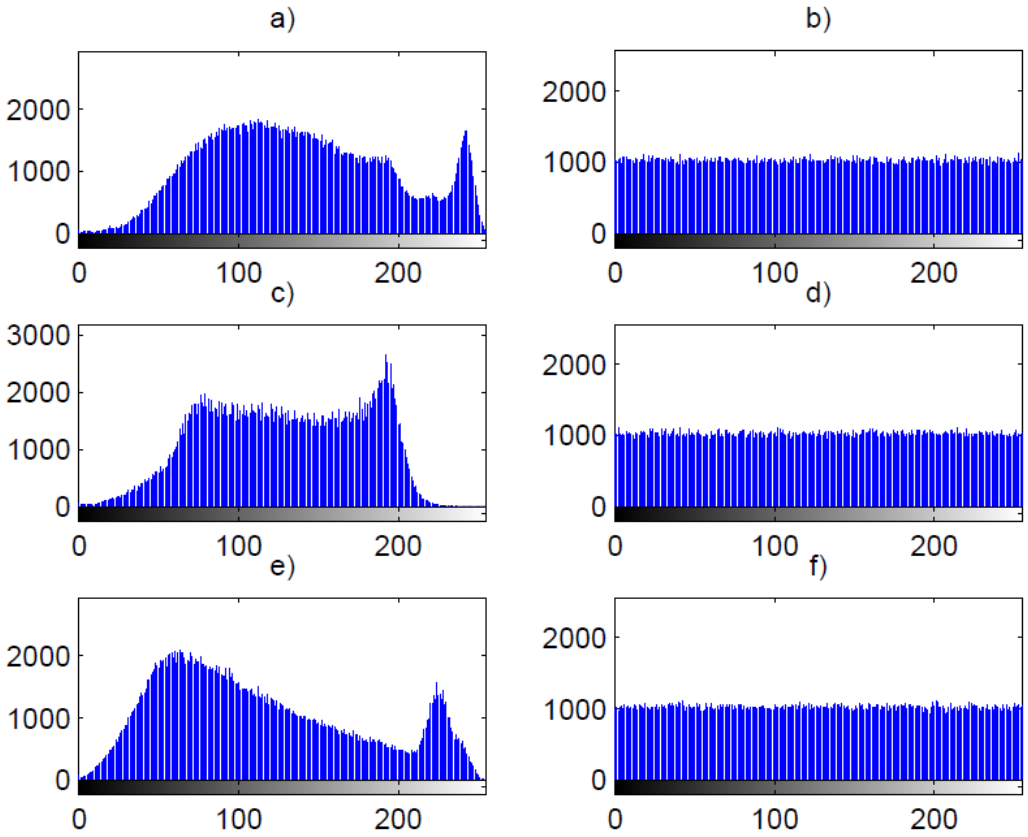

Figure 4.2: (Color online) Histogram analysis for a color Lena test image. Left column: Histograms for each color channel of the plain-image. Right column: The respective histograms of the encrypted cases for each color channel of the plain-image.

direction from the plain-images and their ciphered-images. Then, in each case, we have computed the correlation coefficient of each pair. In Fig. 4.3 the gray level distribution of adjacent pixels at the horizontal, vertical and diagonal directions in the Lena image (top row) and its ciphered version (bottom row) is shown. The results of the rest of gray plain-images and their respective ciphered versions are given in Table 4.1.2.3. Since the correlation coefficients of the plain-images are near to 1, it is clear that each pixel of these images is highly correlated with its adjacent pixel either in horizontal, vertical or diagonal

Table 4.1: Correlation coefficient between plain-images and their corresponding ciphered-images.

Image	C_{RR}	C_{RG}	C_{RB}	C_{GR}	C_{GG}	C_{GB}	C_{BR}	C_{BG}	C_{BB}
Lena	0.0014	-0.0032	-0.0049	-0.0024	-0.0047	-0.0018	-0.0042	0.0020	-0.0011
Peppers	0.0007	-0.0014	-0.0034	-0.0030	-0.0020	0.0035	-0.0023	0.0054	-0.0020
Mandrill	0.0003	0.0005	0.0005	-0.0042	-0.0021	0.0003	-0.0033	0.0005	-0.0033

direction. On the other hand, the correlation coefficients of the cipher-images are close to 0, hence there is no correlation among the pixels when the proposed encryption scheme is used. Similar results for the color images and their encrypted versions were obtained, see Table 4.3 for the case of the color Lena image.

Table 4.2: Correlation coefficients of two adjacent pixels in gray-scale images.

	Correlations coefficients in					
	Plain-image			Ciphered-image		
Direction	Lena	Mandrill	Peppers	Lena	Mandrill	Peppers
Horizontal	0.9662	0.8873	0.9691	0.0345	-0.0094	-0.0097
Vertical	0.9737	0.7877	0.9787	-0.0038	0.0159	0.0316
Diagonal	0.9538	0.7575	0.9583	0.0273	0.0017	-0.0005

4.1.2.4 Information entropy

The information entropy can be used to measure randomness in the texture of an image. This expresses the degree of uncertainties in the system, and it is defined as

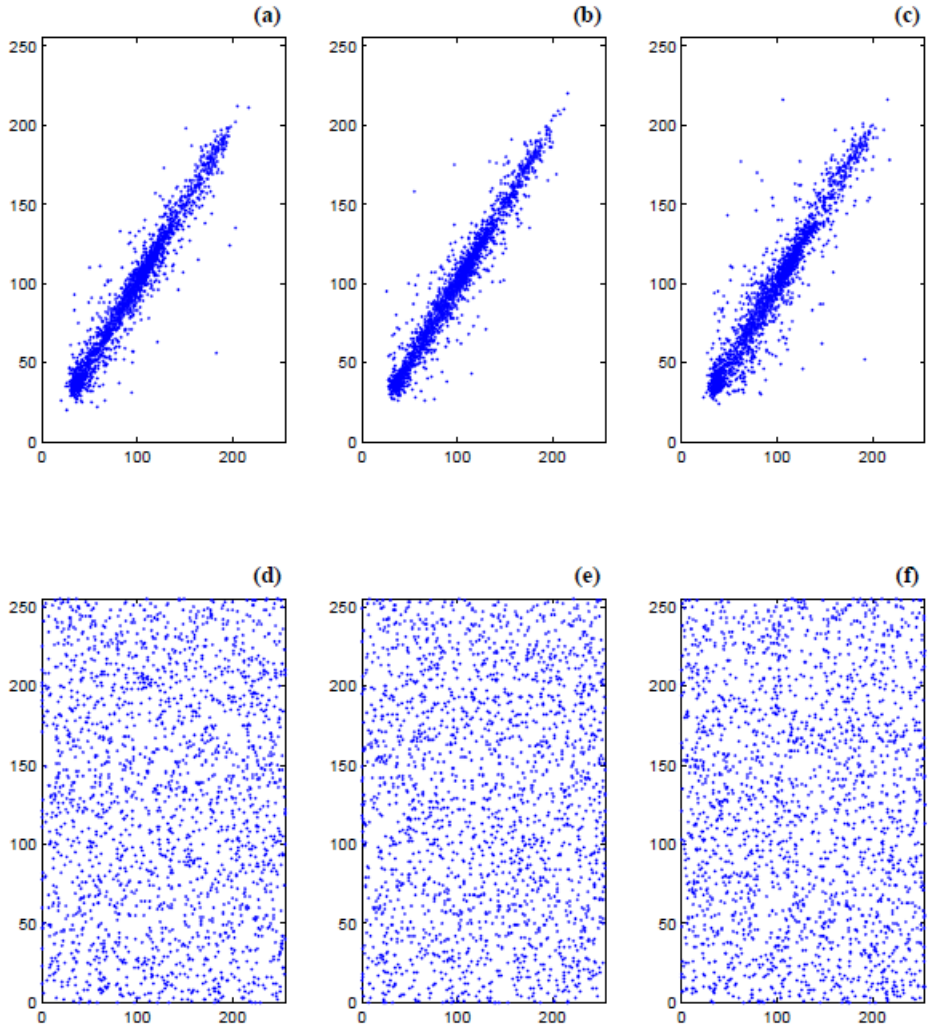

Figure 4.3: Correlation plot of two adjacent pixels for a gray Lena test image (top) and its encrypted version (bottom) at the horizontal (first column), vertical (second column) and diagonal (third column) direction.

$$H(m) = -\sum_{i=0}^{2^N-1} P(m_i)\log_2 P(m_i) \tag{4.1}$$

where N is the number of bits to represent a symbol m_i and $P(m_i)$ represents the

Table 4.3: Correlation coefficients of two adjacent pixels in the RGB Lena image.

	Correlations coefficients in					
	Plain-image			Ciphered-image		
Direction	Red	Green	Blue	Red	Green	Blue
Horizontal	0.9730	0.9595	0.9730	-0.0001	-0.0003	-0.0017
Vertical	0.9764	0.9693	0.9764	0.0021	-0.0009	-0.0007
Diagonal	0.9575	0.9459	0.9575	-0.0005	0.0023	-0.0023

probability of symbol m_i, hence the entropy is expressed in bits. For a truly random source emitting 2^N symbols, the entropy is $H(m) = N$, therefore, for a ciphered-image with 256 gray levels, the entropy should ideally be $H(m) = 8$. For the proposed image encryption scheme, the information entropy is $H(m) = 7.999$, which is very close to the ideal value. This means a high diffusion is achieved by the proposed algorithm. The results obtained for all test images are in Table 4.4.

Table 4.4: Entropy information of the original and the ciphered images

Images	Original image entropy	Encrypted image entropy
Lena	7.47671588418625	7.99925248383686
Mandrill	7.76243605368237	7.99932406628112
Peppers	7.66982551417632	7.99934896282480

4.1.2.5 Key sensitivity analysis

A good cipher should hide the information so well, that even flipping one bit of the secret key the encryption process would be totally different. This feature makes the encryption system more effective against brute-force attacks. To evaluate the key sensitivity feature

of the ESCA system, we first encrypt the color Lena plain-image using the test key $k = 44653600$. In Fig. 4.4(a) the resultant ciphered-image is shown. With a one-bit change in the original key, we used three different decryption keys to decrypt the ciphered-image. In Figs. 4.4(b)–4.4(d) the resultant decrypted images for the keys $k_1 = 44653501$, $k_2 = 44657600$ and $k_3 = 04653600$ are shown respectively, and we can observe that there are no patterns or visible information from the plain-image. For completeness, the correlation coefficient between the ciphered-image and the different decrypted images are also calculated. The results are shown in Table 4.5, where the resultant correlation coefficients are close to zero, showing that there is no correlation between the ciphered-image with key k and the respective decrypted images with keys k_1, k_2 and k_3.

Table 4.5: Correlation coefficients of the images of Fig. 4.4.

Images	C_{RR}	C_{GG}	C_{BB}
Figure (a) and figure (b)	-0.0007	0.0004	-0.0002
Figure (a) and figure (c)	9.28E-5	0.0012	-0.0011
Figure (a) and figure (d)	-0.0012	0.0010	0.0010

4.1.2.6 Chosen-plain image attack

As is pointed out in Ref. [35], if a cryptosystem is secure against the Chosen-plain image attack, it is also secure against other cryptanalytic attacks such as Cipher image-only attack or Known-plain image attack. Before the improved version of the encryption system described in Section 5 of Chapter 3, the ESCA system exhibited a weakness under the model attacks: Chosen-plain image attack and Known-plain image attack (CPIA, KPIA). The enhanced version of the ESCA system is resistant to these cryptanalysis attacks and it can encrypt multimedia content. To illustrate the insecurity of the system, we are going to

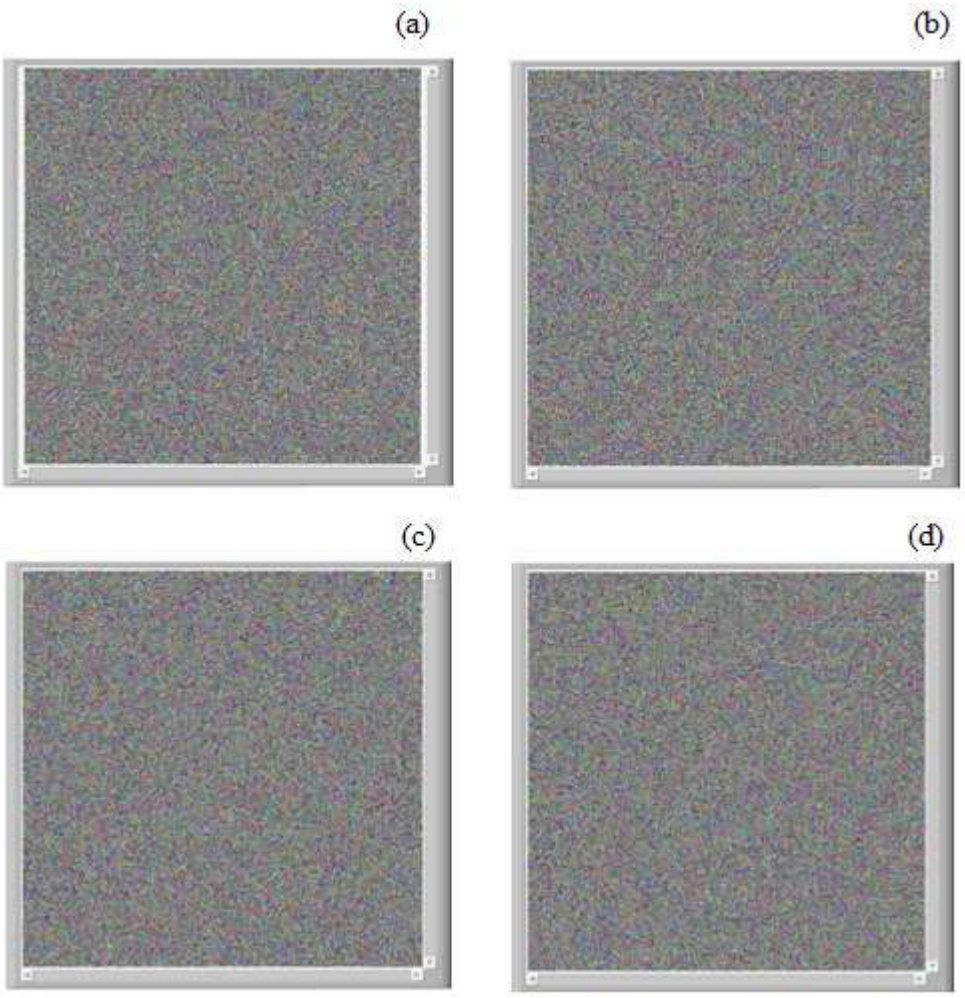

Figure 4.4: (a) Encrypted color Lena image using the key $k = 44653600$. The corresponding decrypted images using the different keys (b) $k_1 = 44653501$, (c) $k_2 = 44657600$ and (d) $k_3 = 04653600$.

show briefly the image encryption process and the CPIA, with the ESCA system without the improvement.

The scenario of CPIA consists in that the attackers can choose some plaintexts and get the

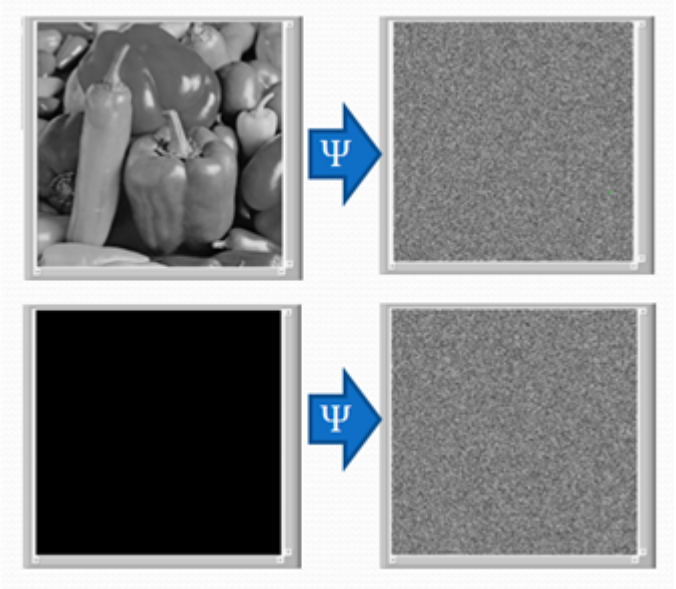

Figure 4.5: On the first column are the Chosen-plain images, on the second column their encrypted versions

corresponding ciphertexts, all under the same conditions of encryption. In the Fig. 4.5 are shown two plain-images and their respective encrypted versions. The next step consists in make an XOR operation between the second image (solid image) and its encrypted version, the result of this operation is called mask (F_m), see Fig. 4.6. Finally, we make an XOR operation between the encrypted image of the peppers and F_m, if the resulting image reveals information of the original image (peppers image) the system is considered insecure, see Fig. 4.7.

If we want to improve the ESCA system, we need to understand why the system is vulnerable. For this, we are going to review in detail the encryption process and the operations of CPA based on Fig. 4.5. In this scenario, we focus in the encryption of one pixel $\mathbf{m}$ of the first image (peppers image), and the result is called $\mathbf{c}_m$. With same secret

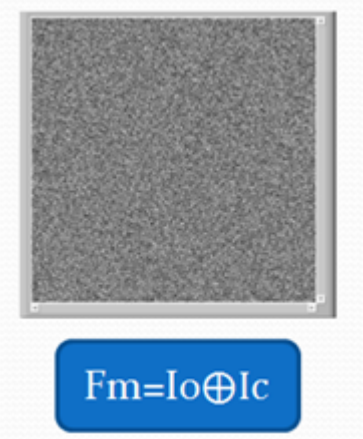

Figure 4.6: Mask F_m

Figure 4.7: Recovered image by CPA

key $\mathbf{k}$, we encrypt a pixel of the second chosen image (all the coefficients are $\mathbf{m} = 0$), this encrypted pixel is denoted by $\mathbf{c}_0$, the encryption operations are

$$\mathbf{c}_m = \mathbf{P}_N\mathbf{x}_1 \oplus \mathbf{Q}_N\mathbf{m}, \text{ and } \mathbf{c}_0 = \mathbf{P}_N\mathbf{x}_1 \oplus \mathbf{Q}_N 0. \tag{4.2}$$

The computation of the mask F_m for every pixel is obtained making an XOR operation

between the pixel of the solid image $\mathbf{m} = 0$ and its encrypted version $\mathbf{c}_0$:

$$F_{m_0} = \mathbf{c}_0 \oplus 0 = \mathbf{c}_0. \tag{4.3}$$

The last operation consists of the XOR between the encrypted pixel $\mathbf{c}_m$ of the first image and the corresponding pixel of the mask F_{m_0}.

$$(\mathbf{P}_N\mathbf{x}_1 \oplus \mathbf{Q}_N\mathbf{m}) \oplus (\mathbf{P}_N\mathbf{x}_1 \oplus \mathbf{Q}_N 0) = \mathbf{Q}_N\mathbf{m}. \tag{4.4}$$

The previous result is due to the facts that $\mathbf{P}_N\mathbf{x}_1 \oplus \mathbf{P}_N\mathbf{x}_1 = 0$ and $\mathbf{Q}_N 0 = 0$.

It can be easily noticed that under this attack the key encryption is nullified and the only protection offered by the ESCA system depends on the operation between the matrix $\mathbf{Q}_N$ with the vector $\mathbf{m}$. To takcle this problem, we applied a pre-processing to the plaintext before to be encrypted. This pre-processing is sensitive to an initial condition, and it also avoids encrypting directly the plaintext sequence. Despite the high adjacent correlation, the plaintext blocks are replaced without reveal information. As we said before in Section 3.5 of Chapter, the proposed solution consists in the use of h function to pre-process the plaintext data into an intelligible form. In this case, at first the $\mathbf{m}$ vector, instead the $\mathbf{x}$ vector, and the initial random $\mathbf{z}$ vector instead $\mathbf{y}$ vector, are introduced as input to obtain an $\widehat{\mathbf{m}}$ sequence.

A processed image with this operation changes all coefficients as shown in Fig. 4.8b), where we can see the steps for the image encryption.

An original image, see Fig. 4.8 a), processed with this operation will change all of its coefficients as is shown in Fig. 4.8 b). The encrypted modified image is illustrated in Fig. 4.8 c).In the Figure below is the respective histogram analysis. We can see an

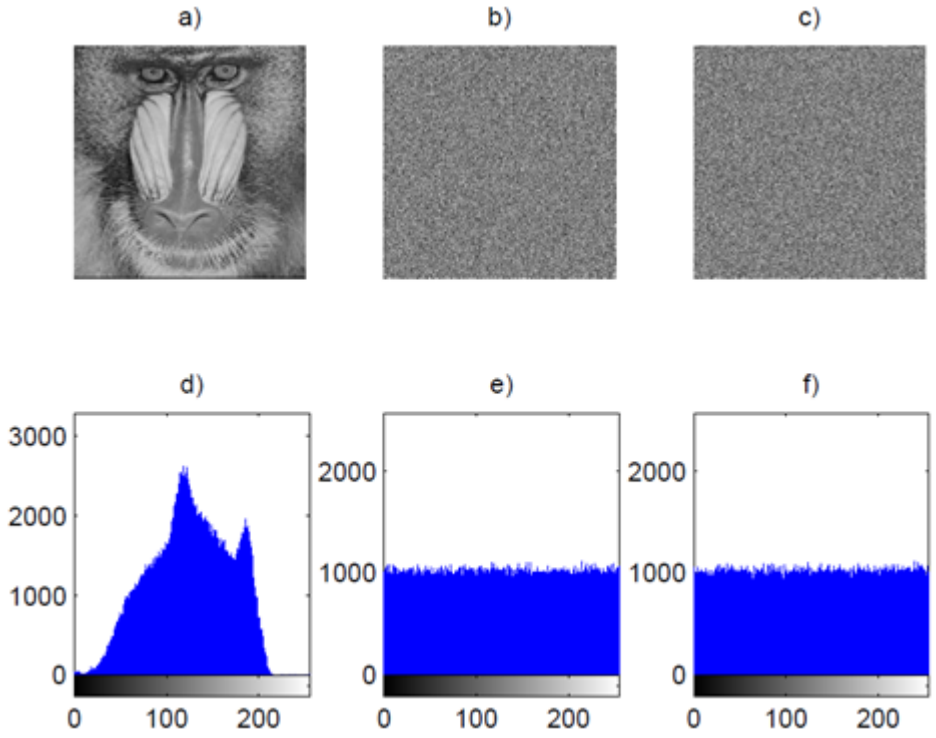

Figure 4.8: Histogram analysis for the gray-scale mandrill test image. Top row: (a) The plainimage, (b) the processed-image and (c) the ciphered-image. Bottom row: (d) The histogram of (a), (e) the histogram of (b) and (f) the histogram of (c).

uniform distribution for the pre-processed and encrypted versions, i.e., the redundancy of the original image was hidden even before to be encrypted. The performance of the new scheme under the CPA is illustrated in the Fig.4.9.

Now, the recovered image does not reveal information about the original image. Despite the enhanced ESCA system is secure, when a solid image $\mathbf{I}_0$ with $\mathbf{m} = 0$ is processed, its histogram does not has a uniform distribution. With the aim to improve this aspect, we developed another version of the pre-process based on three interconnected $\mathbf{H}_{Nt}$ matrices, as is shown in the Fig. 4.10. This option allows to disperse a solid black image in a better

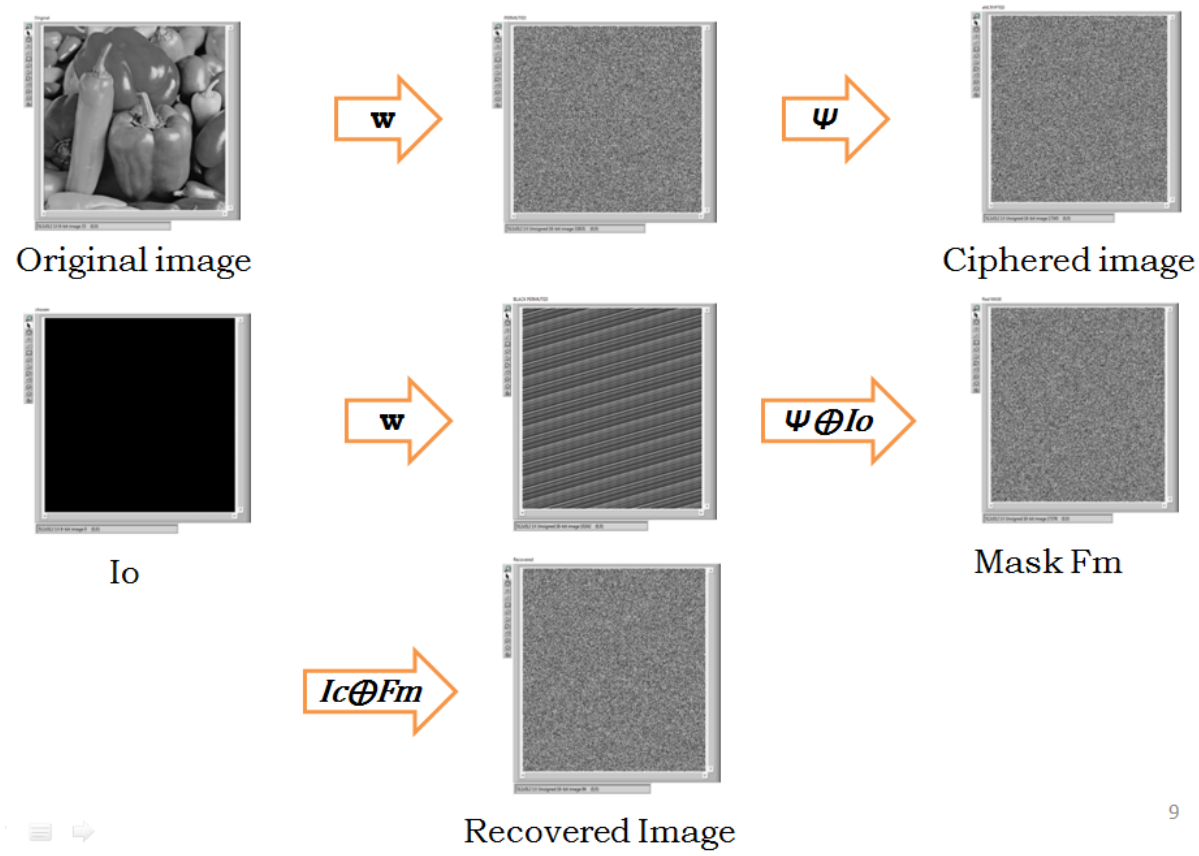

Figure 4.9: New scheme under the Chosen-plaintext attack

way, as it probes the uniform histogram of the processed image shown in the Fig. 4.11.

4.1.2.7 Differential cryptanalysis

In image encryption, the cipher resistance to differential attacks is commonly analyzed with two measures: NPCR (number of pixels change rate) and UACI (unified averaged changing intensity). Both measures are based on slight changes of two images keeping the key unchanged. To illustrate this, let us assume two ciphered images $\mathbf{C}1$ and $\mathbf{C}2$, whose

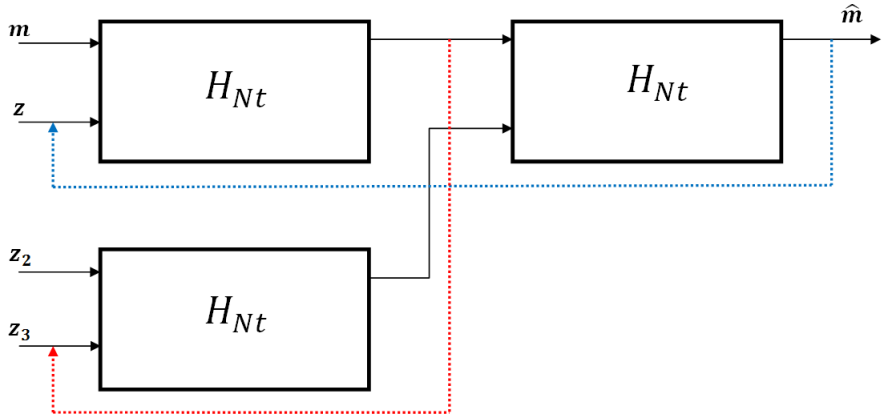

Figure 4.10: Modified process with three interconnected functions h.

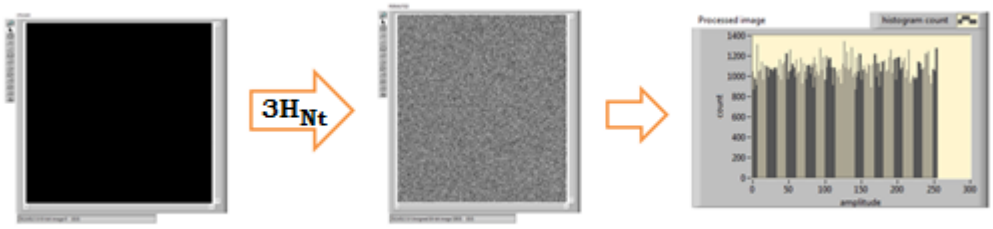

Figure 4.11: Pre-processed version of a solid image

corresponding plain-images have only one-pixel difference under the same key. The gray-level values of ciphered images $\mathbf{C}1$ and $\mathbf{C}2$ at row i, column j are labeled as $\mathbf{C}1(i,j)$ and $\mathbf{C}2(i,j)$, respectively. The NPCR and UACI measures are defined as:

$$NPCR(\mathbf{C}1,\mathbf{C}2) = \sum_{i=0}^{N}\sum_{j=0}^{N}\frac{D(i,j)}{T} \times 100 \tag{4.5}$$

$$UACI(\mathbf{C}1,\mathbf{C}2) = \sum_{i=0}^{N}\sum_{j=0}^{N}\frac{\|\mathbf{C}1(i,j) - \mathbf{C}2(i,j)\|}{F \times T} \tag{4.6}$$

where $D(i,j) = 0$ if $\mathbf{C}1(i,j) = \mathbf{C}2(i,j)$, otherwise $D(i,j) = 1$, T is the total of pixels of the images, and F denotes the largest supported pixel value compatible with the cipher-text image format. For a 256 gray levels image the maximum theoretical values are 33.464% for UACI, and 99.609% for NPCR. This process is illustrated in Fig. 4.12, where the changed pixel is on the center of the test images and only varies the less significant bit. The results obtained with our algorithm for the test images are shown in the Table 4.6:

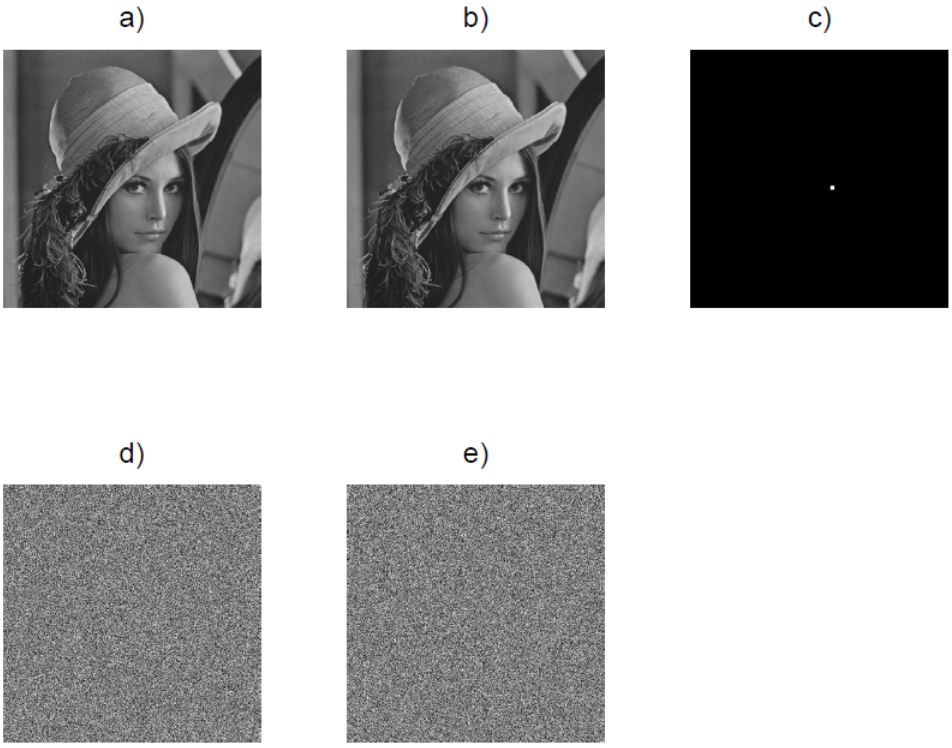

Figure 4.12: a) Plain-image. b) Plain-image after change a pixel c) different pixel between a) and b). d)ciphered version of a). e) ciphered version of b)

Table 4.6: NPCR & UACI of the test images

Image	NPCR	UACI
Mandrill	99.601%	33.347%
Lena	99.602%	33.365%
Peppers	99.601%	33.387%

4.1.3 Compression stage

As we see previously, the ESCA system demonstrated to be a secure system but with a high latency. To reduce the processing time, we decided to use a lossy compression scheme based on the two-dimensional Discrete Wavelet Transform (2D-DWT), in fact, the two-dimensional Haar wavelet transform was considered in this application, because with this wavelet function the algorithm is memory efficient and reversible. The compression procedure that we employ is shown in Fig. 4.13. First of all, the Haar wavelet transform is applied to the initial image. Next, the transformed image is submitted to an elimination process of the transformed coefficients which lie below a threshold value. In fact, the key step here is to choose a threshold through an energy criterion. We look at the normalized cumulative energy applied to the ordered transformed coefficients. To select the threshold value we consider the magnitude of the coefficient for which a proposed energy percentage is obtained. With an established threshold value ε any coefficient in the wavelet transformed data whose magnitude is less than ε will be reset to zero. Hence the amount of obtained compression can be controlled by varying the threshold parameter ε [36].

In the numerical implementation, the wavelet compression scheme was tested with the same images. The individual results of the wavelet compression procedure for the gray-

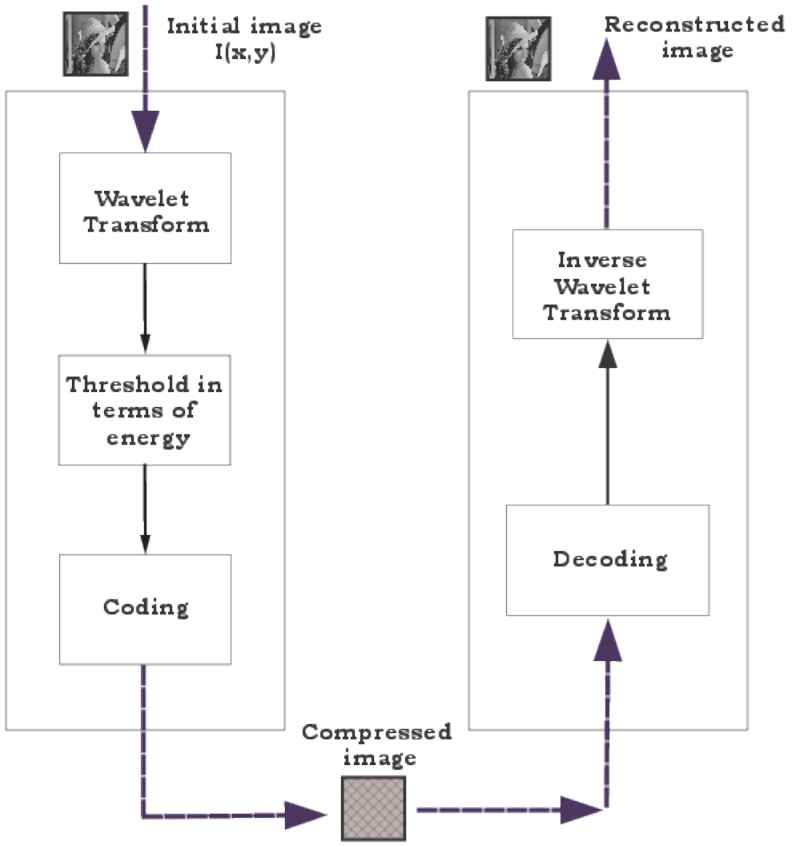

Figure 4.13: Illustration of the basic wavelet compression procedure based on energy approach.

level mandrill image can be observed in Fig. 4.14, where some reconstructed images for different energy criteria are shown. The quality of the recovered image in the inverse process is measured with the Peak signal-to-noise ratio (PSNR). In Table 4.7 are shown the PSNR and compression rate for the three test images. Based on these results is clear that the Haar wavelet transform will allows us to achieve good compression rates, without much degradation on the reconstructed images.

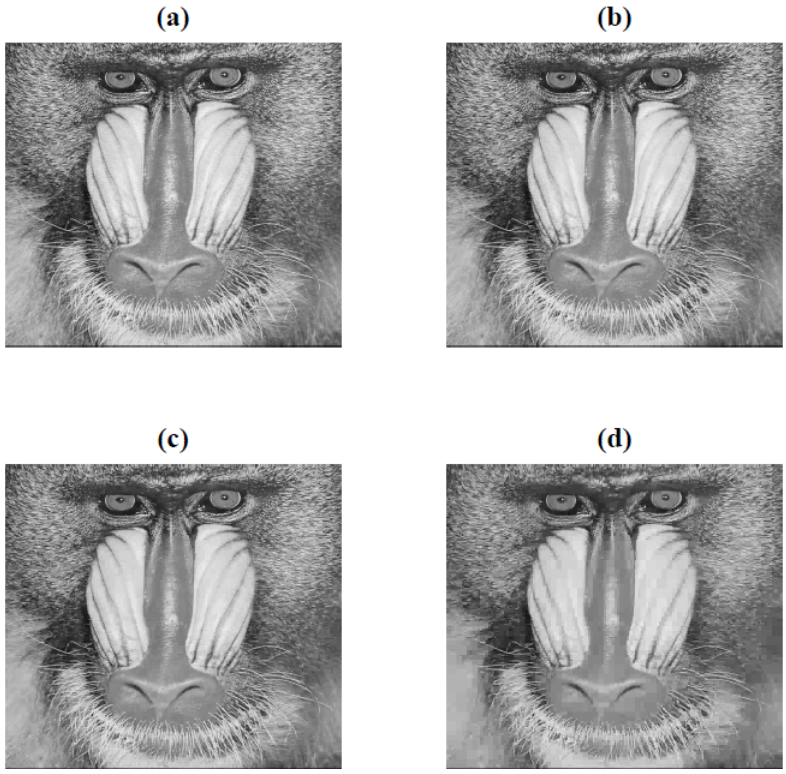

Figure 4.14: a) The source image. Reconstructed images for a b) 90 %, c) 75 %, and d) 50 % of energy considered in the compression stage.

Table 4.7: Compression rate in terms of energy percentage, and the respective PSNR parameter.

	Lena			Mandrill			Peppers		
	90%	75%	50%	90%	75%	50%	90%	75%	50%
PSNR (dB)	35.9798	28.7091	21.0777	31.8236	24.9937	19.0352	36.829	29.7466	20.4291
Compression Ratio	2.5:1	6.7:1	48.9:1	2.1:1	3.9:1	12.3:1	2.6:1	8.4:1	82.4:1

4.1.4 Numerical implementation of joint compression encryption scheme

To implement numerically the joint scheme, we consider the graphical programming language of LabVIEW, a trademark of National Instruments. In Fig. 4.15 is depicted this scheme, which comprises two stages. The top block, Module A, carries out the compression and encryption of images, as was described above, whereas the bottom block, Module B, performs the reverse process to obtain a reconstructed image. It is worth to say that we have two signals after the compression stage, the wavelet coefficients that survived to the value of the threshold, which are the values to encrypt, and a binary vector indicating the positions of such coefficients. Hence the encrypted image and the binary vector are available to be transmitted through a public channel, but it will depend on the application if it is required to convey. Of course, in the Module B the encrypted image received is decrypted, and the decompression procedure takes place to the decrypted image with the binary position vector obtaining a reconstructed image.

Figure 4.16 illustrates the results obtained in the stages contained in Module A for a source Lena image. In this case, an energy criterion of 90 % was considered to compress the source image. As an efficiency or time execution analysis, in Table 4.8 are shown the results of the execution time of the ESCA system without compression and the new scheme for the same energy criterion of 50%. In both systems the encryption key is of 31 bits. This decrease in the latency is due to the reduction of the data to encrypt. To see the recovered RGB images in module B, the Fig. 4.17 shows the results of the test images. With such a speed, this approach is appealing for a wide range of applications that require real-time processing.

In LabVIEW we developed a Virtual Instrument (VI) that uses the webcam to capture a video sequence configured at 30 frames per second and 128×128 pixels, and the sequence

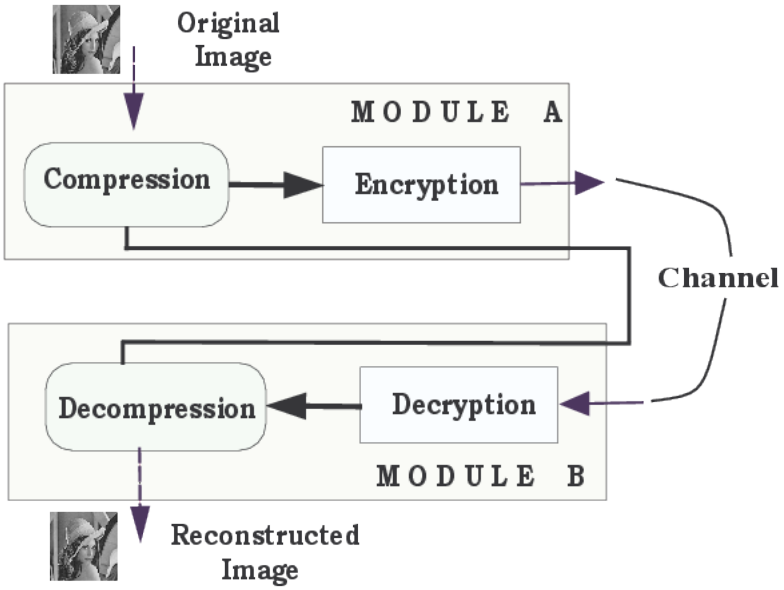

Figure 4.15: Image compression-encryption scheme.

	Grayscale 8 bits		RGB 24 bits	
	ESCA 31 bits	Proposed scheme	ESCA 31 bits	Proposed scheme
Lena	692 ms	136 ms	940 ms	380 ms
mandrill	692 ms	138 ms	940 ms	384 ms
peppers	692 ms	128 ms	940 ms	356 ms

Table 4.8: Comparison of the execution times of both schemes.

of images are processed with our proposed scheme. The modules A and B of Fig. 4.15 are included in this program. As we can see in the Fig. 4.18, the compressed-encrypted video signal is shown in the left display of the front panel. The recovered video signal is shown on the right display of the front panel. In this implementation, the user assigns the levels of the 2D-DWT and the energy percent to preserve, and the VI program calculates

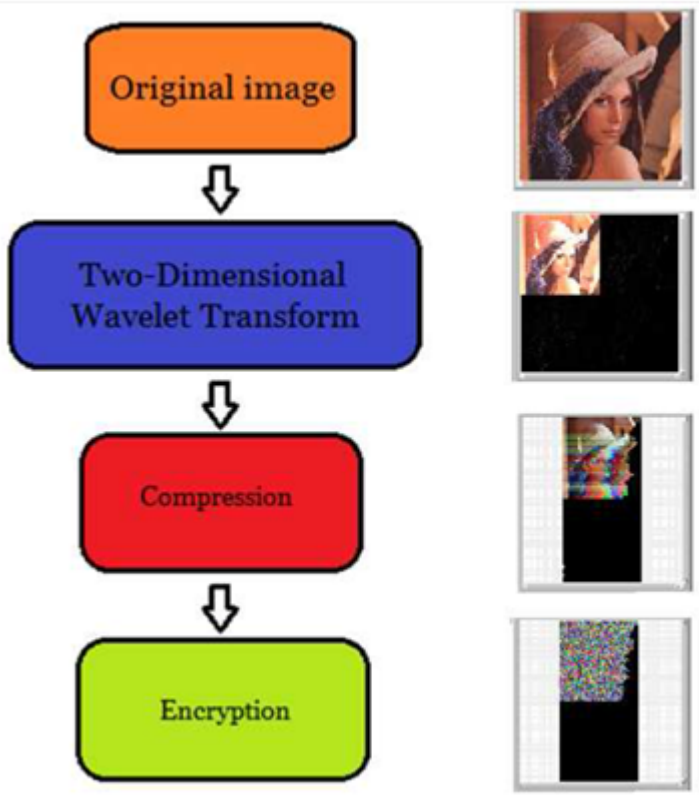

Figure 4.16: Results of the application of Module A to a source RGB Lena image.

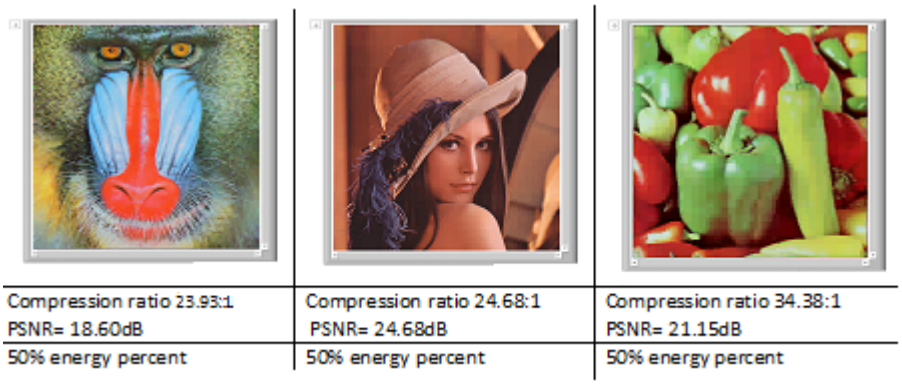

Figure 4.17: Recovered images after be compressed-encrypted.

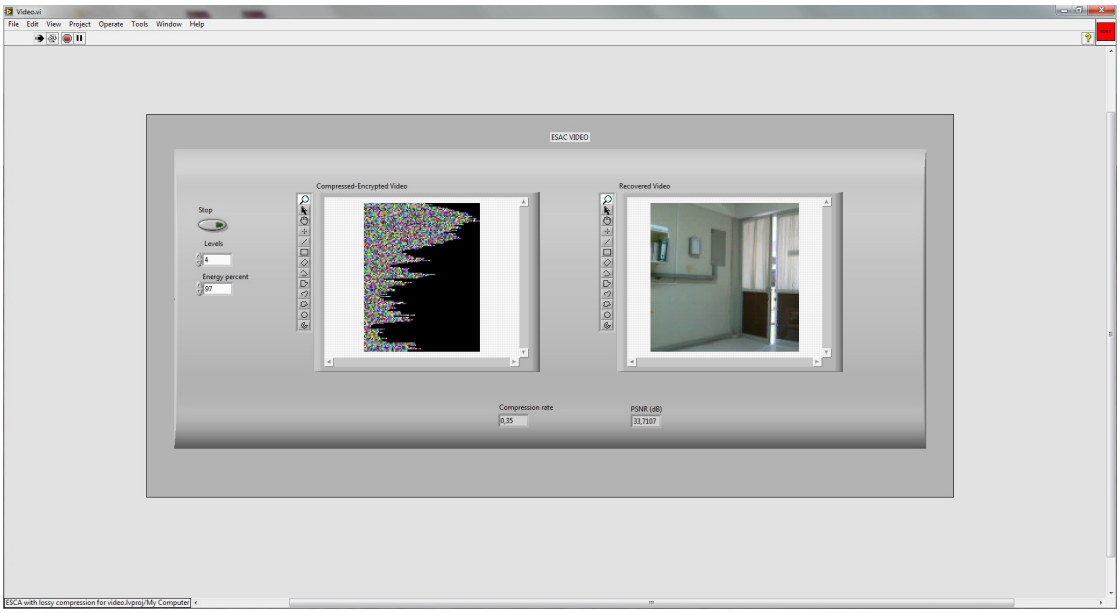

Figure 4.18: Compression-encryption scheme on a real-time video application.

the compression rate of the compressed-encrypted images, the PSNR of the recovered images, and the recovered video is played smoothly with the appropriate parameters. Obviously, the performance will depend on the computer and webcam, besides the user can manipulate the parameters to get a certain quality of the recovered video.

4.2 FPGA implementation of the ESCA system

The second application of the ESCA system is a hardware implementation using the soft processor core MicroBlaze in a Virtex-5 VC5VLX110T FPGA. With this embedded system we can have a reconfigurable encryption system that allows to the user may select among different configurations without the need to program the FPGA every time. With the different options to configure the ESCA system, we may get an increase in security or

have a customized system for a faster performance. With this experimental implementation we carry out the encryption of grayscale and RGB color images, and in addition a sparse matrix format was implemented to reduce the latency. This implementation is described as follows, in section 4.2.1 are briefly described the embedded systems, in section 4.2.2 is described the softcore processor. The hardware implementation on the FPGA is detailed in section 4.2.3, the sparse matrix format is explained in section 4.2.4. And in section 4.2.5 is described the reconfigurable image encryption system..

4.2.1 Embedded systems

A computer is defined as a electronic device with a processor unit, memory and I/O peripherals. The computers can be classified in two main categories: of general purpose and embedded. A computer of a general purpose is for example a desktop PC, a laptop, a tablet, a smartphone, etc. These devices are designed to perform common computing tasks, where the user interacts directly with them. The typical peripherals of the general purpose computers (keyboard, mouse, monitors) use standard codifications. On the other hand, an embedded system is described as specialized computer, where generally it is a component of a bigger product and has a specific purpose. Usually, the final user does not directly interact with the embedded system, or interacts with a limited interface. These systems are ideal to work with special peripherals, and they can still work with standard peripherals.

Nowadays, the Field Programmable Gate Arrays (FPGAs) have a wide acceptance due to their capacity and flexibility, tThese are some reasons that the designers of embedded systems take into account to choose them for their projects. The circuits of a FPGA can be even reprogrammed when it has already been installed in a product. As their resources

are reconfigurable, they can be used efficiently in function of a specific architecture. The current capacity of the FPGAs allows us to implement a processor (or several processors) in their resources. Therefore a physical processor can be avoided in a design and this kind of processors are called soft processor core. Before to continue, it is necessary to define some terms as IP core, diffused core and soft core. An IP core (Intellectual Property core) is a hardware specification to physically manufacture an integrated circuit or to configure resources of an FPGA. A Diffused core is a physical processor in the silicon device, and it is also called hard core. Finally, a soft processor core is specified by software, its design and architecture. It is implemented in the reconfigurable resources of an FPGA[37]. The principal soft processor cores in the market are MicroBlaze and PicoBlaze by Xilinx, and NIOS II by Altera. Besides these processors offered by their respective companies, there are an independent platforms like LEON of 32 bits.

4.2.2 MicroBlaze

MicroBlaze is a soft processor core of 32 bits with a Harvard architecture, it was developed and commercialized by Xilinx for the FPGAs families Spartan, Virtex, Kintex and Artix. The Fig. 4.19 shows a block diagram of MicroBlaze.

The utility Base System Builder (BSB) allows to develop a system based on a MicroBlaze processor. BSB helps to build a "hardware" platform, it allows designers to set up system attributes as select or interconnect processor-cores (MicroBlaze or PowerPC), add peripherals, select memory size and type, among others. BSB and MicroBlaze are provided by Xilinx Project Studio (XPS), that is a tool of the Embedded Design Kit (EDK).

There are several advantages of using a MicroBlaze processor instead a hard core processor. One of the reason is that in this case it is possible transfer designs between

devices of the same family without compatible problems. There is no necessity to include a physical processor in the designs. MicroBlaze is highly parameterizable, it allows designers to make choices based on his own design requirements to build a custom hardware platform. The Fig. 4.19 shows in blue color, the optional features of MicroBlaze (divide unit, barrel shifter, FPU, etc). In the same way, the designer selects the peripherals to implement (UARTs, timers, memories, network controllers etc.).

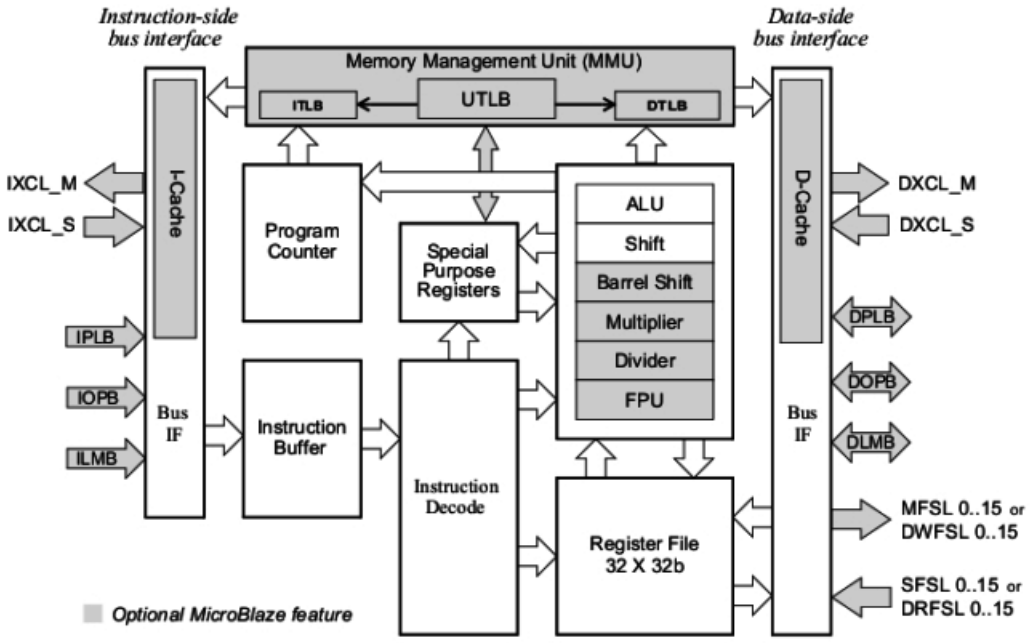

Figure 4.19: MicroBlaze block diagram, the optional MicroBlaze features are in blue. Image taken from Xilinx's website.

The On-Chip Peripheral Bus (OPB) is used to connect the peripherals. The speed of a system based on MicroBlaze will depend on the size of the entire FPGA design and the ability of the implementation tools to place and route the design optimally. Therefore, to improve the embedded system speed is necessary to optimize FPGA resources, because is less logic to be placed and routed.

4.2.3 Hardware implementation of ESCA system

For the hardware implementation of the ESCA system with matrix approach, we use a Virtex-5 VCLX110T FPGA with the soft processor core MicroBlaze. This implementation is mostly a MicroBlaze design, as opposed to being an IP design. With this embedded system, the user can configure the encryption system according to his needs.

The hardware platform implemented with the BSB wizard for the ESCA system, includes a MicroBlaze processor with a clock of 125 MHz, and an RS-232 serial port. The HyperTerminal is used to visualize the calculations of the ESCA system. At first, the user gives the parameter of the key length, where the options are 15, 31 and 63 bits. After that, the user provides the bit-length of the block information, selecting among 8, 16, 24, 32 and 56 bits. Finally, the user assigns the number of PRNGs to compute the secret keys the options are one or three. In Fig. 4.20 is illustrated this process, whereas in the Fig. 4.21 is shown two different configurations for the ESCA system and how it begins to calculate the matrices. The main matrices of the functions Ψ, Φ and h are calculated when the user assigns the key and plaintext lengths. The latter process saves resources of the FPGA, because is not necessary store all the matrices for different possible keys and plaintext lengths.

4.2.4 Sparse matrix

In order to reduce the time of execution in the matrix operations, we consider the Yale sparse matrix. With this format a matrix is stored in three one-dimensional arrays. The first array, named A, holds all the matrix values different to zero, left-to-right and top-to-bottom order. The second array, named IA, holds the sum of the non-zero elements

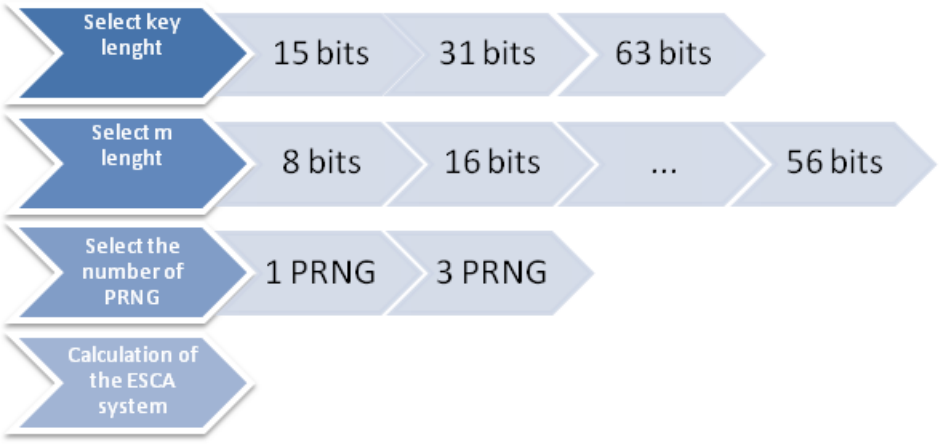

Figure 4.20: Configuration of the ESCA system.

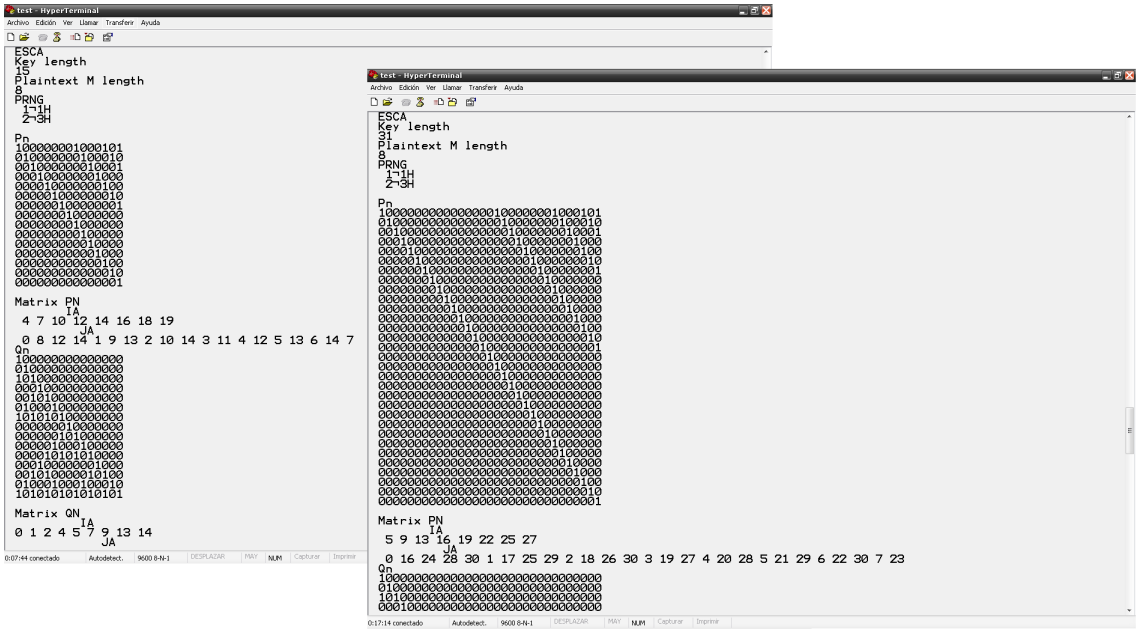

Figure 4.21: Encryption systems with k=15, 31 bits and for both m=8 bits, seen from Hyperterminal.

Table 4.9: Latency time comparison between the 3 PRNGs and 1 PRNG algorithms.

	3 PRNG				1 PRNG			
m bits	56	24	16	8	56	24	16	8
k 63 bits	5379	2734	2530	2347	3326	1849	1635	1453
k 31 bits		1902	1676	1472		1084	857	653
k 15 bits				669				356

from each row from top-to-bottom order, where the first element should be zero. The last array, JA, lists the column index of each elements of the array A. The implementation of this procedure reduces the latency time until 73% approximately to carry out the matrix operations. Besides without this format, it would not be possible to implement the ESCA system of 63 bits without increasing the program memory. Because we can save memory if the operation matrix is computed and passed to its vectors IA and JA, the 2D-arrays needed are less and the vector A is omitted because all values different to zero are one. The Fig. 4.22 shows the version of 63 bits, now the initial condition that the user can select is of 311 bits.

It is important to say, that depending of the chosen configuration, the system has different latency times. The Table 4.9 shows the execution times for each configuration to encrypt 10,000 data, where the results are in milliseconds. The resource utilization of the FPGA in our design is shown in Fig. 4.23.

4.2.5 Hardware implementation for image encryption

To encrypt images, we developed a host program in LabVIEW, see Fig. 4.24. The host sends and receives the images from the FPGA, and saves the encrypted/decrypted images.

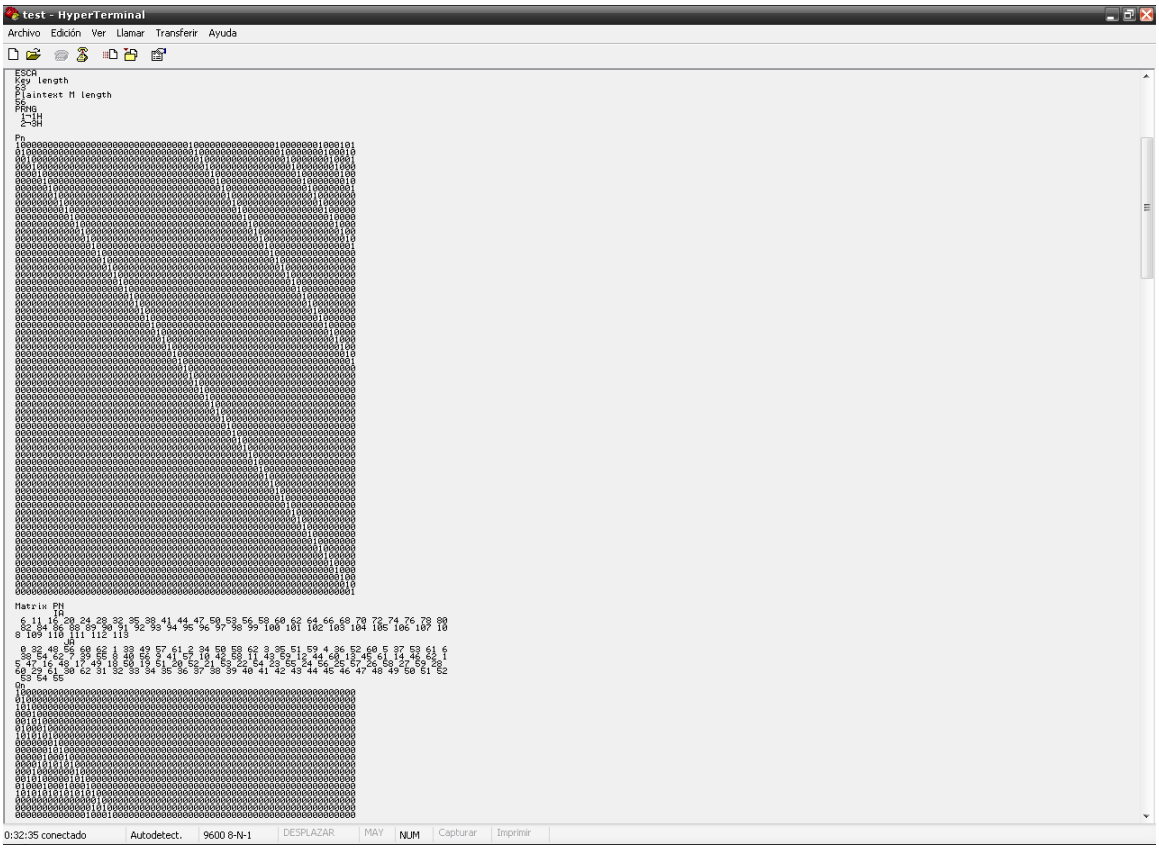

Figure 4.22: ESCA system for k=63 bits and m=56 bits, seen from Hyperterminal.

The communication between the FPGA and the host program is via the RS-232 serial port. After the user configures the ESCA system, the host program writes a pixel or a block of pixels on the port as the $\mathbf{m}$ data. The embedded system encrypts the $\mathbf{m}$ data and writes on the serial port the corresponding $\mathbf{c}$ data.

```
Device utilization summary:
---------------------------

Selected Device : 5vlx110tff1136-1


Slice Logic Utilization:
 Number of Slice Registers:            1998  out of  69120     2%
 Number of Slice LUTs:                 1843  out of  69120     2%
    Number used as Logic:              1747  out of  69120     2%
    Number used as Memory:               96  out of  17920     0%
       Number used as RAM:               64
       Number used as SRL:               32

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:   3027
   Number with an unused Flip Flop:    1029  out of   3027    33%
   Number with an unused LUT:          1184  out of   3027    39%
   Number of fully used LUT-FF pairs:   814  out of   3027    26%
   Number of unique control sets:       185

IO Utilization:
 Number of IOs:                         11
 Number of bonded IOBs:                 11  out of    640     1%

Specific Feature Utilization:
 Number of Block RAM/FIFO:              16  out of    148    10%
    Number using Block RAM only:        16
 Number of BUFG/BUFGCTRLs:               2  out of     32     6%
 Number of DSP48Es:                      3  out of     64     4%
 Number of PLL ADVs:                     1  out of      6    16%
```

Figure 4.23: Synthesis report.

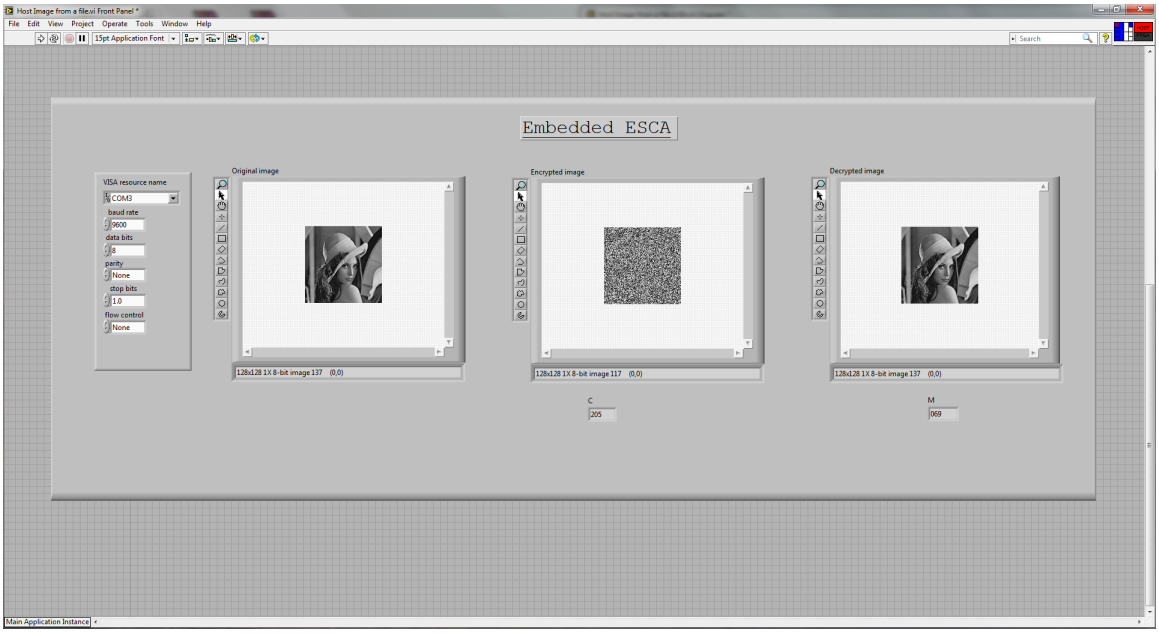

Figure 4.24: Host-program for image encryption.

5

Conclusions

In this work, we present an improved version of the encryption system ESCA, and its implementation. This new version is resistant to model threat chosen-plaintext attack, and it is perfectly secrecy. The different kinds of analysis show that the ESCA system hides the redundancy of the original image, the encrypted image is an independent image without adjacent correlation in the pixel neighborhood. Also the ESCA system proved has a high key sensitivity, and under the assumptions of the Chosen-plaintext attack the algorithm does not reveal any information of the original image. Besides, the improvements done to the ESCA system have also a matrix approach, allowing a reconfigurable implementation.

Our proposed scheme that joins a compression stage with encryption procedure could be an option to real-time video transmissions. The proofs and results show that this scheme reduces latency almost at 80% if we compare with an only encryption program, and the recovered image has a low degradation.

The hardware implementation in an FPGA allows us, to use new architectures and platforms. The embedded system is able to generate different encryption configurations

from a single code due to matrix approach. The optimization of the main program using a sparse matrix format, brings us the possibility of increase the initial condition to 311 bits and reduces the latency time at least 75%.

We have used the wavelet-based variant of the multifractal DFA to reveal the multifractal features of some matrices that carry out the main functions in an encryption system, with this spirit, now we determine scaling properties in the encrypted images by means of a two-dimensional DFA approach. The results in Ref. [41] point to the possibility that the DFA scaling exponent can be used as an appropriate and objective measure of the quality of encryption schemes. We are still working on new calculations and tests.

The cryptography is not a rigorous science, it maintains art aspects, this allow to be creative and propose new ways and methods to protect the information. The hardware security shows that there are still many aspects to remedy[39]. Physical attacks for example the side-channel attack demonstrates that an insecure hardware implementation allows to break secure cryptography codes like AES[40]. Only when a scheme in the real world can satisfy the security needs, many areas will be fully on electronic platforms, increasing their efficiency.

Appendix A

In the same spirit that Ref. [27] the multifractal features of the main matrices of ESCA system were obtained. For this purpose, we consider the scaling method known as the wavelet transform multifractal detrended fluctuation analysis (WT-MFDFA). The efficient WT-MFDFA method reveals multifractal properties of the sum of ones in the sequences of the rows of the encryption main matrices that can be used for their characterization. In addition, we analyze the multifractal structure of the matrices of different dimensions, and find that there are minimal differences in all the examined multifractal quantities such as the multifractal support, the most frequent singularity exponent, and the generalized Hurst exponent.

The WT-MFDFA method is based on the calculation of the so-called qth order fluctuation function defined as

$$F_q(s;m) = \left\{ \frac{1}{2M_s} \sum_{v=1}^{2M_s} \mid F^2(v,s;m) \mid^{q/2} \right\}^{1/q}, \tag{1}$$

where $q \epsilon Z$ with $q \neq 0$. The diverging behavior for $q \rightarrow 0$ can be avoided by using a logarithmic averaging for the 0th order fluctuation function $F_0(s;m) = \exp\left\{ \frac{1}{2M_s} \sum_{v=1}^{2M_s} \ln \mid F^2(v,s;m) \mid^{q/2} \right\}$. To assess the fractal scaling behavior of a time series, the fluctuation function $F_q(s;m)$ should reveal a power law scaling

$$F_q(s;m) \sim s^{h(q)}. \tag{2}$$

The exponent $h(q)$ is the generalized Hurst exponent since it depends on q, while the original Hurst exponent is $h(2)$. If $h(q)$ is constant for all q then the time series is monofractal, otherwise it has a multifractal behavior. In the latter case, one can calculate various other multifractal scaling exponents, such as the singularity (Hölder) spectrum $f(\alpha)$ of a signal or distribution $g(t)$ as the Legendre transform of the scaling exponent $\tau(q)$.

$$\alpha = \frac{d\tau(q)}{dq} \qquad \text{and} \qquad f(\alpha) = q\alpha - \tau(q), \tag{3}$$

where α characterizes the strength of singularities, and the Hölder spectrum of dimensions $f(\alpha)$ is a non-negative convex function that is supported on the closed interval $[\alpha_{\text{min}}, \alpha_{\text{max}}]$. The spectrum $f(\alpha)$ can be interpreted as the Hausdorff fractal dimension of the subset of data characterized by the Hölder exponent α. The most "frequent" singularity, which corresponds to the maximum of $f(\alpha)$, occurs for the value of $\alpha(q = 0)$, whereas the boundary values of the support, α_{min} for $q > 0$ and α_{max} for $q < 0$, correspond to the strongest and weakest singularity, respectively. On the other hand, a linear behavior of $\tau(q)$ indicates monofractality whereas a nonlinear behavior indicates a multifractal signal. Additionally, there is a relation between the scaling exponent $\tau(q)$ and the generalized Hurst exponents $h(q)$, which is given by $\tau(q) = qh(q) - 1$.

As we saw before in this thesis, the main sequence matrix $\mathbf{Q}_N$ is based on the evolution of rule 90, so we analyze with the WT-MFDFA method the sum of ones in the sequences of the rows of this matrix, using the db-4 wavelet function belonging to the Daubechies orthogonal family. We select this wavelet function because of its desirable properties,

such as orthogonality, approximation quality and numerical stability; in addition, the algorithm with the Daubechies family wavelet functions is memory efficient and is reversible, whereas other wavelet bases have a slightly higher computational overhead and are conceptually more complex.

The results for the row sum of the encryption matrix $\mathbf{Q}_N$ are illustrated in Fig. 1. We consider $N = 2^{12} - 1 = 4095$ data points of the time series, and the fact that the generalized Hurst exponent is not a constant horizontal line is indicative of a multifractal behavior in this time series, see Fig. 1(b). Since the scaling exponent is not of a single slope, Fig. 1(c), it can be considered as another clear feature of multifractality. The strength of the multifractality is roughly measured with the width $\triangle\alpha = \alpha_{\text{min}}, \alpha_{\text{max}}$ of the "parabolic" singularity spectrum $f(\alpha)$ on the α axis, see Fig. 1(d). For this matrix, the width $\triangle\alpha_{\mathbf{Q}_{4095}} = 1.0160 - 0.0080 = 1.0080$, and the "most" frequent singularity occurs at $\alpha_{\text{mf}_{\mathbf{Q}_{4095}}} = 0.5540$.

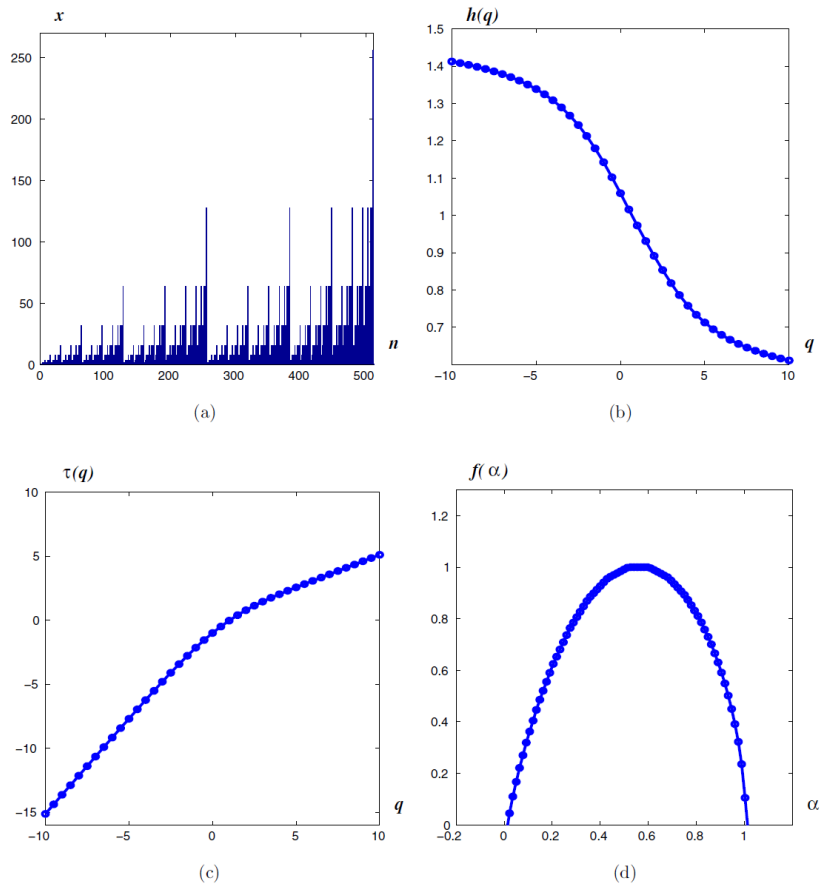

Figure 1: (Color online) (a) Time series of the row signal of $\mathbf{Q}_{4095}$. Only the first 2^9 points are shown of the whole set of $(2^{12}-1)$ data points. (b) The generalized Hurst exponent $h(q)$, (c) the τ exponent, where $\tau(q) = qh(q) - 1$ and (d) the singularity spectrum $f(\alpha) = q\frac{d\tau(q)}{dq} - \tau(q)$.

On the other hand, the multifractal results for the decryption matrices, $\mathbf{T}_N$ and $\mathbf{R}_N$, are displayed in Fig. 2. We notice that both time series have a smaller that the previous one. However, both of them present a similar multifractal behavior.

For instance, the width $\triangle\alpha_{\mathbf{R}_{4095}} = 0.9740 - 0.2180 = 0.7560$ and $\triangle\alpha_{\mathbf{T}_{4095}} = 0.9600 -$

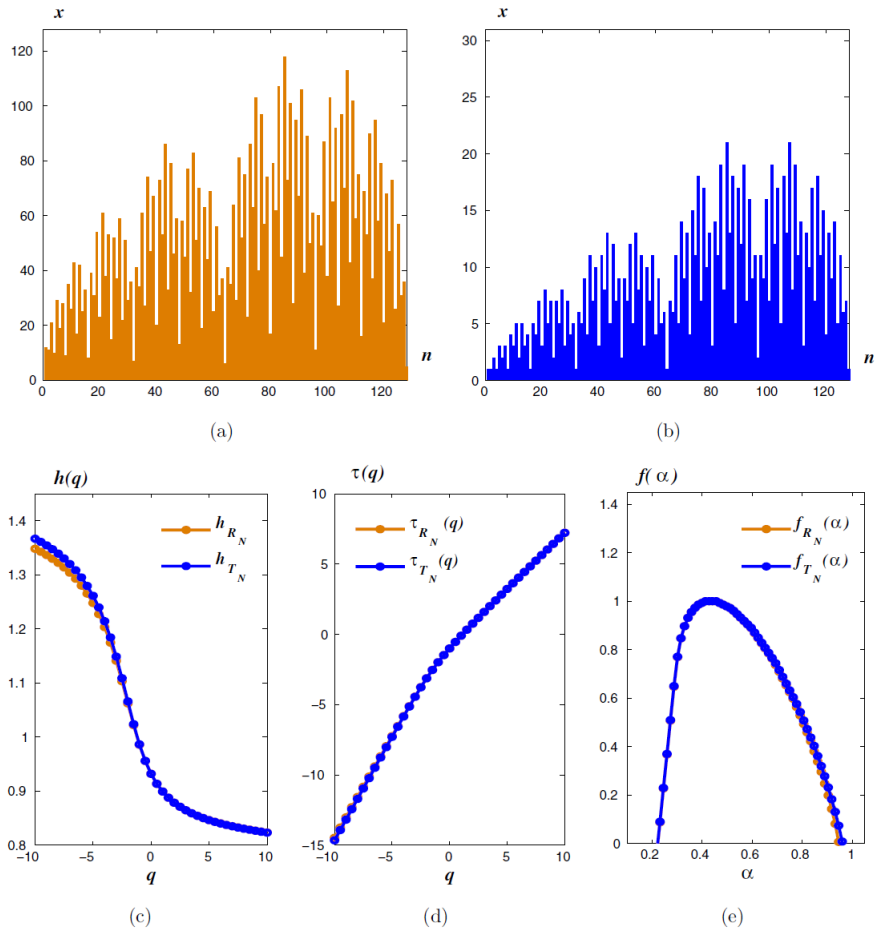

Figure 2: (Color online) Time series of the row signal of the matrices (a) $\mathbf{R}_{4095}$ and (b) $\mathbf{T}_{4095}$. Only the first 2^8 points are shown of the whole set of $(2^{12}-1)$ data points. (c) The generalized Hurst exponent $h(q)$ for the row signals $\mathbf{R}_{4095}$ and $\mathbf{T}_{4095}$, (d) the exponent τ for each signal and (e) the corresponding singularity spectrum $f(\alpha)$.

$0.2180 = 0.7420$ do not present a big difference. In fact, the "most" frequent singularity occurs at the same alpha value $\alpha_{\text{mf}_{\mathbf{R}_{4095}}} = \alpha_{\text{mf}_{\mathbf{T}_{4095}}} = 0.4280$. These results were expected since these decryption matrices depend on the matrix $\mathbf{Q}_N$.

In order to assess the impact of finite signal lengths on the multifractal quantities, we apply the WT-MFDFA technique to the time series obtained from the encryption matrices involved in the ESCA system for different lengths: $N = 2^n - 1$, $n = 9, 10, 11, 12, 13$. Table 1 summarizes the results obtained, where we notice that the results of matrix $\mathbf{Q}_N$ are more stable than the results obtained from the decryption matrices, $\mathbf{T}_N$ and $\mathbf{R}_N$.

Table 1: Table 2. The values of the width $\triangle\alpha = [\alpha_{\text{min}}, \alpha_{\text{max}}]$ and the most "frequent" singularity, α_{mf} , for different dimensions of the three matrices $\mathbf{Q}_N$, $\mathbf{R}_N$ and $\mathbf{T}_N$ obtained by means of the WT-MFDFA method.

		n				
Matrix		9	10	11	12	13
$\mathbf{Q}_N$	α_{min}	0.1200	0.0500	0.0220	0.0080	0.0080
	α_{max}	1.1000	1.0580	1.0300	1.0160	1.0160
	$\triangle\alpha$	0.9800	1.0080	1.0080	1.0080	1.0080
	α_{mf}	0.6520	0.5960	0.5820	0.5540	0.5540
$\mathbf{R}_N$	α_{min}	0.0360	0.1060	0.1620	0.2180	0.2460
	α_{max}	0.8060	0.8760	0.9320	0.9740	0.9880
	$\triangle\alpha$	0.7700	0.7700	0.7700	0.7560	0.7420
	α_{mf}	0.2880	0.3440	0.4000	0.4280	0.4560
$\mathbf{T}_N$	α_{min}	0.0220	0.1060	0.1620	0.2180	0.2460
	α_{max}	0.8060	0.8760	0.9320	0.9600	0.9880
	$\triangle\alpha$	0.7840	0.7700	0.7840	0.7420	0.7420
	α_{mf}	0.3020	0.3440	0.4000	0.4280	0.4560

It is worth pointing out that we studied the intrinsic multifractal properties of the main sequence matrices of an encryption system having in mind their possible usage in

cryptanalysis. Some authors have considered the generalized Hurst exponent $h(q)$ as a representative multifractal parameter. They analyze a chaotic carrier with an embedded signal and found that this quantifier helped to detect the presence of a message, meaning that it can be used as an efficient measure of encryption schemes. But, the analysis is performed on encrypted signals and not directly to the encryption elements of the encryption system as we do here. In Ref. [42] are shown in detail these results.

Bibliography

[1] L. Kocarev, IEEE Circuits Syst. Mag. 1, 6 (2001).

[2] N. Masuda and K. Aihara, IEEE Trans. Circuits and Syst.-I 49, 28 (2002).

[3] V. Patidar and K. K. Sud, Electron. J. Theor. Phys. 6, 327 (2009).

[4] S. J. Xu, X. B. Chen, R. Zhang, Y. X. Yang and Y. C. Guo, Phys. Lett. A 376, 1003 (2012).

[5] Zhi-Hong Guan, Fangjun Huang, Wenjie Guan "Chaos-based image encryption algorithm" Physics letters A, 2005

[6] Xiaojun Tong, Minggen Cui "Image encryption with compound chaotic sequence cipher shifting dynamically" Image and vision computing, ELSEVIER, 2008

[7] C. Cokal and E. Solak, Phys. Lett. A 373, 1357 (2009).

[8] Chengqing Li a, Shujun Li, Guanrong Chen and Wolfgang A. Halang "Cryptanalysis of an Image Encryption Scheme Based on a Compound Chaotic Sequence" Image and vision computing, ELSEVIER, 2009.

[9] S. Wolfram, Advances in Cryptology: Crypto'85 Proceedings, Lecture Notes in Computer Science, Vol. 218 (Springer-Verlag, 1986), pp. 429.

[10] S. Nandi, B. K. Kar and P. P. Chaudhuri, IEEE Trans. Comput. 43, 1346 (1994).

[11] M. Sipper and M. Tomassini, Int. J. Mod. Phys. C 7, 181 (1996).

[12] J. Urías, E. Ugalde and G. Salazar, Chaos 8, 819 (1998).

[13] F. Seredynski, P. Bouvry and A. Y. Zomaya, Parallel Comput. 30, 753 (2004).

[14] A. Fúster-Sabater and P. Caballero-Gil, Appl. Soft Comput. 11, 1876 (2011).

[15] Patent US 5677956 A "Method and apparatus for data encryption/decryption using cellular automata transform"

[16] Patent CN 102523365 B "Method for encrypting and decrypting image based on cellular automata"

[17] Olu Lafe. "Cellular Automata Transforms". Bluwer Academic. USA. (2000).

[18] J. von Neumann, "The Theory of Self-Reproducing Automata" A.W. Burks (ed.), University of Illinois Press, Urbana, IL, 1966.

[19] Stephan Wolfram, "Universality and complexity in cellular automata", Physics D 10, 1984.

[20] J. Urías, G. Salazar, E. Ugalde, Chaos 1998, 8, 814-818.

[21] Edward Aboufadel, Steven Schlicker, "Discovering wavelets", John Wiley and sons (2011)

[22] S. Mallat, "A Wavelet Tour of Signal Processing", 2nd. Edition, Academic Press (1999).

[23] Susan Hansche, John Berti, Chris Hare "Official (ISC)2 Guide to the CISSP Exam" CRC Press, 2003.

[24] J. Urías, E. Ugalde and G. Salazar, "A cryptosystem based on cellular automata" Chaos 8, 819, 1998.

[25] J. S. Murguía, G. Flores-Eraña, M. Mejía Carlos and H. C. Rosu, "Matrix approach of an encryption system based on cellular automata and its numerical implementation" Int. J. Mod. Phys. C 23, 1250078, 2012.

[26] M. T. Ramírez-Torres, J. S. Murguía, and M. Mejía Carlos "Image encryption with an improved cryptosystem based on a matrix approach" Int. J. Mod. Phys. C Vol. 25, No. 10 1450054, 2014.

[27] J. S. Murguía, M. Mejía Carlos, H. C. Rosu, and G. Flores-Eraña, "Improvement and analysis of a pseudo-random bit generator by means of cellular automata", Int. J. Mod. Phys. C, 21, 741 (2010).

[28] G. Flores-Eraña, J.S. Murguía, and M. Mejía-Carlos, "Numerical implementation and analysis of an encryption system", ISUM 2011.

[29] M. T. Ramírez-Torres, J. S. Murguía, and M. Mejía Carlos "Numerical implementation of a real-time encryption system" ENIINVIE-2012, Procedia Engineering 35 (2012) 182 – 191.

[30] M. T. Ramírez-Torres, J. S. Murguía, and M. Mejía Carlos "FPGA implementation of a reconfigurable image encryption system" ReConFig 2014

[31] M. Mejía Carlos, Ph. D. Thesis, Universidad Autónoma de San Luis Potosí, SLP (2001).

[32] F. Han, J. Hu, X. Yu and Y. Wang, Appl. Math. Comput. 185, 931 (2007).

[33] A. Uhl and A. Pommer, "Image and Video Encryption: From Digital Rights Management to Secured Personal Communication" (Springer, 2005).

[34] S. Lian, "Multimedia Content Encryption, Techniques and Applications" (CRC Press, 2008).

[35] A. Martín del Rey, G. Rodríguez Sanchez and A. de la Villa Cuenca, Hybrid Artif. Intell. Syst. Lecture Notes Comput. Sci. 7209, 78 (2012).

[36] James S. Walker "A Primer on Wavelets and Their Scientific Applications", Chapman and Hall/CRC.

[37] Ronald Sass, Andrew G. Schmidt "Embedded Systems Design with Platform FPGAs: Principles and Practices", Morgan Kaufmann Editor

[38] Francisco Rodriguez-Henriquez, N.A. Saqib, Arturo Díaz Pérez, Cetin Kaya Koc "Cryptographic Algorithms on Reconfigurable Hardware" Springer Science & Business Media - Editor

[39] Mohammad Tehranipoor, Cliff Wang "Introduction to Hardware Security and Trust", Springer.

[40] Stefan Mangard "A Simple Power-Analysis (SPA) attack on implementations of the AES key expansion" Lecture Notes in Computer Science Volume 2587 (2003).

[41] C. Vargas-Olmos, J. S. Murguía, M. T. Ramírez-Torres, M. Mejía Carlos, H. C. Rosu, H. Gonzalez-Aguilar, "Two-dimensional DFA scaling analysis applied to encrypted images" Int. J. Mod. Phys. C Vol. 26, No. 8 1550093, 2015.

[42] J. S. Murguía, M. Mejía Carlos, C. Vargas-Olmos, M. T. Ramírez-Torres and H. C. Rosu, "Wavelet Multifractal Detrended Fluctuation Analysis of Encryption and Decryption Matrices" Int. J. Mod. Phys. C Vol. 24, No. 9 1350069, 2013.