Miguel García Bosque

Digital Design of New Chaotic Ciphers for Ethernet Traffic

Departamento

Ingeniería Electrónica y Comunicaciones

Director/es

Celma Pueyo, Santiago
Sánchez Azqueta, Carlos

ISSN 2254-7606

ISSN 2254-7606

Tesis Doctoral

DIGITAL DESIGN OF NEW CHAOTIC CIPHERS FOR ETHERNET TRAFFIC

Autor

Miguel García Bosque

Director/es

Celma Pueyo, Santiago
Sánchez Azqueta, Carlos

UNIVERSIDAD DE ZARAGOZA
Ingeniería Electrónica y Comunicaciones

2019

DIGITAL DESIGN OF NEW CHAOTIC CIPHERS FOR ETHERNET TRAFFIC

Miguel García Bosque

Grupo de Diseño Electrónico – I3A
Departamento de Ingeniería Electrónica y Comunicaciones
Facultad de Ciencias
Universidad de Zaragoza

DIGITAL DESIGN OF NEW CHAOTIC CIPHERS FOR ETHERNET TRAFFIC

A thesis submitted to the University of Zaragoza
in partial fulfillment of the requirements for
the Degree of Doctor of Philosophy in Physics

by
Miguel García Bosque

Thesis Supervisors
Dr. Santiago Celma Pueyo
Dr. Carlos Sánchez Azqueta

Electronic Design Group – I3A
Department of Electronic Engineering and Communications
Faculty of Science – University of Zaragoza

Zaragoza, September 2019

DIGITAL DESIGN OF NEW CHAOTIC CIPHERS FOR ETHERNET TRAFFIC

Tesis presentada a la Universidad de Zaragoza
para optar al grado de
Doctor en Física

por
Miguel García Bosque

bajo la supervisión de los Doctores
Dr. Santiago Celma Pueyo
Dr. Carlos Sánchez Azqueta

Grupo de Diseño Electrónico – I3A
Departamento de Ingeniería Electrónica y Comunicaciones
Facultad de Ciencias - Universidad de Zaragoza

Zaragoza, Septiembre 2019

A mi familia.

Agradecimientos

En primer lugar, me gustaría agradecer a mis directores, Dr. Santiago Celma y Dr. Carlos Sánchez-Azqueta, por todo su apoyo, su paciencia y sus consejos. Por compartir conmigo su conocimiento y su entusiasmo por la física y la electrónica. Gracias por haber estado siempre disponibles para echarme una mano cuando lo he necesitado.

A todas las personas que durante estos años han formado parte del GDE, por conseguir que el trabajo diario, el que no queda recogido en este documento, haya sido mucho más fácil y divertido.

También quiero agradecer a mi supervisor durante mis estancias en la Universidad de Siena, Dr. Tommaso Addabbo, por toda la ayuda prestada. Además, me gustaría agradecer a todo el grupo de investigación por haberme hecho la estancia mucho más llevadera.

Gracias a toda mi familia y amigos, por haber estado conmigo en todo momento y por hacerme merecedor de vuestro tiempo. Gracias por todas las cosas bonitas que me habéis dicho y por el apoyo que me habéis dado.

Por último, agradezco también a todas las instituciones que han financiado la investigación recogida en este trabajo: Ministerio de Economía y Competitividad del Gobierno de España, Fondo Europeo de Desarrollo Regional, Gobierno de Aragón y Horizon 2020 a través de los proyectos (TEC2014-52840-R, RTC-2015-3807-7, A38790, T26-17R y TEC2017-85867-R) y Ministerio de Educación a través del contrato predoctoral FPU (FPU14/03523). También agradecer a la *Associazione di Fondazioni e di Casse di Risparmio S.p.A*. y a la *Circuits and Systems Society* por financiar mi segunda estancia a través del programa YITP.

Contents

List of Figures

List of Tables

List of Symbols

a_i, a_i'	Elements (bits) of a binary sequence
A, A'	Binary sequence
ASE	Average Shannon Entropy
$\mathrm{ASE_k}$	Partial Average Shannon Entropy
b_k	kth byte
b_k	kth byte bit
β	Set of bits
c, c'	Ciphertext
c_i	ith bit or symbol of the ciphertext
$c(t)$	Analog masked signal (ciphertext)
C, C_i	Ciphertext of the size of an encryption block
d_k	Decryption function
DC_i	Deficit counter
e_k	Encryption function
f	Feedback/state-update function
F	Flag
g	Extraction function
h	Function that combines plaintext and ciphertext
h	Hash function
k	Key
K	Cipher bank
l	Integer number
m	Transmitted message
n	Integer number

p, p'	Plaintext
p_i	ith bit or symbol of the plaintext
$p(t)$	Analog message signal (plaintext)
P, P_i	Plaintext of the size of an encryption block
Pr	Probability
Q_i	Quantum
r	Integer number
R_j	Correlations
s_i	Element of the sequence partition
$\{s_i\}$	Sequence partition
t_i	ith time step
ΔT	Sampling period
T	Period of a sequence
U, V	Metric space
x_i	State of the system at ith iteration/time step (one dimension)
X_i	State of the system at ith iteration/time step (multiple dimensions)
z	Keystream
z_i	ith bit or symbol of the keystream
$z(t)$	Analog masking signal
β	Set of bits
$\mathcal{C}$	Set of possible ciphertexts
δ	Arbitrarily small distance
$\mathcal{E}$	Set of all encryption functions (rules) $e_k : \mathcal{P} \rightarrow \mathcal{C}$
γ	Control parameter
Γ	Set of control parameters
κ	Key space size

$\mathcal{K}$	Keyspace (set of possible keys)
λ	Lyapunov exponent
$\mathcal{P}$	Set of possible plaintexts
τ_d	Delay

List of Acronyms

AES	Advanced Encryption Standard
APN	Almost Perfect Nonlinear
ASE	Average Shannon Entropy
ASIC	Application-Specific Integrated Circuit
ASR	Average Shannon Redundancy
BER	Bit-Error-Rate
CA	Certification Authority
CBC	Cipher Block Chaining
CDR	Clock and Data Recovery
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMOS	Complementary Metal-Oxide-Semiconductor
COOK	Chaos On-Off Keying
CRC	Cyclic Redundancy Check
CSK	Chaos Shift Keying
CTR	Counter
DCSK	Differential Chaos Shift Keying
DES	Data Encryption Standard
DNO	Digital Nonlinear Oscillators
DWRR	Deficit Weighted Round Robin
ECB	Electronic Codebook
ENISA	European Union Agency for Network and Information Security
FIFO	First In First Out
FIPS	Federal Information Processing Standard

FM-DCSK	Frequency Modulated Differential Chaos Shift Keying
FPGA	Field Programmable Gate Array
HDDCS	Higher-Dimensional Chaotic Systems
HMAC	Hash-Based Message Authentication Code
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IFG	Interframe Gap
ISO	International Organization for Standardization
IV	Initialization Vector
LAN	Local Area Networks
LFSR	Linear Feedback Shift Register
LM	Logistic Map
LSB	Least Significant Bit
LUT	Lookup Table
MA	
MAC	Message Authentication Code/Medium Access Control
MAN	Metropolitan Area Networks
MEMS	Microelectromechanical
MLE	Maximal Lyapunov Exponent
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MSB	Most Significant Bit
NAND	Not-And
NIST	National Institute of Standards and Technology
NLD	Nonlinear Dynamic Forecasting
OFB	Output Feedback
PCB	Printed Circuit Board

PCS	Physical Coding Sublayer
PHY	Physical Layer
PLD	Programmable Logic Device
PMA	Physical Medium Attachment
PMD	Physical Medium Dependent
PMOD	Peripheral Module
PRBS	Pseudorandom Binary Sequence
PRNG	Pseudorandom Number Generator
RAM	Random Access Memory
RNG	Random Number Generator
RSA	Rivest Shamir Adleman
RTL	Register-Transfer Level
RX	Receiver
SERDES	Serializer/Deserializer
SFP	Small Form-Factor Pluggable
SHA	Secure Hash Algorithm
STM	Skew Tent Map
TRNG	True Random Number Generator
TSMC	Taiwan Semiconductor Manufacturing Company
TX	Transmitter
WAN	Wide Area Networks

1

Introduction

This introduction starts with a brief overview of modern cryptography, pointing out some of the greatest achievements carried out in the last century in this field. In Section 1.2, a formal definition of cryptography as well as some basic concepts that will be used during the whole Thesis are presented. Section 1.3 will cover the state of the art as well as some important research lines. This section includes some of the issues that current cryptosystems present, and present some of the alternatives that have been proposed. Based on the study of the state of the art, the objectives will be presented in Section 1.4. Finally, the Thesis organization is presented in Section 1.5.

1.1. Brief history of modern cryptography

Traditionally, cryptography has been intimately linked to military and diplomatic applications. The usage of cryptography outside of military circles, can be dated at the end of World War II. In those years, communications suffered several changes, and the encryption and decryption processes had to adapt to those changes. In that sense, it can be highlighted that the information started to be transmitted in chains of bits instead of words of a certain language. Furthermore, with the development of computers, there was a crescent number of people that needed to protect the transmitted information, as well as the stored data.

During the 1970's decade, cryptography suffered two major public advances. The first one was the development of the Data Encryption Standard (DES) and its publication as an official Federal Information Processing Standard (FIPS) for the United States [NAT77]. This algorithm, became the standard used by most of the commercial applications.

The second advance, was the proposal of a new key-exchange protocol by Whitfield Diffie and Martin Hellman [DIF76]. Up to that point, all the ciphers were symmetric key algorithms, which needed both transmitter and receiver to share the same secret key. This key had to be exchanged between the communicating parties in some secure way, such as face-to-face contact or using a secure channel. Thanks to Diffie and Hellman's work keys could be exchanged in a secure way and was the precursor of a new class of encryption algorithms: public-key algorithms. Some of the principal ones are Rivest Shamir Adleman (RSA) [RIV78] and ElGamal [ELG85].

One of the most important contributions of public-key cryptosystems were the digital signatures, whose first international standard (ISO/IEC 9796:1991, nowadays ISO/IEC 9796-2:2010) was adopted in 1991.

More recently, in 1997, the National Institute of Standards and Technology (NIST) announced the necessity of replacing the DES algorithm with a new one, "an unclassified publicly disclosed encryption algorithm capable of protecting sensitive government information well into the next century". After a selection process, Rijndael algorithm [DAE98] was chosen, becoming the Advanced

Encryption Standard (AES), which is the most commonly used encryption algorithm nowadays.

1.2. Definition

The main objective of cryptography is to allow two people to communicate over an insecure channel but preventing an attacker from obtaining the information contended in the message. Often, in cryptography explanations, the person that sends the information is called Alice and the person who receives the message is called Bob. For the attacker, usually the names Mallory or Eve are used (in this Thesis, the name Mallory will be used). The information that Alice wants to send to Bob is called plaintext. To prevent Mallory from obtaining the information, Alice encrypts the plaintext using a key and sends it to Bob. The encrypted message is called ciphertext. This way, even if Mallory manages to eavesdrop the communication, she cannot determine what the plaintext was. However, Bob knows the key needed to decrypt the message and can reconstruct the plaintext.

A cryptosystem is the set of algorithms that allows Alice and Bob to exchange information confidentially. Usually, cryptosystems consist on 3 algorithms: one for key generation, one for encryption and one for decryption. Generally, the term *cipher* is used to refer to the pair of algorithms used for encryption and decryption while the term cryptosystem includes everything. However, in many cases it is common to use the terms “cipher” and “cryptosystem” [KAT14] interchangeably. These ideas can be described mathematically as follows.

A cryptosystem is a five-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where the following conditions are satisfied [STI05]:

- $\mathcal{P}$ is a finite set of possible plaintexts.
- $\mathcal{C}$ is a finite set of possible ciphertexts.
- $\mathcal{K}$, the keyspace, is a finite set of possible keys.
- $\mathcal{E} = \{e_k : k \in \mathcal{K}\}$ is the set of all encryption functions (rules) $e_k : \mathcal{P} \to \mathcal{C}$.
- $\mathcal{D} = \{d_k : k \in \mathcal{K}\}$ is the set of all decryption functions (rules) $d_k : \mathcal{C} \to \mathcal{P}$.
- For each $k \in \mathcal{K}$, there is an encryption function $e_k \in \mathcal{E}$ and a corresponding decryption function $d_k \in \mathcal{D}$ such that $d_k(e_k(p)) = p$ for any plaintext element $p \in \mathcal{P}$.

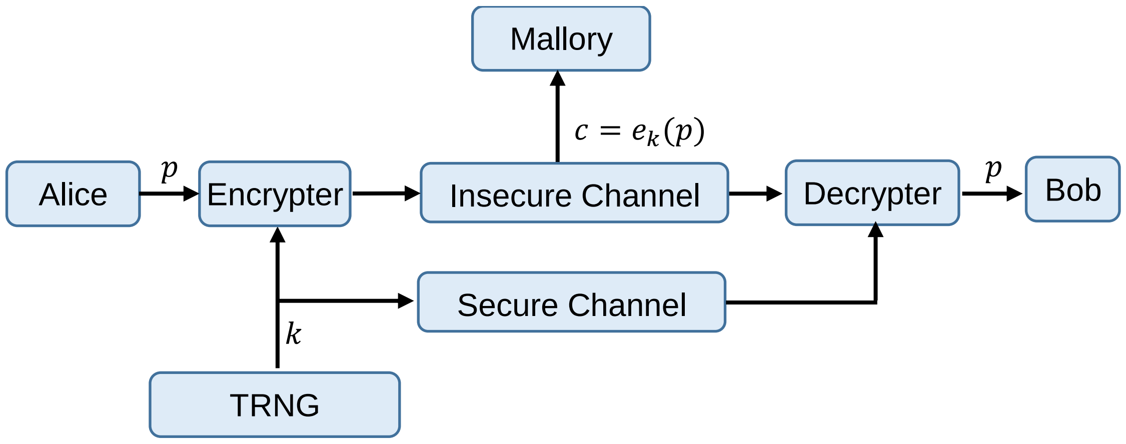


Fig. 1.1. Basic scheme of a communication process over an insecure channel.

With this cryptosystem definition, if Alice and Bob wanted to establish a communication over an insecure channel, the process would be as follows:

- Alice and Bob agree on the communication protocol (the set of encryption and decryption rules) and choose a random key $k \in \mathcal{K}$. While the encryption and decryption rules can be known by anyone, the key should be kept in secret. Therefore, the key exchange should be done secretly, when they are not being observed by Mallory, or via a secure channel.
- When Alice wants to send a message to Bob over an insecure channel, Alice uses the encryption rule e_K corresponding to the chosen key k.
- Bob receives the ciphertext and decrypts it using the decryption rule d_k corresponding to the chosen key k.

This process has been illustrated in Fig. 1.1.

It is clear that the encryption function must be injective (one-to-one) since, otherwise, the decryption cannot be accomplished unambiguously, i.e.: $p_1 \neq p_2 \Rightarrow$

$\Rightarrow e_k(p_1) \neq e_k(p_1)\ \forall k \in \mathcal{K}, p_1, p_2 \in \mathcal{P}$.

Apart from the encryption and decryption functions, there are other functions that can be used in security-related applications. Usually, the term cryptographic primitives is used to refer to any low-level algorithm used to build cryptographic protocols. This term includes the encryption and decryption algorithms, but there are other important cryptographic functions that will be explained in the next chapter.

1.3. State of the art and research lines

During the last decades, there has been a great development in the field of cryptography, and many encryption algorithms as well as other cryptographic functions have been proposed.

However, despite this development, there has been a great interest in the last years in creating new cryptographic primitives or improving the current ones. Some of the reasons why it is necessary are the following:

- First, with the development of the communication technologies, the amount of information transmitted is constantly increasing. In this context, there are many applications that require to encrypt a high amount of data in real time or in a limited amount of time. A simple example, could be high-quality video encryption. Unfortunately, most of the current encryption algorithms are not able to encrypt high amounts of data at a high speed, while maintaining high security standards.
- Due to the development of computer technology, many algorithms that have traditionally been considered secure, can be now brute-forced in a reasonable amount of time. As an example, DES was first released with a key size of only 56 bits while, for current symmetric cryptographic algorithms, NIST recommends that they should have, at least, key sizes of 112 bits [BAR11]. On the other hand, there is currently a big amount of research in the area of quantum computing and it is expected that, at some point, large-scale quantum computers will be developed. It has been proven that, some of the current encryption algorithms such as RSA could be easily broken by quantum computers [SHO94a].
- Along with the development of cryptography, there has been a big development in the field of mathematical cryptanalysis. Therefore, new attacks are constantly being proposed and new vulnerabilities are constantly being found in some of the current encryption algorithms. Thus, new algorithms robust to all kind of attacks must be proposed to substitute the vulnerable ones. On the other hand, the security of some of the most

commonly used public-key encryption algorithms such as RSA or ElGamal, are based on the assumption that some mathematical problems such as factoring the product of two large primes or computing discrete logarithms are difficult to solve. However, there is no proof that, in the future, new algorithms capable of solving these problems fast (in a polynomial time) could be developed.

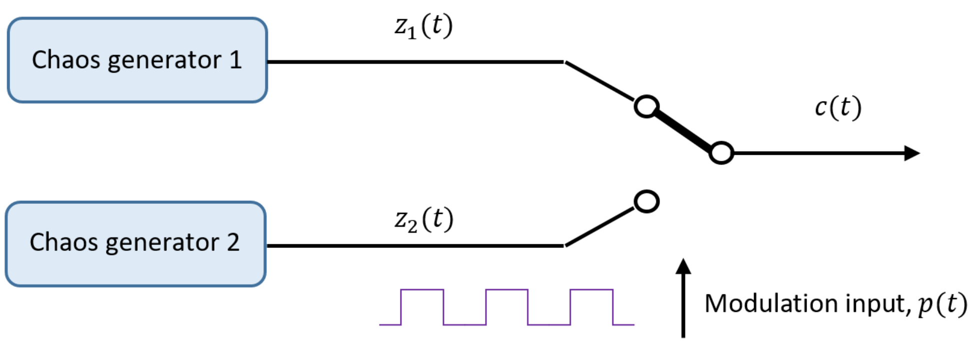
- Ideally, the keys used to encrypt data should be randomly generated to be as unpredictable as possible. Since the sequences generated by a Pseudo Random Number Generator (PRNG) are, in principle, predictable, they are potentially vulnerable to cryptanalysis. Therefore, the keys are usually generated using True Random Number Generators (TRNGs). Unfortunately, typically TRNGs offer slower bit generation rates than PRNGs and usually provide sequences with non-ideal statistical properties that need to be improved using some kind of post-processing techniques. A poor TRNG can compromise the security of a system even if it uses a secure encryption algorithm, as it has happened in some cases [BER13]. Finding a TRNG fast and with good statistical properties is an important research topic.

To sum up, it is clear that the field of cryptography is a hot research topic with numerous research lines. Since the field of cryptography is too extensive to be covered in a Thesis, this work has focused on three particular research lines: the design of new TRNGs, the design of new chaos-based synchronous stream ciphers for secure fast communications and, finally, the implementation of new cryptosystems suitable for Optical Gigabit Ethernet. These last cryptosystems should include both a TRNG for generating the keys and a chaos-based ciphering algorithm. Below, the antecedents and state of the art in the area of chaos-based cryptography, TRNGs and Optical Gigabit Ethernet encryption are presented.

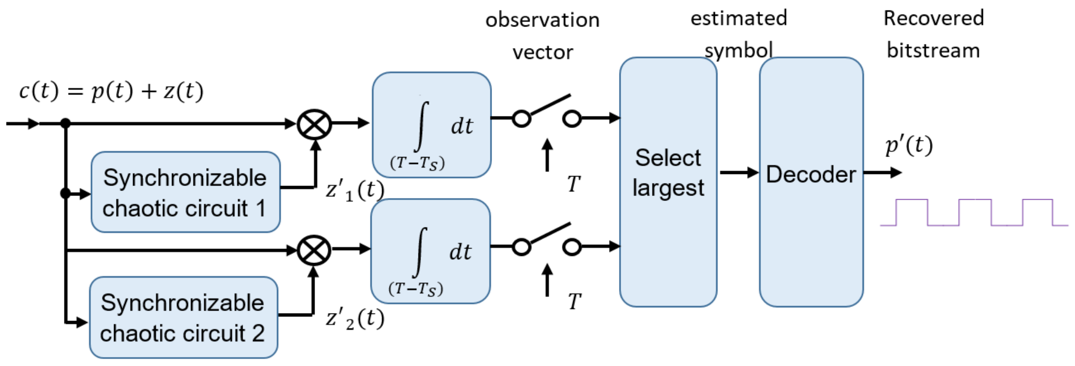
1.3.1. Chaos-based ciphers

The origin of synchronizable (analog) chaos-based cryptography starts in 1990, when Louis M. Pecora and Thomas L. Carroll proved that certain chaotic systems could synchronize with each other by linking them to common signals [PEC90]. Soon after this discovery, the usage of chaos for secure communication was proposed in numerous papers in the following years [CAR91, PEC92, CUO92a, CUO92b, KOC92, PAR92, HAL93, WU93]. Some of the first proposed encryption techniques were:

- Chaotic masking [KOC92, WU93, CUO93a, CUO93b]. In this case, in the transmitter, the analog message signal, $p(t)$, is added to a chaotic signal, $z(t)$. The masked signal, $c(t) = p(t) + z(t)$ is then sent to the receiver. As long as the message signal is much smaller than the chaotic signal ($p(t) \ll z(t)$), the receiver can generate an approximate synchronous chaotic signal, $z'(t) \approx z(t)$. When this synchronous chaotic signal is subtracted from the encrypted signal, the message signal is approximately recovered $p'(t) = c(t) - z'(t) \approx p(t)$.
- Chaos shift keying (CSK) [DED93, PAR92]. In this case, each symbol is used to choose among several different chaotic circuits (Fig. 1.2a). The symbols are then detected at the receiver by using cascaded synchronizable chaotic circuits (Fig. 1.2b). Some other methods based on CSK that have been proposed are chaos on-off keying (COOK), differential chaos shift keying (DCSK) and frequency modulated differential chaos shift keying (FM-DCSK). A survey of these methods is presented in [KEN00].



(a)



(b)

Fig. 1.2. (a) CSK transmitter. (b) CSK receiver.

- Chaotic modulation [HAL93, CUO93a, YAN96]. In this case, the message signal, $p(t)$, is used to modulate a parameter of the chaotic generator. In these systems, the receiver can exactly synchronize with the transmitter even when the message is incorporated.
- Chaos control methods [HAY93, HAY94]. The double scroll attractor in Chua's circuit consists of a random-like sequence of positive and negative peaks, each peak can be associated to a 1 or a 0. By using small control perturbations is it possible to cause the signal to follow an orbit whose binary sequence represents the information we wish to communicate. The same approach can be used with other chaotic systems [LAI99].
- Inverse system approach [FEL96, ZHO97b]. The information signal, $p(t)$, controls a chaotic system, the transmitter. The output of the transmitter, $c(t)$ is the input of a receiver which has to retrieve the information signal. In order to do this, the receiver has an input-output relation inverse to that of the transmitter.

Unfortunately, most of these methods were soon proved to be insecure and several methods to attack these systems were proposed. The first method that was proposed, nonlinear dynamic forecasting (NLD), was capable of extracting the carrier signal, $z(t)$, for the chaotic masking and some chaotic modulation schemes [SHO94, SHO96, SHO97, SHO98, PAR01]. The message signal, $p(t)$, could then trivially be recovered by removing the carrier signal, $z(t)$, from the transmitted ciphertext signal, $c(t)$.

Other methods could directly extract the message signal. Some of these methods were power spectral analysis [YAN98a], return map analysis [PER95, ZHO97], [YAN98b], correlation analysis [ZHO97], generalized synchronization technique [YAN98c] or short-time period [YAN95].

Finally, other methods were proposed to try to estimate the parameters needed for decryption [ZHO97, DED97, ZHO99, GED99].

Despite that many new synchronization-based cryptosystems were proposed in the following years to resist these attacks, new attacks were proposed and were able to successfully break most of these of systems [LI05, ALV06].

Due to this fact, most of the research (including the one in this Thesis) carried out nowadays focuses on digital chaos-based cryptosystems. In these systems, the chaotic system is digitized and implemented in hardware or software using a finite precision. Apart from the fact that they usually offer higher security than synchronization-based communication systems, these systems are easier to be implemented in current communication standards where the plaintext and ciphertext are expected to be digital signals and the most commonly used ciphering algorithms (RSA, RC4, Trivium, AES, …) are all digital.

The first papers that proposed digital ciphers with dynamical systems were based on cellular automata and proposed in 1985 and 1987 [WOL85, GUA87]. However, the first paper usually cited as the first chaos-based cryptosystem was proposed by Robert A. J. Matthews in 1989 [MAT89]. After that paper, digital chaotic ciphers attracted the attention of many cryptographers and many papers were published in a short period of time, proposing new chaos-based PRNGs and stream ciphers [BER90, FOR91, BER91, BIA91], block ciphers [HAB90, HAB91, BIH91] and a few public-key ciphers [GUA87, DEL91, FRE93]. Almost at the same time, some papers [WHE91a, WHE91b, BIH91, ERD92, AND92] proved the vulnerabilities of most of those systems and, since then, the number of papers published in this area dropped.

Despite this fact, new digital chaos-based cryptosystems have been continuously being proposed and they have constantly improved in terms of encryption speed, power consumption, area and security. Although some chaotic algorithms have been proposed to be used in public-key cryptosystems (e.g. public-key ciphering based on Chebyshev polynomials [KOC05, KOC11]), this Thesis will focus on symmetric algorithms. The main reason is that the ultimate purpose of this Thesis is to implement in hardware a chaos-based cryptosystem suitable for Optical Gigabit Ethernet communications. For the same reason, this Thesis will only focus on hardware oriented algorithms so software oriented algorithms will not be considered.

In this area, several cryptosystems have been proposed and implemented in the last years, achieving promising results. As the chaotic encryption algorithms, most of them are based on the Logistic Map (LM) although some modifications are usually introduced for improving the security. Some of the best performing algorithms based on the LM are [CHE10a, CHE10b, LI10, DAB11, DAB14, LI12, PAN13]. Other chaotic algorithms that have been used for designing secure cryptosystems are the Rényi Map [ADD07], the Lorenz's attractor [AZZ09], or higher-dimensional chaotic systems (HDDCS) [WAN16a]. In some cases, several chaotic algorithms are implemented and compared in the same work. This is the case of [DAB12] that compares the implementations of the Logistic Map and the Henon map or [GIA12] that compares implementations of the Bernoulli, Chebyshev, Tent and Cubic maps. Recently, chaotic iterations have been proposed to post-process linear pseudorandom number generators. Unfortunately, most of these algorithms are not fast enough for Gigabit Ethernet or have not passed a strict security analysis [BAK16, BAK18].

1.3.2. True Random Number Generators

The best and most complete TRNGs solutions that have been proposed in literature (and on the market as well) so far, have been designed in ASICs [ACO17]. These generators, can be classified according to the source of randomness that it is used:

- Jitter oscillation [TAN14, YAN14, YAN16]. These generators use as a source of randomness the deviation of an oscillator output from its true periodicity, causing uncertainty in the low-high/high-low transition times.
- Electronic noise. In this case, the conventional method is to amplify noise with a high-gain and high-bandwidth amplifier followed by quantization. Some of the common noise sources are thermal noise [PET00], oxide trap noise [BRE06], SiN device noise [MAT08] or oxide breakdown noise [LIU11].
- Metastability [TOK07, MAT12]. When there are setup or hold time violations in flip-flops they enter in a metastable state. These generators exploit the unpredictability of the metastable states as a source of randomness.
- Finally, some generators use several entropy sources such as [BAE17] which uses both jitter oscillation and metastability or [KUA14] that uses both electronic noise and metastability.

Besides ASIC design, researchers are exploring the design of TRNGs in Programmable Logic Devices (PLDs). Compared to ASICs, PLD-TRNGs present a great advantage in terms of cost and versatility. Usually, the sources of randomness used by these systems are the same as the ones explained before: thermal noise [DAN09], metastability [DAN09, HAT12] or jitter oscillation [GOL06, SUN07, WOL08, WAN16b, LIU17]. Unfortunately, PLD-TRNGs still suffer from a lack of trust in information security communities, mainly due to some major cryptographic weaknesses found in solutions based on ring oscillators [BAU11, RAI15].

Finally, there is a possibility of using the noise generated by a sensor to generate random numbers. The main advantage of this approach is that it is possible to reuse a sensor (temperature, acceleration, pressure, …) that it is already present in a device to use it as a TRNG. Although some studies about this possibility have been made [VOR11, WAL16] and some sensor-based TRNGs have been proposed [HON15, REV17], this approach has been relatively unexplored.

To sum up, although some quite complete TRNG solutions have been proposed for ASIC, in our opinion, the potential of PLDs or sensors for the design of reliable and efficient TRNGs is far from being completely explored. Due to the fact that, as explained above, both of these approaches present some advantages against ASIC-TRNGs, we have decided to focus our research on these kind of generators.

1.3.3. Optical Gigabit Ethernet

Ethernet has been expanded widely in local area networks (LAN), metropolitan area networks (MAN) and wide area networks (WAN). In recent decades, it has also been used in industrial control systems and critical infrastructures, replacing the traditional communication field buses [SAU11a, DEP08].

Optical Ethernet is widely used since it has some advantages over other wired methods such as higher bandwidth, less signal losses, and more immunity to electromagnetic interference. In addition, due to the fact that it does not emit radiation, it is safer than wireless systems that are more exposed to eavesdropping.

However, vulnerability and threat analysis in the physical layer (PHY) of optical systems is critical to guarantee secure communications [SKO16, FUR14]. At present, low-cost method for intercepting the optical signal through the fiber coupling devices and optoelectronic converters are available without the need to perceptibly interfere in communications (splitting attack) [ZAF11]. To avoid or detect eavesdropping, encryption and intrusion detection systems have been proposed as solutions [SKO16].

In a layered communication model, encryption methods can be implemented at different communication levels, depending on the communication layer where confidentiality is needed. In the particular case of industrial Ethernet, solutions are usually proposed for network and transport layers (layers 3 and 4), such as IPsec or transport layer security protocols [SAU11b, BHA06]. Other solutions are proposed for the data link layer (layer 2), such as MACsec standard [LAZ17]. Although some protocols have been proposed for physical layer (layer 1) encryption in some telecommunication networks [ELK13, JI17, GUA16], as far as we know, no solutions have been proposed for optical Ethernet networks.

An encryption at a physical layer would bring some advantages against encryption at other layers such minimum latency and line rate (zero overhead). As an example, in IPsec, the inherent overhead introduced during encryption reduces the overall throughput between 20% and 90% of the maximum achievable [TRO05], so the improvement achieved by an encryption at a physical layer would be considerable. Finally, another advantage is that, by performing the encryption at a physical layer, obfuscation of customer data traffic patterns can be achieved. Since all of these advantages would result in an overall improvement over the state of the art, we intend to implement an encryption system that works at the physical layer (1000Base-X in the case of 1 Gb Ethernet and 10GBase-R in the case of 10 Gb Ethernet).

As far as we know, no chaos-based cryptosystems have been proposed as an Ethernet solution so far. On the other hand, both IPsec [SAU11b] and MACsec [LAZ17] protocols indicate the ciphering algorithm as well as the key exchange process, but they do not specify how the keys should be generated. Therefore, designing a chaos-based cryptosystem that includes a good TRNG used to generate the keys would undoubtedly be a significant contribution to the state of the art.

1.4. Objectives

It has been seen that most of the chaos-based cryptosystems proposed so far cannot achieve enough speed for Gigabit Ethernet communications, require a big amount of area for being implemented or lack of a strict security analysis. In addition to that, most of the encryption algorithms (both chaotic and non-chaotic) proposed so far do not address the key generation process. Ideally, a TRNG should be used for key generation. This generator should not only be capable of generating random sequences but it should also be fast and robust (i.e., it should always present good randomness properties regardless of any external interference). However, no complete solutions of TRNGs for PLDs or TRNGs based on the noise generated by sensors have been proposed so far, despite the advantages that these approaches would present against ASIC-TRNGs.

The scope of this work is the design of new secure physical layer encryption schemes suitable for Optical Gigabit Ethernet traffic. In order to design and implement a full communication scheme, we will design new secure chaos-based encryption algorithms as well as new TRNGs suitable for key generation. Finally, in order to encrypt at a physical layer, the algorithms will have to be adapted to preserve the data coding.

To design new TRNGs suitable for key generation, two possibilities will be studied. One consists of the usage of the noise generated by a Microelectromechanical (MEMS) accelerometer while, the other one will be based on the jitter generated by digital nonlinear oscillators (DNOs). For the last case, the proposed structures will be tested using an Arty board, which includes an Artix-7 Field Programmable Gate Array (FPGA). This board includes a set of ports that will be helpful for measuring the proposed structures. Xilinx Vivado Design Suite will be used for the implementations as well as to perform behavioral, post-synthesis and post-implementation simulations. Once implemented, to carry out the measurements, this FPGA will be connected to a computer using a LabVIEW platform.

The proposed encryption algorithms will be first implemented in a Virtex 7 FPGA, which is faster than the Artix 7 FPGA. Once the designs have been implemented and tested in the FPGA, we intend to select one of the encryption algorithms and implement it in an Application-Specific Integrated Circuit (ASIC) using Cadence. A 0.18-μm CMOS process will be used for this implementation.

For the implementation of the Optical Gigabit Ethernet encryption system, we will also use the FPGA. In the setup for test, the FPGA will be connected to two Small From-Factor Pluggable (SFP+) modules capable of transmitting at a rate of 10.3125 Gbps at 850 nm over multimode fiber.

The concrete objectives of this work are summarized below:

- Study of the state of the art, including the encryption algorithms that are currently being used. This part will analyze the main issues that current standard encryption algorithms present and what solutions, if any, have been proposed. This study will help us to design new algorithms that overcome these issues.
- Proposal of new TRNGs suitable for key generation. We will explore two different possibilities: the usage of the noise generated by a MEMS accelerometer and the jitter generated by Digital Nonlinear Oscillators (DNOs). Both cases will be analyzed in detail by performing several statistical analyses at different sampling frequencies. If necessary, a simple post-processing algorithm will be proposed and implemented to improve the randomness of the generated sequences. Finally, the possible usage of these TRNGs as key generators will be discussed.
- Proposal of new encryption algorithms that are fast, secure and can be implemented using a small amount of resources. Among all the possibilities, this work will focus on cryptosystems based on chaotic since, thanks to their intrinsic properties such as ergodicity or random-like behavior, they can be a good alternative to classical encryption. To overcome some of the issues that appear when these systems are digitized, several strategies will be studied: using a multi-encryption scheme, changing the chaotic control parameters and perturbing the chaotic orbits.
- Implementation of the proposed encryption algorithms. For this purpose, a Virtex 7 FPGA and Vivado Design Suite will be used. The different implementations will be tested and compared, and some important aspects such as their power consumption, area usage, throughput and security will be discussed. One of these designs will be selected and implemented in an

ASIC using a 0.18-µm technology. Nevertheless, the solutions that we will look for will be generic so they will also be able to be implemented in other platforms or other technologies.

- Finally, the proposed algorithms will be adapted and applied to Optical Gigabit Ethernet communications and will use one of the proposed TRNG to generate the keys. In particular, we intend to implement and analyze cryptosystems that work at the physical layer for 1 Gb and 10 Gb Ethernet communications. In order to perform the encryption at the physical layer, these algorithms will have to be adapted to preserve the data coding, 8b/10b in the case of 1 Gb Ethernet and 64b/66b in the case of 10 Gb Ethernet. In both cases, the cryptosystems will be implemented on a Virtex 7 FPGA and an experimental setup, including two SFP modules capable of transmitting at a rate up to 10.3125 Gbps at 850 nm over multimode fiber, will be design. With this setup we will check that the encryption systems work correctly and synchronously without harming data traffic or link establishment between Ethernet interfaces. Furthermore, we will check that the encryption is good (i.e., passes all the security tests) and that the data traffic pattern is hidden.

1.5. Thesis organization

This Thesis is divided into six chapters, of which the first one is devoted to this introduction and the last one presents the general conclusions of the work. The remaining four chapters form the core of the work that has been carried out during the whole Thesis.

Chapter 1 is an introduction that includes a brief overview of the field of cryptography, the antecedents of this work, its motivation and the main objectives.

Chapter 2 includes an extensive theoretical background with all the concepts and definitions needed to understand this work. First, it gives a general overview of the whole protocol needed to transmit information confidentially and explains some of the most important cryptographic primitives. Second, it explains the most important aspects of chaos based cryptography with its advantages and the current state of the art. Third, it explains the main aspects of TRNGs and their applications in cryptography. Finally, this chapter presents the randomness tests and cryptanalysis that must be used to evaluate the security of both TRNGs and the encryption algorithms.

Chapter 3 explains the basic scheme of a chaos-based stream cipher, its advantages, and its drawbacks. Furthermore, it presents several chaos-based stream ciphers that have been proposed. All of them are based on simple known chaotic maps, such as the Skew Tent Map or the Logistic Map, but include some techniques to improve their security.

Chapter 4 presents two different TRNGs that have been proposed and analyzed in this work. The first one, uses the thermal noise produced by a MEMS accelerometer at rest while, the second one, consists on a family of TRNGs based on nonlinear oscillators that can be implemented in CPLs.

Chapter 5 applies the proposed encryption algorithms for Optical Gigabit Ethernet communications. Both 1 Gb and 10 Gb Ethernet communications systems are proposed, implemented and analyzed in this section.

Finally, in Chapter 6, the conclusions of the work carried out in this Thesis as well as possible future research lines are presented.

1.6. References

[ADD07] T. Addabbo, M. Alioto, A. Fort, A. Pasini, S. Rocchi, V. Vignoli, “A Class of Maximum-Period Nonlinear Congruential Generators Derived from the Rènyi Chaotic Map,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 4, pp. 816-828, 2007.

[ALV06] G. Alvarez, S. Li, “Some Basic Cryptography Requirements for Chaos-Based Cryptosystems,” *International Journal of Bifurcation and Chaos*, vol. 16, no. 8, pp. 2129-2151, 2006.

[AND92] R. Anderson, "Letter to the Editor: Chaos and Random Numbers," *Cryptologia*, vol. 16, no. 3, pp. 226, 1992.

[AZZ09] M. S. Azzaz, C. Tanougast, S. Sadoudi, A. Dandache, "Real-Time FPGA Implementation of Lorenz’s Chaotic Generator for Ciphering Telecommunications," *Proceedings of the IEEE International Circuits and Systems and Taisa Conference*, 2009.

[BAE17] S.-G. Bae, Y. Kim, Y. Park, C. Kim, "3-Gb/s High-Speed True Random Number Generator Using Common-Mode Operating Comparator and Sampling Uncertainty of D Flip-Flop," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 2, pp. 605-610, 2017.

[BAK16] M. Bakiri, J.-F. Couchot, C. Guyeux, “FPGA Implementation of F2-Linear Pseudorandom Number Generators Based on Zynq MPSoc: A Chaotic Iterations Post Processing Case Study,” *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications*, pp. 302-309, 2016.

[BAK18] M. Bakiri, C. Guyeux, J.-F. Couchot, L. Marangio, S. Galatolo, “A Hardware and Secure Pseudorandom Generator for Constrained Devices,” *IEEE Transactions on Industrial Informatics,* vol. 14, no. 8, pp. 3754-3765, 2018.

[BAR11] E. Barker, A. Roginsky, “Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths,” *NIST Special Publication 800-131 A*, 2011.

[BAU11] M. Baudet, D. Lubiez, J. Micolod, A. Tassiaux, “On the Security of Oscillator-Based Random Number Generators,” *Journal of Cryptology,* vol. 24, no. 2, pp. 398-425, 2011.

[BER90] G. M. Bernstein, M. A. Lieberman, “Secure Random Number Generation Using Chaotic Circuits,” *IEEE Transactions on Circuits and Systems*, vol. 37, no. 9, pp. 1157-1164, 1990.

[BER91] G. M. Bernstein, M. A. Lieberman, “Method and Apparatus for Generating Secure Random Numbers Using Chaos,” *US Patent No. 5008087*, 1991.

[BER13] D. J. Bernstein, Y. Chang, C. Cheng, L. Chou, N. Heninger, T. Lange, N. van Someren, “Factoring RSA Keys from Certified Smart Cards: Coppersmith in the Wild,” *Advances in Cryptology – ASIACRYPT*, pp. 341-360, 2013.

[BHA06] D. V. Bhatt, S. Schulze, G. P. Hancke, “Secure Internet Access to Gateway Using Secure Socket Layer,” *IEEE Transactions on Instrumentation and Measurements*, vol. 55, no.3, pp. 793-800, 2006.

[BIA91] M. E. Bianco, D. A. Reed, “Encryption System Based on Chaos Theory,” *US Patent No. 5048086*, 1991.

[BIH91] T. Habutsu, Y. Nishio, I. Sasage, S. Mori, “Cryptanalysis of the Chaotic-Map Cryptosystem Suggested at EuroCrypt’91,” *Advances in Cryptology – EuroCrypt’91,* pp. 532-534, 1991.

[BRE06] R. Brederlow, R. Prakash, C. Paulus, R. Thewes, “A Low-Power True Random Number Generator Using Random Telegraph Noise of Single Oxide-Traps,” *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC). Digest of Technical Papers,* pp. 1666-1675, 2006.

[CAR91] T. M. Carroll, L. M. Pecora, “Synchronizing Chaotic Circuits,” *IEEE Transactions on Circuits and Systems*, vol. 38, pp. 453-456, 1991.

[CHE10a] S.-L. Chen, T. Hwang, W.-W. Lin, “Randomness Enhancement Using Digitalized Modified Logistic Map,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 12, pp. 996-1000, 2010.

[CHE10b] S.-L. Chen, T. Hwang, S.-M. Chang, W.-W. Lin, “A Fast Digital Chaotic Generator for Secure Communication,” *International Journal of Bifurcation and Chaos*, vol. 20, no. 12, pp. 3969-3987, 2010.

[CUO92a] K. M. Cuomo, A. V. Oppenheim, S. H. Isabelle, “Spread Spectrum Modulation and Signal Masking Using Synchronized Chaotic Systems,” *MIT Technical Report*, no. 570, 1992.

[CUO92b] K. M. Cuomo, A. V. Oppenheim, “Synchronized Chaotic Circuits and Systems for Communications,” *MIT Technical Report*, no. 575, 1992.

[CUO93a] K. M. Cuomo, A. V. Oppenheim, “Circuit Implementation of Synchronized Chaos with Applications to Communications,” *Physical Review Letters*, vol. 71, no. 1, pp. 65-68, 1993.

[CUO93b] K. M. Cuomo, A. V. Oppenheim, S. H. Strogatz, “Synchronization of Lorenz-Based Chaotic Circuits with Applications to Communications,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 40, no. 10, pp. 626-633, 1993.

[DAB11] P. Dabal, R. Pelka, “A Chaos-Based Pseudo-Random Bit Generator Implemented in FPGA Device,” *Proceedings of the International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pp. 151-154, 2011.

[DAB12] P. Dabal, R. Pelka, “FPGA Implementation of Chaotic Pseudo-Random Bit Generators,” *Proceedings of the 19th International Conference on Mixed Design of Electronics Circuits and Systems (MIXDES)*, pp. 260-264, 2012.

[DAB14] P. Dabal, R. Pelka, “A Study on Fast Pipelined Pseudo-Random Number Generator Based on Chaotic Logistic Map,” *Proceedings of the 17th International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pp. 195-200, 2014.

[DAE98] J. Daemen, V. Rijmen, "AES Proposal: Rijndael," *AES submission*, 1998.

[DEL91] J.-P. Delahaye, “Les Automates (in French),” *Pour la Science (French Edition of Scientific American)*, pp. 126-134, 1991.

[DEP08] A. Depari, P. Ferrari, A. Flammini, D. Marioli, A. Taroni, “A New Instrument for Real-Time Ethernet Performance Measurement,” *IEEE Transactions on Instrumentation and Measurements*, vol. 57, no. 1, pp. 121-127, 2008.

[DIF76] W. Diffie, M. E. Hellman, “New Directions in Cryptography,” *IEEE Transactions on Information Theory*, vol. it-22, no. 6, pp. 644-654, 1976.

[ELG85] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Transactions on Information Theory*, vol. IT-31, pp. 469-472, 1985.

[ELK13] D. Elkouss, J. Martinez-Mateo, A. Ciruana, V. Martin, "Secure Optical Networks Based on Quantum Key Distribution and Weakly Trusted Repeaters," *Journal of Optical Communications and Networking*, vol. 5, no. 4, pp. 316-328, 2013.

[ERD92] D. Erdmann, S. Murphy, "Henon Stream Cipher," *Electronics Letters*, vol. 28, no. 9, pp. 893-895, 1992.

[HON15] S. L. Hong, C. Liu, "Sensor-Based Random Number Generator Seeding," *IEEE Access*, vol. 3, pp. 562-568, 2015.

[HWU93] F. Hwu, "The Interpolating Random Spline Cryptosystem and the Chaotic-Map Public-Key Cryptosystem," *PhD thesis, Faculty of the Graduate School, University of Missouri*, 1993.

[FOR90] R. Forré, "The Hénon Attractor as a Keystream Generator," *Abstracts of Eurocrypt 91*, pp. 469-472, 1990.

[FUR14] M. Furdek, N. Skorin-Kapov, S. Zsigmond, L. Wosinska, "Vulnerabilities and Security Issues in Optical Networks," *Proceedings of the International Conference on Transparent Optical Networks (ITCON),* pp. 1-4, 2014.

[GIA12] P. Giard, G. Kaddoum, F. Gagnon, C. Thibeault, "FPGA Implementation and Evaluation of Discrete-Time Chaotic Generators Circuits," *Proceedings of the IEEE 38th Annual Conference of the Industrial Electronics Society (IECON 2012)*, pp. 3221-3224, 2012.

[GOL06] J. D. Golíc, "New Methods for Digital Generation and Postprocessing of Random Data," *IEEE Transactions on Computers*, vol. 55, no. 10, pp. 1217-1229, 2006.

[GUA87] P. Guan, "Cellular Automaton Public-Key Cryptosystem," *Complex Systems*, vol. 1, pp. 51-57, 1987.

[GUA16] K. Guan, J. Kakande, J. Cho, "On Deploying Encryption Solutions to Provide Secure Transport-as-a-Service (TaaS) in Core and Metro Networks," *Proceedings of the European Conference on Optical Communications*, pp. 1-3, 2016.

[HAB90] T. Habutsu, Y. Nishio, I. Sasage, S. Mori, "A Secret Key Cryptosystem Using a Chaotic Map," *Transactions IEICE*, vol. 73, no. 7, pp. 1041-1044, 1990.

[HAB91] T. Habutsu, Y. Nishio, I. Sasage, S. Mori, "A Secret Key Cryptosystem by Iterating a Chaotic Map," *Advances in Cryptology – EuroCrypt'91*, pp. 127-140, 1991.

[HAL93] K. S. Halle, C. W. Wu, M. Itoh, L. O. Chua, "Spread Spectrum Communications through Modulation of Chaos," *International Journal of Bifurcation and Chaos*, vol. 3, no. 2, pp. 469-477, 1993.

[JI17] J. Ji, G. Zhang, W. Li, L. Sun, K. Wang, M. Xu, "Performance Analysis of Physical-Layer Security in an OCDMA-Based Wiretap Channel," *Journal of Optical Communications and Networking.*, vol. 9, no. 10, pp. 813-818, 2017.

[KAT14] J. Katz, Y. Lindell, "Introduction to Modern Cryptography," *Chapman & Hall/CRC*, 2014.

[KEN00] M. P. Kennedy, G. Kolumbán, "Digital Communications Using Chaos," *Signal Processing*, vol. 80, no. 7, pp. 1307-1320, 2000.

[KOC92] L. J. Kocarev, K. S. Halle, K. Eckert, L. O. Chua, U. Parlitz, "Experimental Demonstration of Secure Communications Via Chaotic Synchronization," *International Journal of Bifurcation and Chaos*, vol. 2, no. 3, pp. 709-713, 1992.

[KOC05] L. J. Kocarev, J. Makraduli, "Public-Key Encryption Based on Chebyshev Polynomials," *Circuits Systems Signal Processing*, vol. 24, no. 5, pp. 497-517, 2005.

[KOC11] L. J. Kocarev, S. Lian, "Chaos-Based Cryptography: Theory, Algorithms and Applications," *Springer-Verlag*, Berlin Heidelberg.

[KUA14] T. K. Kuan, Y. H. Chiang, S. I. Liu, "A 0.43pJ7bit True Random Number Generator," *2014 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pp. 33-36, 2014.

[LAI99] Y.-C. Lai, E. Bolt, C. Grebogi, "Communicating with Chaos Using Two-Dimensional Symbolic Dynamics," *Physics Letters A*, vol. 255, no. 1-2, pp. 75-81, 1999.

[LAZ17] J. Lázaro, A. Astarloa, J. Araujo, N. Moreira, U. Bidarte, “MACsec Layer 2 Security in HSR Rings in Substation Automation Systems,” *Energies*, vol. 10, no. 2, pp. 162, 2017.

[LI05] S. Li, “Chaotic Cryptography (1): Analog Chaos-Based Secure Communications,” *Invited lecture, Department of Physics, Beijing Normal University, Beijing, China*, 2005.

[LI10] C.-Y. Li, T.-Y. Chang, C.-C. Huang, “A Nonlinear PRNG Using Digitized Logistic Map with Self-Reseeding Method,” *Proceedings of the 2010 International Symposium on VLSI Design, Automation and Test*, 2010.

[LI12] C.-Y. Li, Y.-H. Chen, T.-Y. Chang, L.-Y. Deng, K. To, “Period Extension and Randomness Enhancement Using High-Throughput Reseeding-Mixing PRNG,” *IEEE Transactions on VLSI Systems*, vol. 20, no. 2, 2012.

[LIU11] N. Liu, N. Pinckney, S. Hanson, D. Sylvester, D. Blaauw, “A True Random Number Generator Using Time-Dependent Dielectric Breakdown,” *Proceedings of the 2011 Symposium on VLSI Circuits – Digest of Technical Papers*, pp. 216-217, 2011.

[LIU17] Y. Liu, R. C. C. Cheung, H. Wong, “A Bias-Bounded Digital True Random Number Generator Architecture,” *IEEE Transactions on Circuits and Systems I*, vol. 64, no. 1, pp. 133-144, 2017.

[MAT89] R. A. J. Matthews, “On the Derivation of a “Chaotic” Encryption Algorithm,” *Cryptologia*, vol. 13, no. 1, pp. 29-41, 1989.

[MAT08] M. Matsumoto, S. Yasuda, R. Ohba, K. Ikegami, T. Tanamoto, S. Fujita, “1200 μm2 Physical Random-Number Generators Based on SiN MOSFET for Secure Smart-Card Application,” *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC). Digest of Technical Papers*, pp. 414-624, 2008.

[MAT08] S. K. Matthew et al., “2.4 Gbps 7mW All-Digital PVT-Variation Tolerant True Random Number Generator for 45 nm CMOS High-Performance Microprocessors,” *IEEE Journal of Solid-State Circuits*, vol. 47, no. 11, pp. 2807-2821, 2012.

[NAT77] National Bureau of Standards (U.S), Federal Information Processing Standards Publication 46, “Data Encryption Standard (DES),” 1977.

[PAN13] A. Pande, J. A. Zambreno, “A Chaotic Encryption Scheme for Real-Time Embedded Systems: Design and Implementation,” *Telecommunication Systems,* vol. 52, no. 2, pp. 515-561, 2013.

[PAR92] U. Parlitz, L. O. Chua, L. J. Kocarev, K. S. Halle, A. Shang, “Transmission of Digital Signals by Chaotic Synchronization,” *International Journal of Bifurcation and Chaos*, vol. 2, no. 4, pp. 973-977, 1992.

[PAR01] A. T. Parler, K. M. Short, “Reconstructing the Keystream from a Chaotic Encryption Scheme,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 48, no. 5, pp. 624-630, 2001.

[PEC90] L. Pecora, T. Carroll, “Synchronization in Chaotic Systems,” *Physical Review Letters,* vol. 64, no. 8, pp. 821-825, 1990.

[PEC92] L. M. Pecora, T. M. Carroll, “Synchronized Chaotic Signals and Systems,” *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP-92)*, pp. 137-140, 1992.

[PET00] C. S. Petrie, J. A. Connelly, “A Noise-Based IC Random Number Generator for Applications in Cryptography,” *IEEE Transactions on Circuits and Systems I: Express Briefs,* vol. 47, no. 5, pp. 615-621, 2000.

[RAI15] M. Raitza, M. Vogt, C. Hochberger, T. Pionteck, “Raw 2014: Random Number Generators on FPGAs,” *ACM Transactions on Reconfigurable Technology and Systems,* vol. 9, no. 2, pp. 15:1-15:21, 2015.

[REV17] G. Revadigar, C. Javali, W. Xu, A. V. Vasilakos, W. Hu, S. Jha, “Accelerometer and Fuzzy Vault-Based Secure Group Key Generation and Sharing Protocol for Smart Wearables,” *IEEE Transactions on Information Forensics and Security,* vol. 12, no. 10, pp. 2467-2482, 2017.

[RIV78] R. Rivest, A. Shamir, L. Adleman, “A Method for Obtaining Digital Signatures and Public Key Cryptosystems,” *Communications of the ACM,* vol. 21, no. 2, pp. 120-126, 1978.

[SAU11a] T. Sauter, S. Soucek, W. Kastner, D. Dietrich, “The Evolution of Factory and Building Automation,” *IEEE Industrial Electronics Magazine,* vol. 5, no.3, pp. 35-48, 2011.

[SAU11b] T. Sauter, M. Lobashov, “How to Access Factory Floor Information Using Internet Technologies and Gateways,” *IEEE Tranactions on Industrial Informatics,* vol. 7, no.4, pp. 699-712, 2011.

[SHO94a] P. W. Shor, “Algorithms for Quantum Computation: Discrete Logarithms and Factoring,” *Proceedings of the 35th Annual Symposium On Foundations of Computer Science,* pp. 124-134, 1994.

[SHO94b] K. M. Short, “Steps Toward Unmasking Secure Communications,” *International Journal of Bifurcaction and Chaos,* vol. 4, no. 4, pp. 959-977, 1994.

[SHO96] K. M. Short, “Unmasking a Modulated Chaotic Communications Scheme,” *International Journal of Bifurcaction and Chaos,* vol. 6, no. 2, pp. 367-375, 1996.

[SHO97] K. M. Short, “Signal Extraction from Chaotic Communications,” *International Journal of Bifurcaction and Chaos,* vol. 7, no. 7, pp. 1579-1597, 1997.

[SHO98] K. M. Short, A. T. Parker, “Unmasking a Hyperchaotic Communication Scheme,” *Physical Review E,* vol. 58, no. 1, pp. 1159-1162, 1998.

[SKO16] N. Skorin-Kapov, M. Furdek, S. Zsigmond, L. Wosinska, “Physical-Layer Security in Evolving Optical Networks,” *IEEE Communications Magazine,* vol. 54, no. 8, pp. 110-117, 2016.

[STI05] D. Stinson, “Cryptography: Theory and Practice,” *CRC Press, Boca Raton, 3th edition*, 2005.

[SUN07] B. Sunar, W. J. Martin, D. R. Stinson, “A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks,” *IEEE Transactions on Computers*, vol. 56, no. 1, 2007.

[TAN14] Q. Tang, B. Kim, Y. Lao, K. K. Parhi, C. H. Kim, “True Random Number Generator Circuits Based on Single and Multi-Phase Beat Frequency Detection,” *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference (CICC),* pp. 1-4, 2014.

[TOK07] C. Tokunaga, D. Blaauw, T. Mudge, “True Random Number Generator with a Metastability-Based Quality Control,” *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC). Digest of Technical Papers,* pp. 404-405, 2007.

[TRO05] L. Troell, J. Burns, K. Chapman, D. Goddard, M. Soder-Lund, C. Ward, “Converged vs. Dedicated IPSec Encryption Testing in Gigabit Ethernet Networks,” *Rochester Institute of Technology, Technical Report 1743,* Available online: http://scholarworks.rit.edu/article/1743 (accessed on 30 June 2019), 2005.

[VOR11] J. Voris, D. Saxena, T. Halevi, “Accelerometers and Randomness: Perfect together,” *Proceedings of the 4th ACM Conference on Wireless Network Security,* pp. 115-126, 2011.

[WAL16] K. Wallace, K. Moran, E. Novak, G. Zhou, K. Sun, “Toward Sensor-Based Random Number Generation for Mobile and IoT Devices,” *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1189-1201, 2016.

[WAN16a] Q. Wang, S. Yu, C. Li, J, Lü, X. Fang, C. Guyeux, J. M. Bahi“Theoretical Design and FPGA-Based Implementations of Higher-Dimensional Digital Chaotic Systems,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 10, pp. 302-309, 2016.

[WAN16b] Y. Wang, S. Li, “A High-Speed Digital True Random Number Generator Based on Cross Ring Oscillator,” *IEEE Transactions on Fundamentals of Electronics Communications and Computer Science*, vol. E99, no. 4, pp. 806-818, 2016.

[WHE91a] D. D. Wheeler, R. A. J. Matthews, “Supercomputer Investigations of a Chaotic Encryption Algorithm,” *Cryptologia*, vol. 15, no. 2, pp. 140-151, 1991.

[WHE91b] D. D. Wheeler, R. A. J. Matthews, “Problems with Mitchell’s Nonlinear Key Generators,” *Cryptologia*, vol. 15, no. 4, pp. 355-363, 1991.

[WOL85] S. Wolfram, “Cryptography with Cellular Automata,” *Advances in Cryptology – Crypto’85*, pp. 429-432, 1985.

[WOL08] K. Wold, C. H. Tan, “Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings,” *2008 International Conference on Reconfigurable Computing FPGAs*, pp. 385-390, 2008.

[WU93] C. W. Wu, L. O. Chua, “A Simple Way to Synchronize Chaotic Systems with Applications to Secure Communications,” *International Journal of Bifurcation and Chaos*, vol. 3, no. 6, pp. 1619-1627, 1993.

[YAN95a] T. Yang, “Recovery of Digital Signals from Chaotic Switching,” *International Journal of Circuit Theory and Applications*, vol. 23, no. 6, pp. 611-615, 1995.

[YAN98a] T. Yang, L. B. Yang, C. M. Yang, “Breaking Chaotic Secure Communications Using a Spectrogram,” *Physics Letters A*, vol. 247, no. 1-2, pp. 105-111, 1998.

[YAN98b] T. Yang, L. B. Yang, C. M. Yang, “Cryptanalyzing Chaotic Secure Communications Using Return Maps,” *Physics Letters A*, vol. 245, no. 6, pp. 495-510, 1998.

[YAN98c] T. Yang, L. B. Yang, C. M. Yang, “Breaking Chaotic Switching Using Generalized Synchronization,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 45, no. 10, pp. 1062-1067, 1998.

[YAN14] K. Yang, D. Fick, M. B. Henry, Y. Lee, D. Blaauw, D. Sylvester, “A 23 Mb/s 23 pJ/b Fully Synthesized True-Random-Number Generator in 28 nm and 65 nm CMOS,” *Proceedings of the 2014 IEEE International Solid-State Circuits Conference (ISSCC). Digest of Technical Papers,* pp. 280-281, 2014.

[YAN16] K. Yang, D. Blaauw, D. Sylvester, “An All-Digital Edge Racing True Random Number Generator Robust Against PVT Variation,” *IEEE Journal of Solid State Circuits,* vol. 51, no. 4, 2016.

[ZAF11] M. Zafar, H. Fathallah, N. Belhadj, “Optical Fiber Tapping: Methods and Precautions,” *Proceedings of the 8th International Conference on High-Capacity Optical Networks and Emerging Technologies (HONET)*, pp. 164-168, 2011.

[ZHO97a] C.-S. Zhou, T.-L. Chen, “Extracting Information Masked by Chaos and Contaminated with Noise. Some Considerations on the Security of Communication Approaches using Chaos,” *Physics Letters A*, vol. 234, no. 6, pp. 429-435, 1997.

[ZHO97b] C.-S. Zhou, X. Ling, “Problems with the Chaotic Inverse System Encryption Approach,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 44, no. 3, pp. 268-271, 1997.

2

Important Concepts and Definitions

In this chapter, the main concepts and definitions that will be used in this Thesis are presented. Section 2.1 explains a general overview of a secure communication system, describing the main cryptographic primitives and how these can be used to design a secure communication protocol. Section 2.2 will focus on chaos-based cryptography, explaining some important properties of chaotic systems and its applications to secure communications. In particular, this section will focus on how to design and analyze a chaos-based stream cipher. Finally, Section 2.3 provides some of the statistical tests that can be used to compare non-ideal random number generators and will present some possible post-processing techniques that can be used to improve their statistical properties.

2.1. Overview of a secure communication system

The purpose of this section is to explain the different types of ciphers (symmetric and asymmetric) as well as other important cryptographic primitives. The ultimate scope of this section is to explain how these cryptographic primitives can be combined to construct a secure communication system.

2.1.1. Symmetric and asymmetric ciphers

According to how the keys are distributed, cryptosystems can be divided in two different types [MEN97]:

- **Symmetric or private key cryptosystems**.
  In these systems, the same key is shared by both the sender and the receiver and is used for encryption and decryption. To guarantee the security of these systems, before the communication, the sender and the receiver must have secretly shared the key. These systems, can be divided in two different types: block ciphers and stream ciphers.
- **Asymmetric or public key cryptosystems**.
  In these systems, the receiver creates two keys: a public key and a private one. The public key is used for encrypting the message and can be known by anyone while the private key is used for decrypting the message and can be only known by the receiver. To guarantee the security, it should be infeasible to obtain the private key from the public key.

2.1.1.1. Block ciphers

Within the private key algorithms, block ciphers transform a block (a fixed-length group of bits), $P,$ into another block of the same size, C, using an encryption algorithm, e_k, that depends on a key k. The decryption algorithm, d_k, performs the inverse operation, i.e., $d_k(e_k)(P) = P \;\; \forall k \in \mathcal{K}, \; \forall p \in \mathcal{P}$. If we call n the size of the block and l the size of the key, the block cipher can be specified by an encryption function:

$$e_k(P) \coloneqq e(k,P) : \{0,1\}^l \times \{0,1\}^n \to \{0,1\}^n \tag{2.1}$$

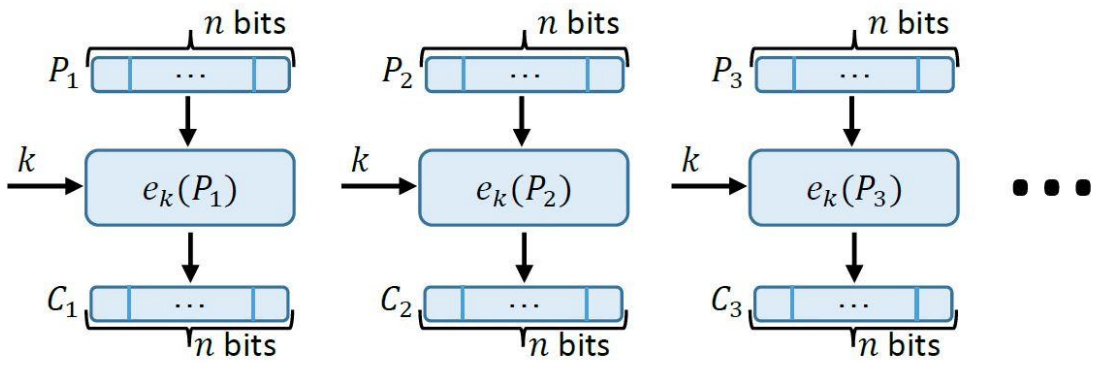
and a decryption function:

$$e_k^{-1}(C) \coloneqq d_k(C) = d(k,C) : \{0,1\}^l \times \{0,1\}^n \to \{0,1\}^n \tag{2.2}$$

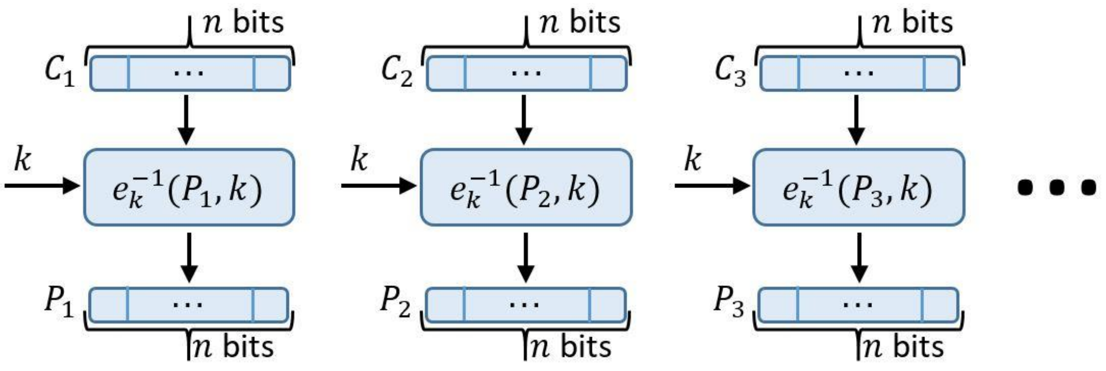
Usually, bigger block sizes offer higher security but they also increase the complexity of the systems (they are more costly to implement). Typical block sizes (n) used in modern cryptosystems are 64, 128, 192, and 256 bits [BUC17].

In these systems, it must be pointed out that the set of possible input blocks, *P*, and the set of possible encrypted blocks, *C*, is usually the same. Furthermore, as explained in Section 1.2, different ciphertexts should correspond to different plaintexts to avoid any ambiguity. Therefore, each encryption e_k or decryption function d_k is actually a permutation over the set of input blocks (i.e., a bijective mapping). In other words, each key is generated selecting a permutation over the $(2^n)!$ possible permutations.

Usually, the sender does not want to encrypt a single block but a variable-length message instead. In this case, the message needs to be partitioned first into separate blocks. If the length of the message is not a multiple of *n*, the last block needs to be completed with extra bits (padding bits). In the simplest operation mode, called Electronic Codebook (ECB), each block is encrypted (Fig. 2.1a) and decrypted (Fig. 2.1b) one by one.



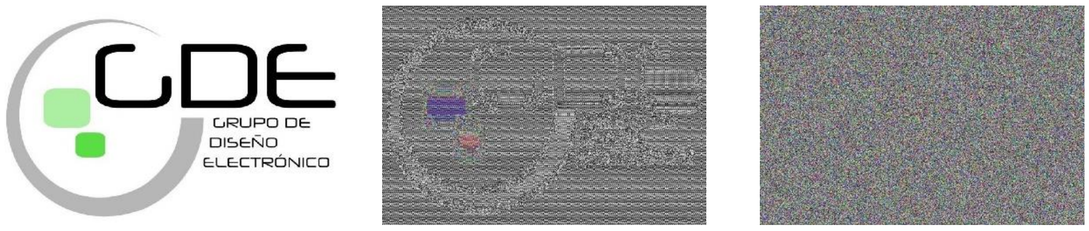
(a)



(b)

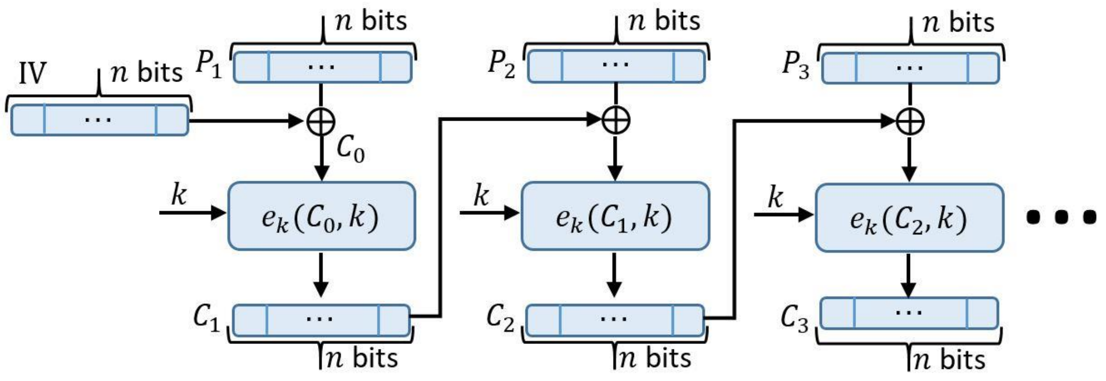
Fig. 2.1. Electronic Codebook (ECB) mode for (a) encryption and (b) decryption.

However, this method has the problem that if two blocks of the message are the same and are encrypted with the same key, their corresponding ciphertexts will be identical so patterns in the message can be detected. A simple example of this problem can be seen when a block cipher in ECB mode is used to encrypt a bitmap image that uses large areas of uniform color (Fig. 2.2a). While each individual color is encrypted, the image can still be recognized since areas of uniform color still have a uniform color after encryption (Fig. 2.2b).

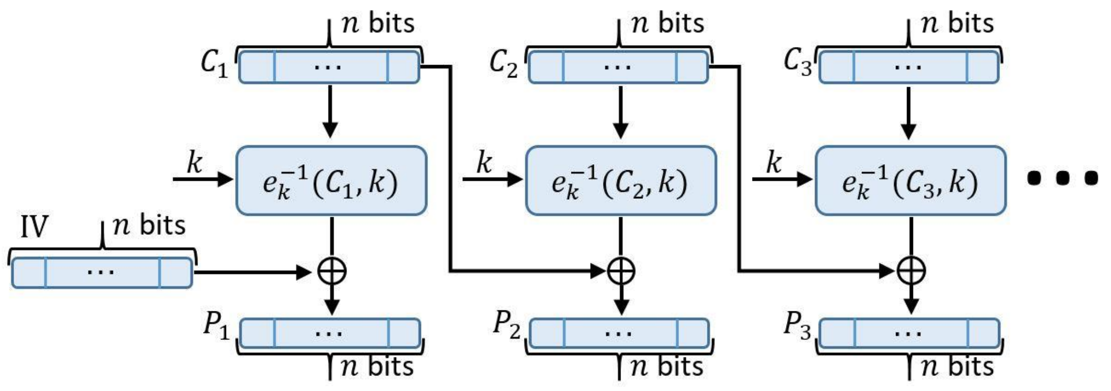


(a) (b) (c)

Fig. 2.2. (a) Original Image. (b) Image encrypted with ECB. (c) Image encrypted with CBC.



(a)



(b)

Fig. 2.3. Cipher Block Chaining (CBC) mode for (a) encryption and (b) decryption.

To prevent this, block ciphers often use more complex modes of operation. Basically, these modes of operation randomize the plaintext using an additional input value called initialization vector (IV). Some of the most common operation modes are: Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB) and Counter (CTR). A scheme of the CBC operation mode is shown in Fig. 2.3, while a full description of these modes of operation as well as some recommendations regarding them can be found in [DWO01]. As an example, in Fig. 2.2c, it can be seen that CBC can achieve a better encryption than ECB.

2.1.1.2. Stream ciphers

The other type of symmetric cryptosystems are the stream ciphers. In this case, using a key and an encryption algorithm, the sender generates a sequence of random bits named keystream. The ciphertext is created by combining each bit of the plaintext with its corresponding bit of the keystream. For decrypting the

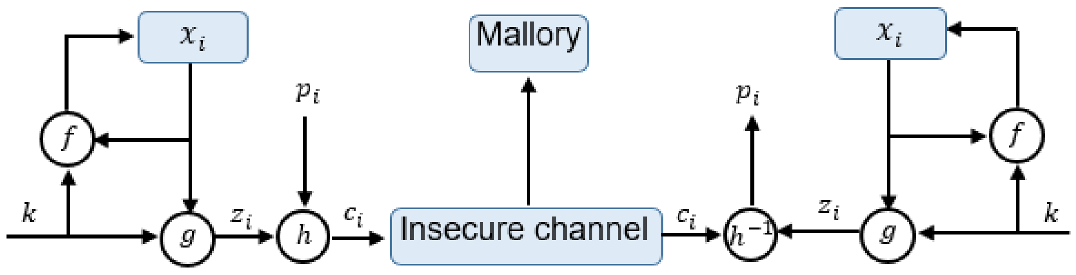


Fig. 2.4. Synchronous stream cipher.

message, the receiver must use the key (that must have been secretly exchanged) and the same algorithm to generate an identical keystream. By doing the inverse combining operation, the original plaintext is recovered. Stream ciphers can also be classified in two types, depending on how they generate the keystream. If the keystream only depends on the key, the stream cipher is synchronous while, if the ciphertext is also used by the algorithm to generate the keystream, the stream cipher is asynchronous. The term synchronous refers to the fact that, in order to work properly, the keystreams must be synchronized (aligned) between the encrypter and the decrypter. On the other hand, asynchronous ciphers do not need this synchronization.

In general, a synchronous stream cipher (Fig. 2.4) has the form:

$$\begin{aligned} x_{i+1} &= f(x_i, k) \\ z_i &= g(x_i, k) \\ c_i &= h(z_i, p_i) \end{aligned} \tag{2.3}$$

where k is the key, x_i is the internal state at the time step i, z_i is the ith bit (or symbol) of the keystream, p_i is the ith bit (or symbol) of the plaintext, c_i is the ith bit (or symbol) of the ciphertext, f is a feedback function, g is the extraction function and h is a function that combines the keystream with the plaintext.

Usually, the h function is an XOR operation $(c_i = z_i \oplus p_i)$ and the feedback (f) and extraction (g) functions do not depend on the key $(x_{i+1} = f(x_i),\ z_i = g(x_i))$. In these cases the ciphers are called binary additive stream ciphers. Furthermore, although z_i, p_i, c_i, can be a bit, a group of bits, or a more complex symbol (e.g., a letter of the alphabet), they are often just a single bit. Therefore, from now on, to simplify the writing, we will refer to them as single bits.

It must be pointed out that synchronous stream ciphers do not guarantee the integrity of the message since, if there is a transmission error or a deliberate attack and the ith bit of the ciphertext, c_i, is changed, the ith bit of the original plaintext, p_i, would differ from the ith bit the plaintext recovered by the receiver, p_i'. Since there is a change only in a single bit, this error could easily be unnoticed. For example, let's imagine that the message was an order to transfer 14000€ and this amount was represented by a 16-bit number (e.g., 0011011010110000). If an attacker knew that the ith bit of the ciphertext corresponds to the Most Significant Bit (MSB) of the amount to transfer, he could change it from '0' to '1', so, instead of 14,000 €, the transferred amount would be 46,768 € (1011011010110000). This kind of attacks where an attacker change the data are called active attacks. Therefore, to transmit information over a noisy channel or to prevent this kind of attacks, a certain protocol must be used (a possible secure communication protocol will be explained in Section 2.1.3).

On the other hand, an asynchronous (also called self-synchronizing) stream cipher has the form:

$$\begin{aligned} x_i &= (c_{i-1}, \dots c_{i-t}) \\ z_i &= g(x_i, k) \\ c_i &= h(z_i, p_i) \end{aligned} \tag{2.4}$$

where k is the key, x_i is the internal state at the time step i, z_i is the ith bit (or symbol) of the keystream, p_i is the ith bit (or symbol) of the plaintext, c_i is the ith bit (or symbol) of the ciphertext, g is an extraction function and h is a function that combines the keystream with the plaintext (see Fig. 2.5). Just like in synchronous stream ciphers, in asynchronous stream ciphers h function is usually an XOR operation $(c_i = z_i \oplus p_i)$ and the extraction function g does not depend on the key. As it can be seen, in this case the keystream depends on the key, but also on a fixed number of preceding ciphertext bits (or symbols). The initial internal state, x_0, can be just a randomly chosen set of bits but can also depend on the key.

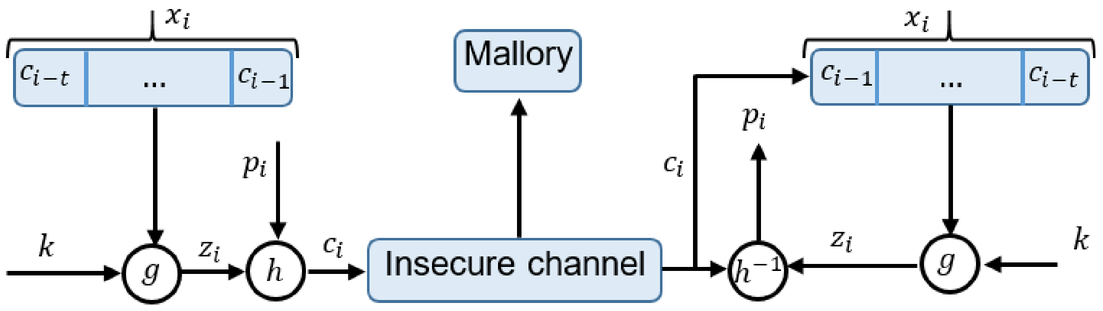


Fig. 2.5. A self-synchronizing stream cipher.

Asynchronous stream ciphers can have some advantages over synchronous ones:

- In these systems, if a bit of the ciphertext is changed, the error propagates so several bits of the recovered plaintext would also change, often resulting in a nonsense message. Therefore, it is easier to detect errors or active attacks.
- If a bit is removed or an extra bit is injected in the ciphertext, only a finite number of bits of the recovered plaintext will change before the cipher stabilizes again (self-synchronizing property).
- Since all the bits of the plaintext have influence on the following bits of the ciphertext, the statistical properties of the plaintext are dispersed through the ciphertext so these systems could be more secure against some attacks based on the redundancy in the plaintext.

Although these properties are useful, an important disadvantage of asynchronous stream ciphers is that they are typically harder to design and implement. This is due to the fact that, while in synchronous stream ciphers the keystream generation process and the encryption process are separated, in asynchronous stream ciphers both processes are mixed, making the implementation more difficult [KLE13]. Furthermore, due to this separation in the keystream generation process and the encryption process, the security analysis of synchronous stream ciphers is easier to make, since both processes can be studied independently. Therefore, although asynchronous stream ciphers might be more secure, it is more difficult to prove it and there is a higher risk that some security holes remain undetected after the security analysis. For these reasons, most of the stream ciphers used nowadays are synchronous stream ciphers [KLE13].

2.1.1.3. Block ciphers vs stream ciphers

The main advantage of block ciphers against stream ciphers is that they are better understood than stream ciphers. The main reason is that, in block ciphers, it is easy to split the problem in several modules. For example, it is possible to study the operation mode without paying attention to the ciphering functions. Furthermore, many block ciphers such as DES and AES consist of several identical rounds, and in each round several functions are applied consecutively.

Therefore, it is possible to study the effects of a particular Almost Perfect Nonlinear (APN) function in a certain round. On the other hand, in the case of stream ciphers, it is difficult to modularize the problem and sometimes the keystream generation interacts with the key scheduling in a complicated way.

On the other hand, since stream ciphers encrypt each bit individually, they do not need to store groups of bits in blocks and they do not need to add padding bits. This results in lower memory requirements and higher encryption speeds.

Therefore, if there are not any special requirements, it is advisable to use a standard block cipher such as AES. However, in applications that need to save gates or energy or might need to encrypt data at high speed, stream ciphers are necessary.

2.1.1.4. Public-key algorithms

As it can be seen, in both block ciphers and stream ciphers, before starting the communication, both Alice and Bob must have secretly shared a key. This can be a big challenge since, often, the sender and the receiver might not be able to stablish a face-to-face contact or use a secure channel to share the secret key. Asymmetric cryptosystems, also known as public-key cryptosystems, are capable of overcoming this problem. As already mentioned at the beginning of Section 2.1.1, these cryptosystems use a pair of keys for the communication process, both of them generated by the receiver. One of them is public (i.e., anyone could know it) and is used to encrypt the plaintext while the other one is private (i.e., only known by the receiver) and is needed to decrypt the ciphertext. These systems are based on a certain kind of functions called one-way functions. A one-way function can be informally described as any function f that presents these properties [WEI02]:

- The description of f is publicly known and does not require any extra information for its operation.
- Given an x in the domain of f, it is easy to calculate $f(x)$.
- Given z an y in the range of f, it is hard to find an x such as $f(x) = y$.

Note that, in this definition, the terms “easy” and “hard” refer to computational time (“easy” when it can be computed fast and “hard” when it is infeasible to be computed in a reasonable amount of time). More precisely, an operation is called "easy" when it exists an algorithm that can find the solution in a polynomial time (P problem) and "hard" when it does not exist an algorithm that can find the solution in a polynomial time (NP problem). It must be noted that the existence of such functions has not yet been proven. If it was true, it would imply that $P \neq NP$, which is currently a very important unsolved problem in computer science [COO71]. Nevertheless, it is assumed that there exist some one-way functions that can be used in the field of cryptography.

Public-key cryptosystems need a particular kind of one-way functions called trapdoor one-way functions $\{e_k : k \in \mathcal{K}\}$. These functions have the peculiarity that, with some secret information (trapdoor), k', it is easy to invert e_k. A public-key cryptosystem has the same components $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, explained in the general definition (Section 1.2) but with these particularities:

- In this case, $\mathcal{K}$ is the public-key space, consisting on a finite set of possible keys used for encryption.
- Each encryption rule e_k is a trapdoor one-way function.
- There is a procedure for generating a random public key $k \in \mathcal{K}$ along with a trapdoor k' for e_k and the inverse map $d_k : \mathcal{C} \to \mathcal{P}$ such that $d_k\left(e_k(p)\right) = p \ \forall p \in \mathcal{P}, \forall k \in \mathcal{K}$

The set of trapdoors, $'$, is the private-key space. Therefore, the communication process in public-key cryptosystems consists of the following steps:

1. The receiver generates a random public key $k \in \mathcal{K}$ and its corresponding trapdoor k' (private key).
2. The receiver places the encryption function e_k in a public directory, keeping the private key k' and the decryption function d_k for himself.
3. The sender encrypts the plaintext using the encryption function. Therefore, the ciphertext, c, is given by $c = e_k(p)$.
4. The receiver deciphers the plaintext using the decryption function: $p = d_k(c) = d_k(e_k(p))$

It must be pointed out that the fact that e_k is a one-way function implies that it is impossible to obtain the private key k' from the public key k.

These cryptosystems, however, have two important drawbacks:

- First, public-key cryptosystems still need to distribute the public keys in a reliable way. For example, an attacker could intercept the public key and change it for one of his own. Then, the sender would encrypt the plaintext with that public key and the attacker would be able to read the message. To prevent this from happening, it is necessary a trusted third party, called certification authority (explained in Section2.1.2.3) that certifies that the public key belongs to the receiver. However, in case that the certification authority was compromised, the security of the system would be lost.
- Typically, the mathematical functions involved in public-key cryptosystems are much more complex than the ones used for private-key encryption. Therefore, they are much slower (2 or 3 orders of magnitude) [KAT14] and require much more hardware or computer resources.

A common scheme in communications consists of using an asymmetric algorithm to exchange the private keys and, then, use a symmetric cryptosystem (a block cipher or a stream cipher) to send the information.

2.1.2. Other cryptographic primitives

Apart from having a secure encryption algorithm, it is necessary to have a communication protocol that makes sure that the messages come from the intended sender and that they have not been modified (intentionally or accidentally). For this purpose, there are several cryptographic primitives that can be used: hash functions, message authentication code (MAC) and digital signatures. Each of them, can achieve some of the following security goals:

- **Integrity:** this property means that the receiver can assure that the message has not been accidentally modified.
- **Authentication:** this property means that the receiver can assure that the message was sent by the intended sender and has not been altered (accidentally or by an attacker).
- **Non-repudiation:** this property means that the sender cannot deny to have sent the message (if the sender denies it, the receiver or any other third person can prove that it was sent by him).

Table 2.1 shows these primitives as well as the security goal they accomplish and the kind of key (if any) that need to accomplish its objective.

Table 2.1. Cryptographic primitives and their security goals

Cryptographic primitive	**Hash**	**MAC**	**Digital signature**
Security Goal			
Integrity	Yes	Yes	Yes
Authentication	No	Yes	Yes
Non-repudiation	No	No	Yes
Kind of keys	None	Symmetric	Asymmetric

2.1.2.1. Hash functions

A hash function is a one-way function that maps data of any size into a bit array of a fixed size (hash). In order to be suitable for cryptography (cryptographic hash function), it must present these properties [RJA08]:

- It is deterministic, the same message always results in the same hash.
- Given a message, it is easy to compute its hash.
- It is difficult to obtain a message from its hash.
- Similar messages should generate very different hashes. In other words, ideally, the hashes corresponding to two messages that differ in a single bit should be totally uncorrelated.
- Since the length of the hash is usually shorter than the typical length of the message, there will be some possible messages that have the same hash. However, it must be infeasible to find in a reasonable amount of time two different messages with the same hash.

It must be pointed out that, if the message has a short length, even in the case of an ideal cryptographic hash function, it is still possible to recover a message from its hash using a brute-force search. For example, if an attacker knows that a hash has been obtained from an 8-bit message, he can calculate the hash for all possible 8-bit messages and check which of them matches with the correct hash. In practice, for most common hash functions, other people have already calculated the hashes of all short length messages and there are public tables, called rainbow tables, with pairs of message/corresponding hash [RAI19].

A hash can have multiple applications such as password storage [MIR05], randomness extraction [CLI09] or hash tables [LIU14] but, in the context of the transmission of a message, it can be used to provide integrity. A possible way to accomplish this is to calculate and append the hash of an encrypted message. In this case, the transmitted message, m, would be given by:

$$m = e_k(p) \,||\, h(e_k(p)) \tag{2.5}$$

where h is the hash function, e_k the encryption function, p the plaintext and $||$ denotes the concatenation function. To check that the message has not been modified during the transmission, the receiver can compute the hash of the encrypted message and check that it matches the transmitted hash. It must be noticed that this method only guarantees to detect accidental errors during the transmission. An attacker could modify the ciphertext, calculate its new hash, and substitute the old hash with the new one. This way, the receiver would not now that the ciphertext has been modified. To detect this intentional modifications, [KLE13] proposes to calculate the hash of the plaintext and append it to the ciphertext:

$$m = e_k(p)||h(p) \tag{2.6}$$

However, this scheme still has some flaws:

- First, by transmitting the hash of the plaintext, we could be transmitting some information about it. In the worst case, as it has been explained before, if the plaintext was very short, it could be obtained from its hash using a brute-force method or some rainbow table of matched hashes.
- Second, this scheme is still vulnerable to active attacks in many cryptosystems. For example, in the case of an additive stream cipher, the ciphertext, c, is given by $c = p \oplus z$, where p is the plaintext and z is the keystream. If an attacker could guess the plaintext p, he could easily change it by a new one, p'. This could be done by calculating a new ciphertext, c', as:

$$c' = c \oplus p \oplus p' = p' \oplus z \tag{2.7}$$

Then, at the end of the message, the attacker would substitute $h(p)$ by $h(p')$.

To guarantee that a message comes from the intended sender and it has not been modified by an attacker (authentication and integrity), a different cryptographic primitive, MAC, can be used.

2.1.2.2. MAC functions

Similarly to hash functions, MAC functions also transform data of any size into a fixed length output (tag). However, in this case, it is necessary to use a secret key during compression. Informally, a MAC consists of three algorithms [HAN08]:

- An algorithm that selects a random key from a key space.
- An algorithm that, given a key and a message, returns a tag.
- An algorithm that, given the key and the tag, verifies the authenticity of the message.

It must be noticed that a MAC requires a private key shared by both the sender and the receiver. Some of the MAC algorithms can be constructed from hash functions (HMAC: hash-based message authentication code) while other MAC algorithms can be constructed from block ciphers (CMAC: Cipher-based Message Authentication Code). To be secure, it must be infeasible to compute a valid tag of a given message without knowing the key, even if the attacker knows some pairs of message/HMAC(message). To use a MAC function to provide authentication of an unencrypted message, the sender simply has to compute the MAC of the message and append it. The receiver will then compute the MAC of the message and check that it matches the MAC sent by the transmitter. If the sender also wants to encrypt the message, there are three different options to assemble the encryption and the MAC function:

- MAC-then-Encrypt: calculate the MAC of the plaintext, append it to it, and encrypt everything.
- Encrypt-and-MAC: calculate the MAC of the plaintext, encrypt the plaintext and append the MAC at the end.
- Encrypt-then-MAC: encrypt the plaintext, calculate the MAC of the ciphertext and append it to the end.

All these approaches offer different levels of security. Explaining the advantages and disadvantages of each method is out of the scope of this work but, in summary, it can be said that the Encrypt-then-MAC offers the highest level of security [BEL08].

It must be pointed out that Mallory could still eavesdrop a MACed and encrypted message sent by Alice and send it later, pretending to be her (replay attack) [MAL02]. To avoid this, some measures must be taken such as including message numbers or timestamps on each message.

2.1.2.3. Digital Signatures

Finally, a digital signature is a scheme capable of providing the properties of authentication, integrity and non-repudiation. This process can be seen as a sort of an asymmetric encryption process where the message is encrypted using a private key but is decrypted using a public key. A signature is created by using a private key and is verified using a public key. This way, only the holder of the private key can sign a message but anyone who knows the public key can verify it.

In order to be reliable, it is necessary to be certain that the public key was indeed created by the person who is supposed to have signed the message. For this purpose, a trusted third party called Certification Authority (CA) is needed. The certification authority must check that the public-key corresponds to the intended user and issue a digital certificate that contains the public key and the identity of the owner. This certificate is digitally signed by the CA using its private key (Fig. 2.6). The public keys of the CA are widely known (often, they are included in web

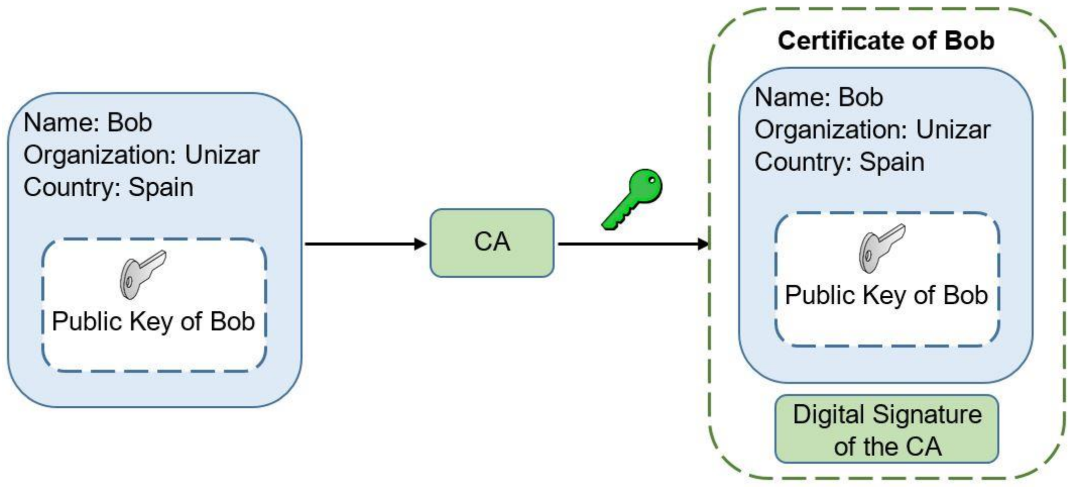


Fig. 2.6. Procedure of obtaining a public key certified by a CA. Bob sends its information and its public key to the CA. The CA verifies the identity and signs it with its private key.

browsers) and trusted so anyone can be sure that the certificate has been issued by the CA and that the user-sign association is valid.

Certification authorities, also often store public keys suitable for asymmetric encryption and the identity of their owners (for example, banks). This way, when the sender and the receiver need to start a communication, the certification authority can send the receiver’s public key to the sender with a certificate indicating that that key belongs to the receiver.

2.1.3. A possible secure communication protocol

Finally, let’s suppose that a sender and a receiver want to communicate with each other. They want to transmit several messages at a high speed, so they need to use a symmetric encryption algorithm. A secure communication protocol, using all of the primitives seen so far would be as follows:

- First, the receiver generates a pair of public-private keys needed for asymmetric encryption.
- The receiver sends the public key to a CA and the CA certifies that the public key belongs to the receiver. In practice, the receiver could be a bank and the CA could already have several public-private key pairs to distribute the keys among different users.
- The CA sends the public key to the sender with its digital signature and some extra information, such as the owner of the key (i.e., the receiver).
- The sender knows the public key of the CA so she can verify the signature, trust the certificate and the public key.
- The sender generates a key for a symmetric encryption algorithm (could be a block cipher or a stream cipher) as well as a key for a MAC.
- With the public key provided, the sender encrypts a message using a public-key algorithm. This message includes the keys of the symmetric cipher as well of the MAC.
- The receiver decrypts the message using its private key. This way, now the sender and the receiver both share some secret keys.
- Now the sender adds a timestamp (to prevent reply attacks), encrypts the message, calculate its MAC and appends it at the end.

- Since only the receiver and the sender know the secret keys (of the cipher and the MAC), the receiver is the only one that can decrypt the message and also verify that it belongs to the sender and has not been modified.

2.2. Chaotic cryptography

In this section, some important concepts used to design the encryption algorithms that have been proposed in this Thesis are presented. First, the definition of a chaotic system and a brief explanation of its main properties and their relationship with some important cryptographic properties is presented. Then, a commonly used structure of chaos-based stream ciphers will be presented. Finally, we will explain some basic security requirements that these systems must meet.

2.2.1. Definition and properties of chaotic systems

Although there is not a universally accepted definition of chaos, the following definition proposed by Robert L. Devaney in 1985 [DEV85] is commonly used:

Definition 2.1. Let V be a metric space. A continuous map $f : V \to V$ is said to be chaotic if:

- f has sensitive dependence on initial conditions
- f is topologically transitive
- Periodic points are dense in V

It must be pointed out that, in 1992, it was proven that, if $f : V \to V$ is transitive and has periodic points, then it has sensitivity dependence on initial conditions [BAN92]. However, since the sensitivity on initial conditions is the most characteristic property of chaotic systems, Devaney's definition is most commonly used. The definitions of these properties with some brief explanations are presented below.

Definition 2.2. $f : V \to V$ has sensitive dependence on initial conditions if there exists $\delta > 0$ such that, for any $x \in V$ and any neighborhood N of x, there exists $y \in N$ and $n \geq 0$ such that $|f^n(x) - f^n(y)| > \delta$

Intuitively, a map possesses this property if, for any point x, there exist other points arbitrarily close that separate from it by a distance bigger than δ after iterating f. It must be pointed out that not all points near x separate from it after iterating f but there must be at least one such point in every neighborhood of x. As a consequence of this property, the orbits of these systems cannot be computed accurately since any small round off errors are magnified upon iteration. Therefore, the computed orbit might present not resemblance at all with the real orbit.

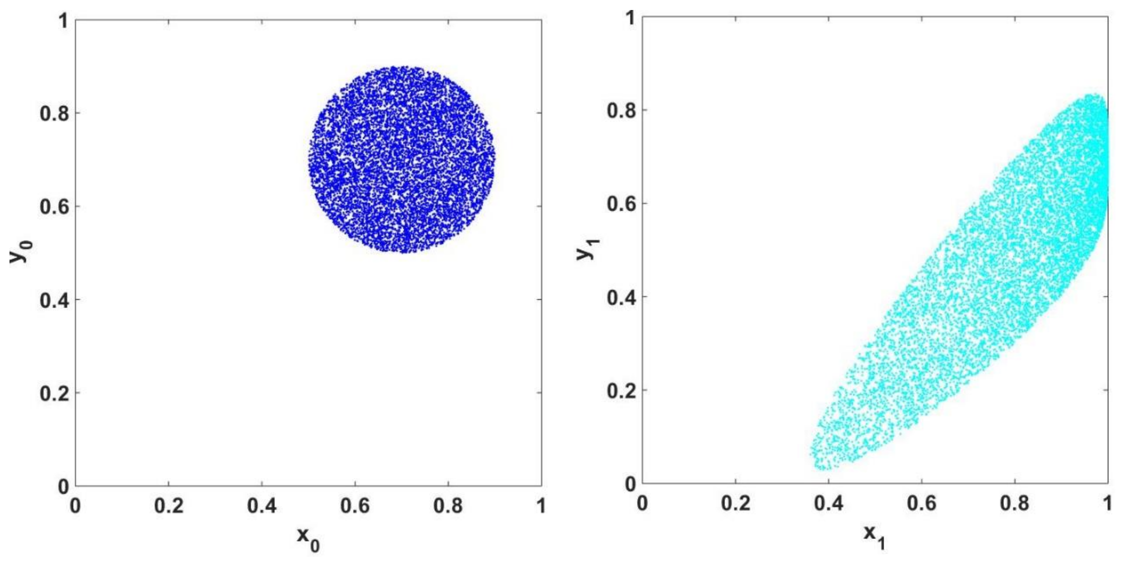
To quantify this property, the Lyapunov exponents are commonly used. Given two trajectories that start infinitesimally close, with an initial separation $|\delta Z_0|$, after some iterations will end up diverging at a rate given by:

$$|\delta Z(t)| \approx \mathrm{e}^{\lambda t}|\delta Z_0| \tag{2.8}$$

where λ is the Lyapunov exponent. It must be noticed that, if the phase space has several dimensions, there are several Lyapunov exponents (equal in number to the dimensionality of the system). In this case, the Maximal Lyapunov Exponent (MLE) is usually used, since it determines the sensitivity to initial conditions of the dynamical system.

Definition 2.3. $f : V \to V$ is topologically transitive if for any pair of open sets $U, V \subset V$ there exists $n > 0$ such that $f^n(U) \cap \mathrm{V} \neq \emptyset$

This property indicates that there are points that, after iterating f, move from an arbitrarily small neighborhood to any other. Therefore, the system cannot be decomposed into two disjoint open sets that are invariant under the map. This property is often referred to as the mixing property which corresponds to the standard intuition of mixing (for example, the mixing of different fluids).



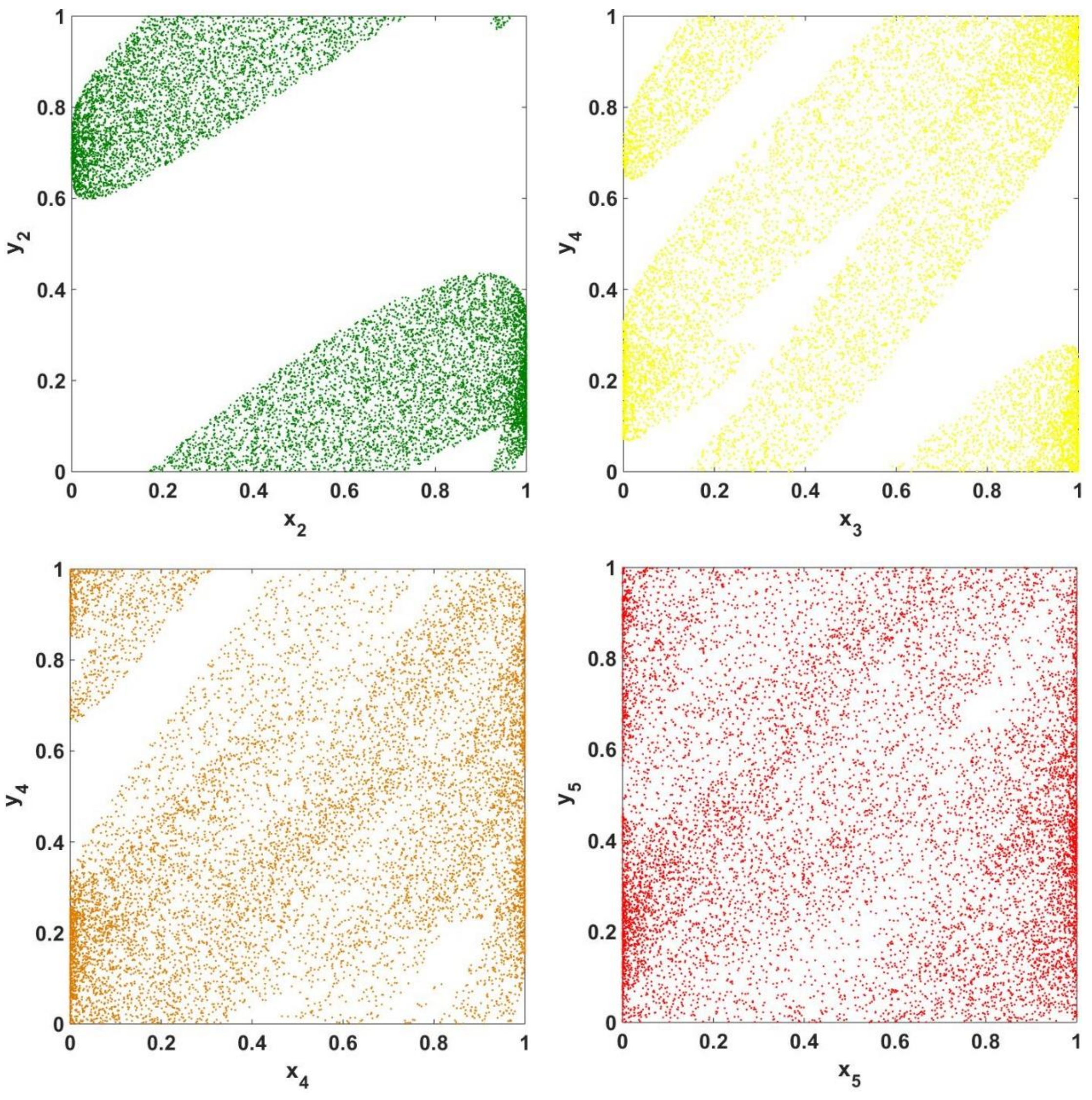


Fig. 2.7. Six consecutive iterations of a set of states $[x, y]$ passed through the logistic map, using the equations: $x_{i+1} = 4x_i(1 - x_i)$ and $y_{i+1} = \begin{cases} x_i + y_i & x_i + y_i < 1 \\ x_i + y_i - 1 & x_i + y_i > 1 \end{cases}$

As an example that illustrates this property, Fig. 2.7 shows how a set of points $[x, y]$ that initially form a circle are scattered through the space after applying some iterations using the Logistic Map equations.

Definition 2.4. A subset A of a topological space V is dense in V if for any point $x \in V$, any neighbourhood of x contains at least one point of A.

Therefore, the third property of Devaney's definition indicates that, given any point in the space and given a distance greater than zero, there will be a periodic orbit within that distance of the point. It must be pointed out that a chaotic system does not have to have periodic orbits at all. However, if it does, they must have this property.

Beyond this formal definition, chaotic systems present some other features that can be related to some cryptographic properties:

- Deterministic dynamics: the initial condition of the system determines unequivocally its evolution.
- Ergodicity: almost every trajectory tends to an invariant distribution that is independent of the initial conditions, and almost every trajectory will eventually visit any arbitrary interval of arbitrary size.
- Structure complexity: although chaotic systems are often determined by simple equations, the evolution process presents a very high complexity.

In 1949, Claude Shannon presented a paper discussing cryptography from the point of view of information theory [SHA49]. In that paper, he defined the properties of confusion and diffusion that were required by any secure cipher. Confusion referred to making the relationship between the ciphertext and the plaintext as complex as possible while diffusion referred to the fact that, small changes in the input, should produce big changes on the output.

These requirements along with other cryptographic properties, are tightly related to some intrinsic properties of chaotic systems, as explained in several works [ALV06, KOC02, KOC11]. In Table 2.2 [ALV06], a comparison between cryptographic and chaotic properties is presented.

2.2.2. Chaos-based stream ciphers

Due to the close relationship between the properties of chaos-based cryptosystems and some cryptographic properties explained in Table 2.2, chaos-based cryptosystems have been used in all kind of cryptographic primitives such as TRNGs [DRU07], public-key ciphers [KOC05, KOC11], hash functions [XIA05, AMI09], MAC functions [ARU07, KAN13], digital signatures [CHA13], block ciphers [AMI10, FOU14]. However, we have focused our research in chaos-based synchronous stream ciphers since they can usually provide higher encryption speed and, therefore, are more suitable for Gigabit Ethernet communications. Most of the proposed chaos based stream ciphers, are based

Table 2.2. Comparison between cryptographic and chaotic properties

Chaotic property	Cryptographic property	Description
Ergodicity	Confusion	The output has the same distribution for any input
Sensitivity to initial conditions/control parameter	Diffusion with a small change in the plaintext/key	A small deviation in the input causes a big change at the output
Mixing property	Diffusion with a small change in one plain-block of the whole plaintext	A small deviation in the local area can cause a large change in the whole space
Deterministic dynamics	Deterministic pseudo-randomness	A deterministic process can cause a pseudo-random behavior
Structure complexity	Algorithm (attack complexity)	A simple process has a very high complexity

on Chaotic Maps, which are discrete maps that exhibit chaotic behavior. These systems are usually defined by the equation:

$$X_{i+1} = f(X_i, \Gamma) \tag{2.9}$$

where, X_i is the state at the discrete time i, that, in general, can be in an *n*-dimensional space ($X_i = [x_{i,0}, x_{i,1}, \dots x_{i,n},]$), $\Gamma = [\gamma_0, \gamma_1, \dots \gamma_l]$ is the set of control parameters that determine the behavior of the system and $f : V \to V$ is a map of the state space into itself. The sequence $\{X_i\}$ obtained by applying (2.9) is unequivocally determined by the initial state, X_0, and the chosen control parameters Γ. If f is chaotic, it is expected that the sequence presents a random-like behavior. When this system is implemented in a digital system (e.g., ASIC or FPGA), a precision must be used to represent each state and each control parameter (usually, the same precision *n* is used for all of them) and an arithmetic (e.g., fixed or floating point) must be defined. One may think that the obtained (digitized) sequence could be directly used to generate a random keystream and constructing a secure stream cipher. This however, presents three important issues:

- First, when a chaotic map is digitized, the generated sequences become periodic (they are not strictly chaotic). This fact can affect the randomness of the generated sequences and, also, can cause the sequence to repeat itself after a few iterations, becoming a non-secure cipher.
- Second, if the map is invertible, if one or several states were leaked, an attacker could use the equations of the map to obtain the control parameters and reveal the whole sequence (reconstruction attack). Even if it is not exactly invertible, the attacker could still try different possibilities and still manage to reconstruct the map. It must be noticed that in a stream cipher states can be easily leaked, for example, when they are XORed with a plaintext consisting with all 0's.
- Third, since each state is usually obtained from the previous one, using only a few operations, it is common that the bits within each x_i present a high correlation.

To solve the first problem, the algorithms must be modified to prevent the system to fall into short period orbits while, for avoiding the reconstruction attack, a one-way function, g, should be used to obtain the keystream from the states of the system. This function can also help to improve the randomness of the sequence, therefore, solving the last problem (Fig. 2.8).

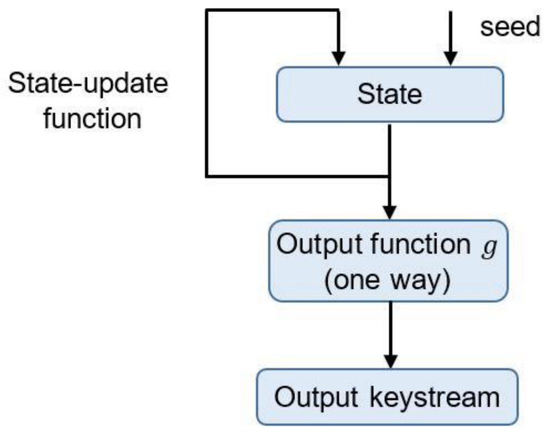


Fig. 2.8. Example of a secure way of generating a keystream.

2.2.3. Basic security requirements of chaos-based stream ciphers

Stream ciphers, as any other ciphers, can be attacked in many different ways and many of the attacks are specific for each algorithm. This means that, often, it is difficult to assure that a proposed algorithm is completely secure since, after some time, a new effective attack targeting this algorithm could be proposed. However, there are some basic tests that can be used to evaluate the security of a stream cipher. Failing any of these tests means that the stream cipher is not secure. On the other hand, although passing all these test does not guarantee that the system is secure, it is often considered as a good indicator of its security. The main requirements that a stream cipher must meet are:

- First, the key size (number of possible keys) should be big enough to prevent brute-forcing. Nowadays, according to National Institute of Standards and Technology (NIST) [BAR11] and European Union Agency for Network and Information Security (ENISA) [ENI14], in order to be secure, the key size should be larger than 2^{112}.
- Second, the system must present a high sensitivity to initial conditions (i.e., two keystreams generated by slightly different keys, should be completely different). In some cases, when several sub-parameters are used as part of the key, knowing some part of the key can reveal partial information about the rest of the key, thus reducing the effective key size. This can sometimes be achieved by fixing one of the sub-parameters and trying to estimate the other ones using a Bit-Error-Rate (BER) attack [WAN04, ALV04]. To avoid this, the statistical differences (if any) between the ciphertext and a keystream generated using a wrong key should always be the same, except when all the sub-parameters that form the key are correct.
- Third, by knowing the keystream and the algorithm used, it should be impossible to reconstruct the map and find the initial parameters in a reasonable amount of time. To avoid this, as explained in Section 2.2.2, it is important to use a one-way function to obtain the bits of the keystream from the internal state of the system.
- Finally, the keystreams generated by any possible key should be indistinguishable from a random sequence. To check if these sequences are random, several batteries of tests are commonly used such as NIST SP 800-

22 battery of tests [RUK10], diehard tests [MAR95] and U01 tests [ECU07]. Since it is impossible to assure that a sequence is really random (even an ideal PRNG could produce a sequence of all 1's and fail the tests), these tests are defined with a significance level, α, which indicates the fraction of sequences that should fail these test assuming that they were generated by an ideal RNG. Therefore, to evaluate if a PRNG is good, several sequences (typically 100 is a suitable number) should be subjected to the randomness tests and it should be checked that approximately only a fraction α of the tests fail these tests.

2.3. TRNGs

Contrary to PRNGs, TRNGs generate the random numbers from a physical process, instead of a deterministic algorithm. Therefore, an ideal TRNG is completely unpredictable, which makes it the best choice for generating secure cryptographic keys. Unfortunately, sequences generated by TRNGs usually present worse statistical properties with respect to sequences generated by PRNGs since physical processes often present some correlations and the instruments used to measure them, can make the sequences biased. As a consequence, these sequences must pass through a process to improve their statistical properties called post-processing. On the other hand, standard randomness tests suites are not usually a good tool to compare not-perfect generators, since they usually provide yes/no answers.

In this section, some simple statistical tests that can be used to compare the randomness of several TRNGs will be presented. Furthermore, we introduce some post-processing algorithms that are commonly used.

2.3.1. Statistical tests

Before presenting the statistical tests, we will present an example of a simple TRNG, to understand the statistical defects that they might present.

Let's imagine that we have a physical variable that varies with time according to a function $f(t)$ given by:

$$\begin{gathered} f(t) = \sin(t + \epsilon(t)) \\ \epsilon(t) = \epsilon(t - \Delta t) + rand(t) \end{gathered} \tag{2.10}$$

where $rand(t)$ is an random function that, at each moment of time, gives a random value in the range $[-0.05, 0.05]$. This function is a sine function with some phase and frequency noise, as shown in Fig. 2.9.

A binary signal could be obtained by measuring the value at discrete times $t_0, t_1, \dots t_n$, separated by a fixed amount time, ΔT, i.e. $t_j = j\Delta T$. Then, each measurement could be compared to 0 to generate a binary sequence $A = (a_0, a_1, \dots a_n)$. If $f(t_j) \geq 0, a_j = 1$ and if $f(t_j) < 0, a_j = 0$.

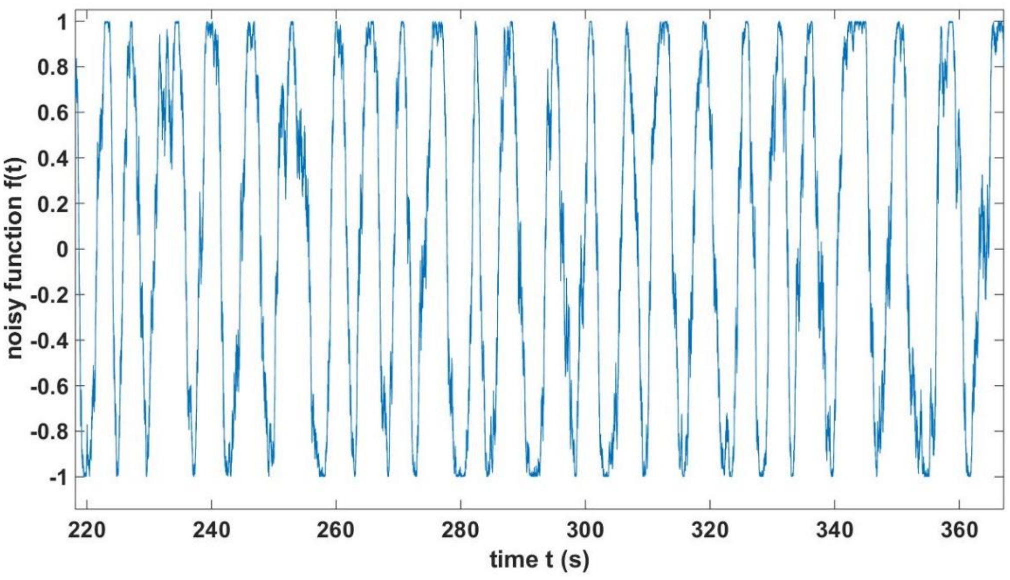


Fig. 2.9. Example of a signal obtained from a physical noise source.

It is clear that, if $\Delta T \ll 2\pi$, we will be sampling through an approximately sine curve, and, in most cases each bit would be equal to the previous and the next ones. The fact that knowing some bits of the sequence gives you information about other bits of the sequence is called statistical dependence.

However, if the sampling period is increased, the phase noise accumulates and the statistical dependence between consecutive bits decreases. Finally, when $\Delta T \gg 2\pi$, the sequence will present almost no statistical dependence.

This pattern is very common in most of the TRNGs. Typically, noise sources have some harmonic components that causes a statistical dependence between the generated bits. This statistical dependence, however, can be reduced and practically eliminated by decreasing the sampling frequency. Therefore, it is relatively easy to design a TRNG that generates independent bits, as long as the throughput is not a constraint.

On the other hand, let's imagine that the instrument used to measure this physical variable is not well calibrated so that if $f(t_j) \geq -0.01, a_k = 1$ and if $f(t_j) < -0.01, a_j = 0$. The generated sequence would probably have more 0's than 1's. This difference is called bias, and is usually associated with imperfections in the measuring equipment.

This bias is usually constant and cannot be reduced by changing the sampling frequency. However, we will show in Section 2.3.2 that there are post-processing methods that, assuming that the sequence has no statistical dependence, can effectively eliminate this bias.

To sum up the quality of a TRNG depends on two parameters: the statistical dependence and the bias.

- A sequence presents statistical dependence if knowing some bits of the sequence gives you some information that helps you guess other bits of that sequence.
- A sequence is biased if the number of 1's is considerably different from the number of 0's.
- An unbiased source with some statistical dependence can generate good random numbers by using a low sampling frequency.
- A biased source with no statistical dependence can generate good random numbers if some post-processing methods are used.

Since both parameters are usually independent, in order to evaluate the sequences generated by the proposed TRNGs, we have used statistical tests that give information about both of them separately.

Pattern distribution of bytes

This test consists of plotting successive bytes (b_j, b_{j+1}) of a sequence in a plane. If the sequence is random, the plotted points should have a uniform distribution (Fig. 2.10d). If instead, there are areas with low density and high density, the sequence is clearly not random (Fig. 2.10a, Fig. 2.10b, Fig. 2.10d). In the case that the density of points increases when the points get close to a corner, it is an indication that the sequence presents a bias (Fig. 2.10a, Fig. 2.10c).

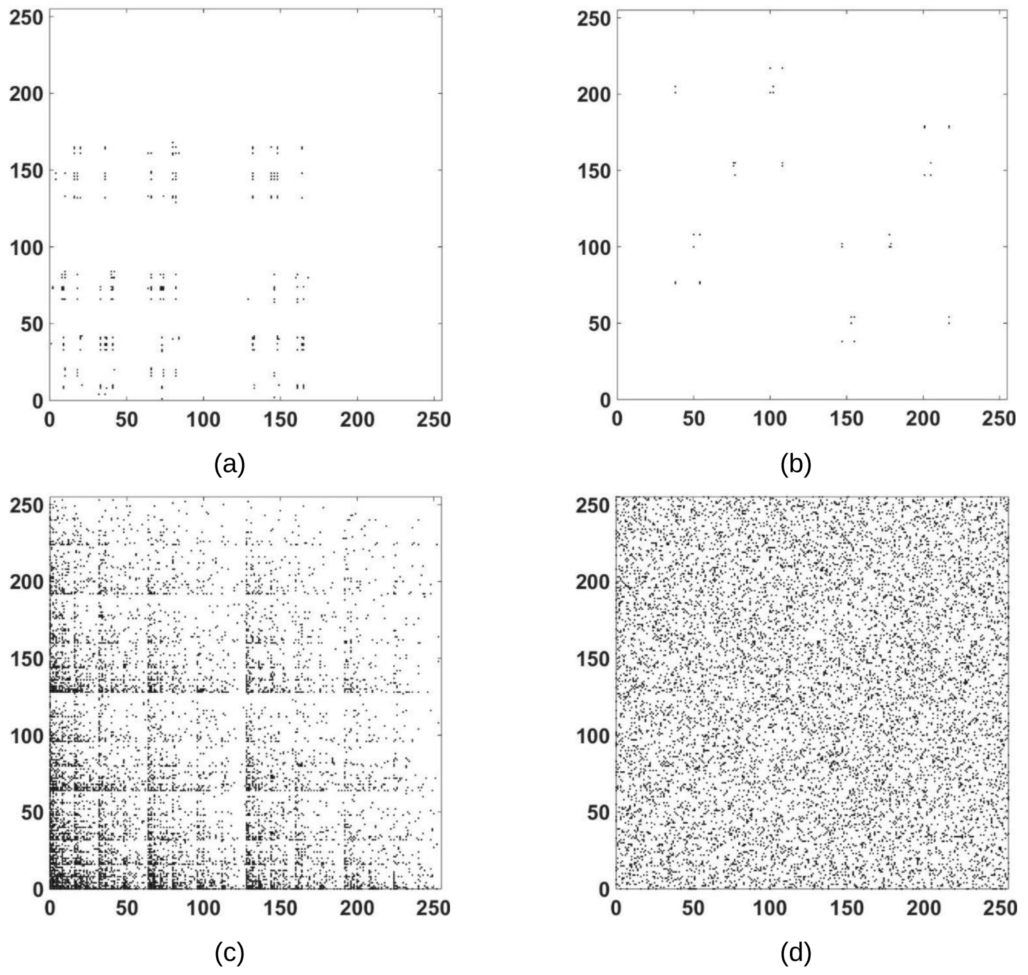


Fig. 2.10. Pattern distribution of bytes of a sequence with (a) bias and statistical dependence, (b) no bias and statistical dependence, (c) bias and no statistical dependence and (d) no bias and no statistical dependence. In each case, for each pair of consecutive bytes, (b_j, b_{j+1}) in the sequence, a point of coordinates $(x = b_j, y = b_{j+1})$ is plotted.

<u>Normalized autocorrelations</u>

Definition 2.5. Let $A = (a_0, a_1, \dots a_{n-1}),\ a_i \in \{0,1\}$ be a binary sequence with n elements. We define the normalized autocorrelations of A as:

$$R_j = \frac{1}{n-j} \sum_{i=0}^{n-j-1} a_i a_{i+j} \tag{2.11}$$

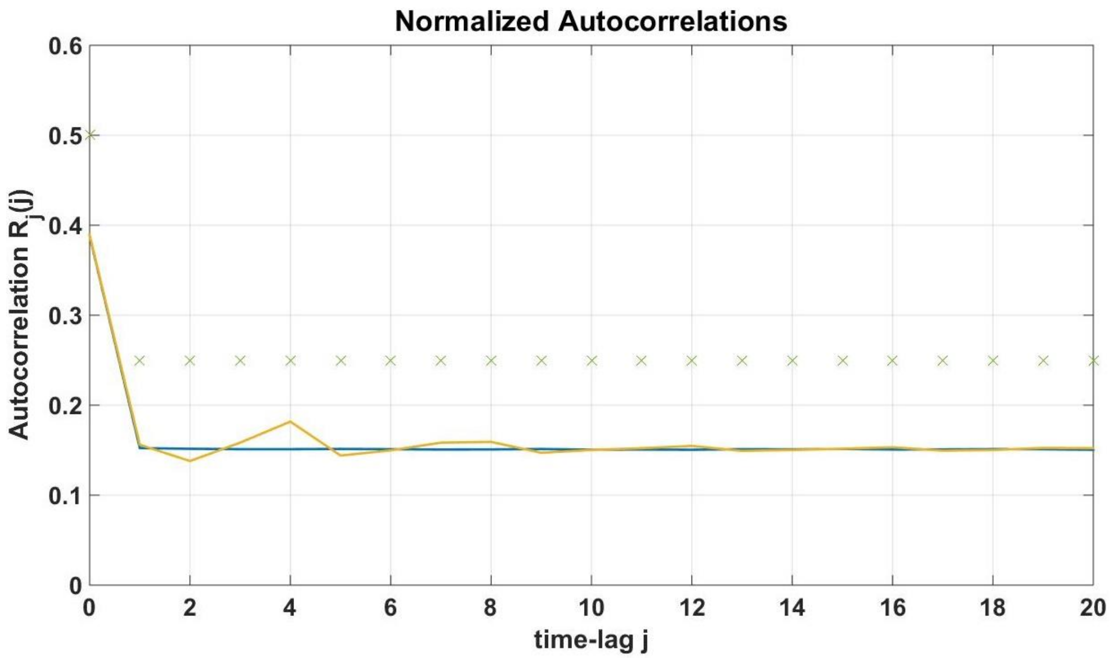
The parameter R_j can be seen as a measurement of how strongly A resembles a *j*-times-shifted version of itself, $A' = \left(a_j, a_{j+1}, \dots a_{n-j-1}\right)$. If $R_j = 1$, both sequences A and A' have perfect autocorrelation and, if $R_j = 0$, they have negative autocorrelation. If the sequence has been generated by an ideal TRNG, the probability that a randomly selected bit is 0 ($Pr(0)$) or 1 ($Pr(1)$) must be the same. In this case ($Pr(0) = Pr(1) = 1/2$), it can be proved that the values of R_j are: $R_0 = 0.5$, $R_j = 0.25$ ($j > 0$). On the other hand, if the sequence has a bias $Pr(0) \neq Pr(1)$, but no statistical dependence, the values of R_j are $R_0 = Pr(1)$, $R_j = Pr^2(1)$ ($j > 0$). Finally, it must be noticed that, increasing the value of j, is usually equivalent to decreasing the sampling frequency. Therefore, in case that the sequence is not ideally uncorrelated, usually the values of R_j will approximate to the ideal value P_1^2 when the value of j is increased, i.e., $\lim_{j\to\infty}(R_j) = P_1^2$. As an example, Fig. 2.11a shows the correlations of a biased sequence with and without correlation and Fig. 2.11b shows the correlations of an unbiased sequence with and without correlation.

Average Shannon Entropy and Redundancy

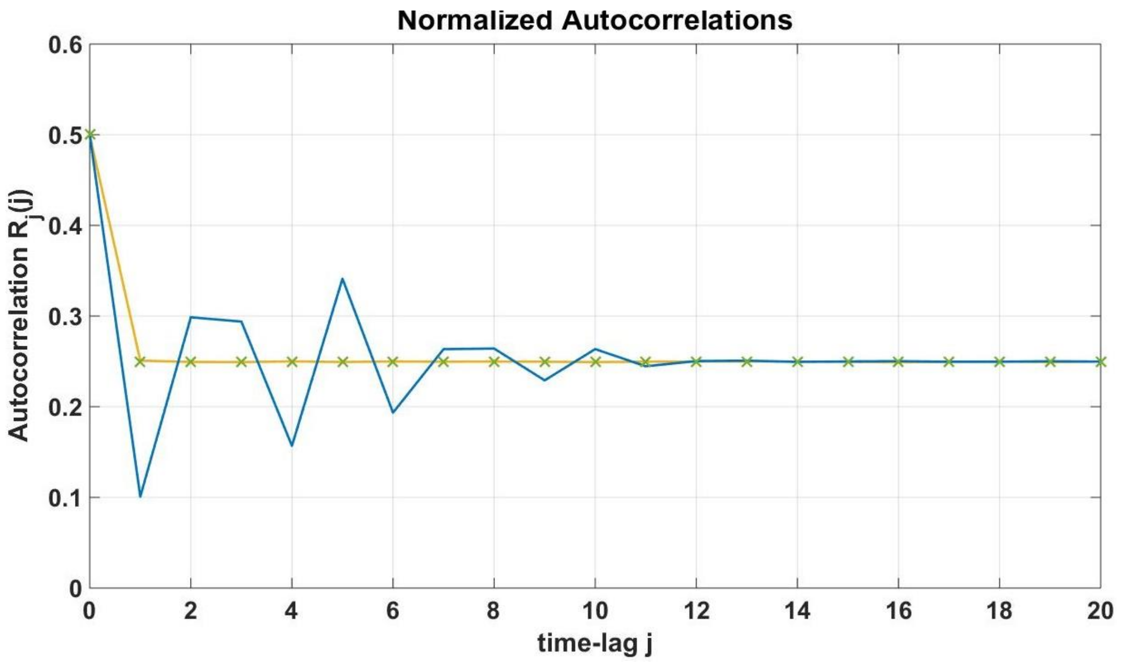
The Average Shannon Entropy (ASE), introduced by Information Theory, is a tool that can be used to measure how well a TRNG approximates the behavior of an ideal unpredictable binary random source. It is defined as:

$$\mathrm{ASE} = \lim_{n\to\infty} -\frac{1}{n} \sum_{\beta\in\{0,1\}^n} Pr(\beta)\log_2 Pr(\beta) \qquad [\frac{\mathrm{bit}}{\mathrm{symbol}}] \tag{2.12}$$

where the summation goes over all the 2^n possible *n*-tuples $\beta = \{b_0, b_1, \dots b_{n-1}\}$, $b_i \in \{0,1\}$ and $Pr(\beta)$ is the probability that the tuple β is generated. The unit of entropy is often expressed as bit/symbol, since it measures how many bits of information are being transmitted on average per symbol. If the TRNG is perfect, the ASE is maximum and equals 1 bit/time-step and, if the TRNG is completely predictable (its output sequences have all 0's or all 1's), its ASE is minimum, and equals 0 bit/time-step.

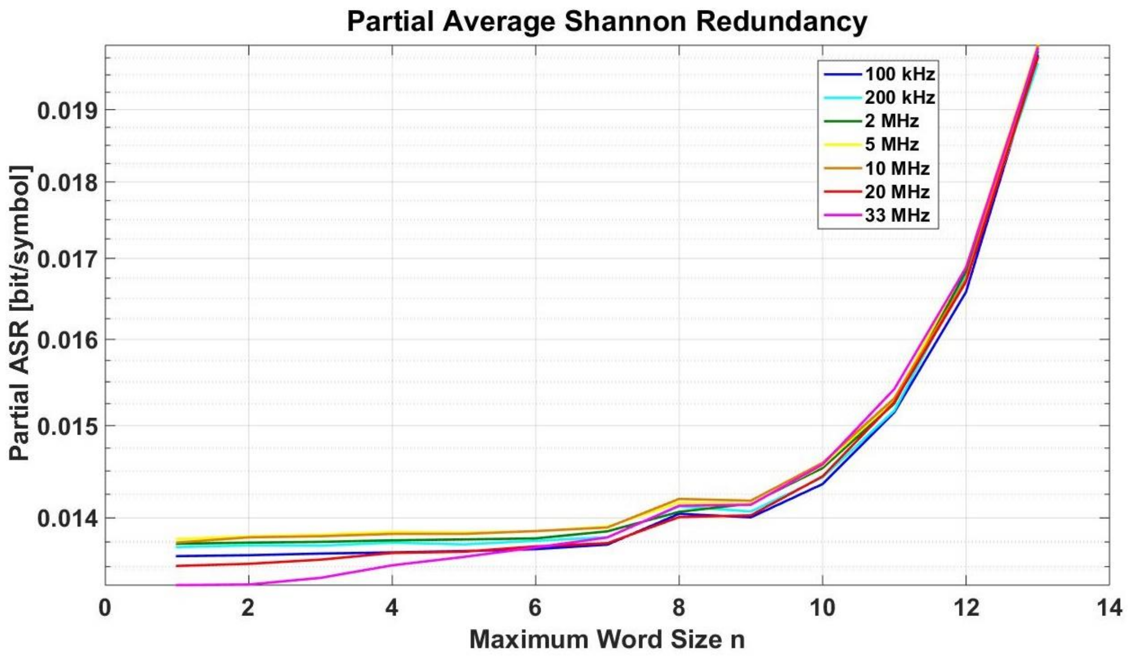


(a)

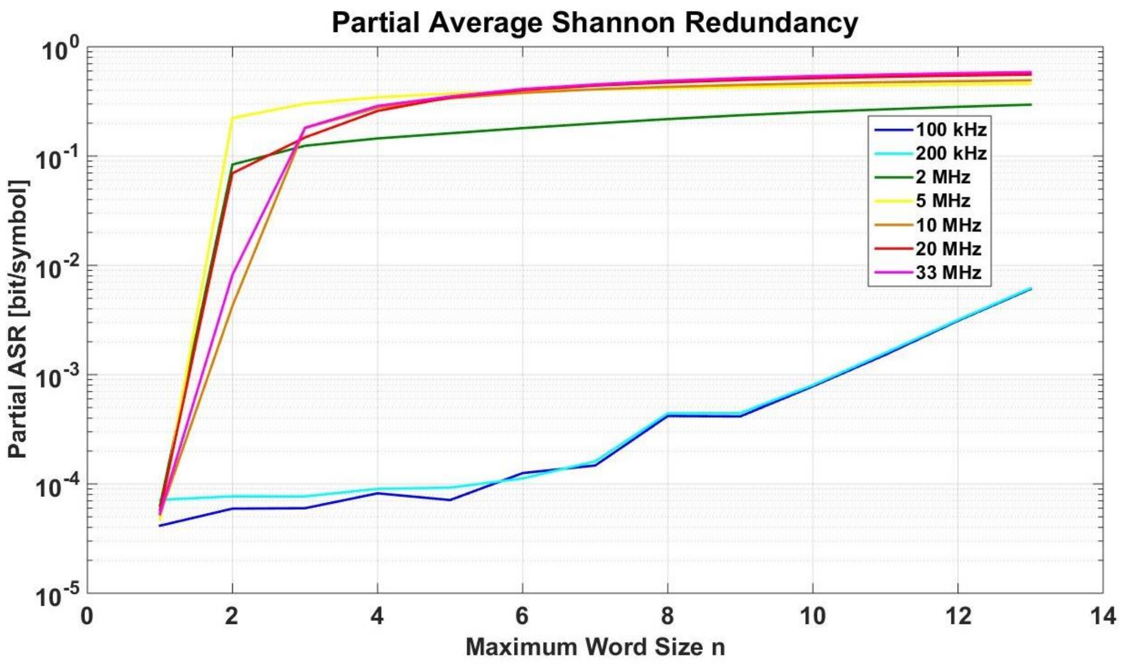


(b)

Fig. 2.11. Correlations of (a) biased source, (b) unbiased source. Sequences in yellow are statistically independent and sequence in blue present some statistical dependence. The green dots represent the values of an ideal random source.



(a)



(b)

Fig. 2.12. ASR_n of sequences generated by a TRNG with (a) bias and no statistical dependence, (b) no bias and statistical dependence.

Since the computation of the ASE is unfeasible, an approximation of the ASE can be obtained by truncating its expression, obtaining the Partial Average Shannon Entropy:

$$\mathrm{ASE}_n = -\frac{1}{n}\sum_{\beta\in\{0,1\}^n} Pr(\beta)\log_2 Pr(\beta) \qquad [\frac{\mathrm{bit}}{\mathrm{symbol}}] \tag{2.13}$$

The probability of each symbol can be estimated as the number of times that that symbol has been observed in the sequence divided by the total number of observations.

This parameter takes into account both the statistical dependence and the bias, giving and absolute number that can be used to compare among several TRNGs. The higher its value, the more random the sequence is. An alternative parameter to be used is the Average Shannon Redundancy, defined as:

$$\mathrm{ASR}_n = 1 - \mathrm{ASE}_n \tag{2.14}$$

This way, sequences with lower redundancies can be considered to be more random. The main drawback of both entropy and redundancy is that, from a single measurement, it is not possible to determine if the deviation of the ideal value is caused by a bias or by a statistical dependence. A possible way to determine it is to measure the redundancy of sequences obtained by sampling the noise source at different frequencies. If the redundancy does not change with the sampling frequency, the TRNG is biased but does not have statistical dependence (Fig. 2.12a). If however, for low sampling frequencies the redundancy is high but, for high sampling frequencies the redundancy is close to 0, the TRNG is unbiased but does not have statistical dependence (Fig. 2.12b).

2.3.2. Post-processing algorithms

Depending on the non-idealities of the TRNG, different post-processing algorithms can be used reduce the bias, the statistical dependence or both of them. Here, some of the most common methods will be presented:

<u>Bias reduction post-processing</u>

One of the most known methods for reducing a bias of a sequence, is the Von Neumann post-processing [VON51]. This method processes the sequence as a stream of non-overlapping pairs of consecutive bits and outputs a post-processed sequence, following the next rules:

- If the input is “00” or “11” there is no output.
- If the input is “01” the output is “0”.

- If the input is “10” the output is “1”.

It can be proved that, if the bits are statistically independent, the bias is completely eliminated. However, this method has two important issues. First, the throughput of the output sequence is considerably reduced (between 75% and 100% of the bits are discarded). The second problem is that the throughput is variable, which can be a problem depending on the implementation. To solve these issues, other post-processing techniques based on this procedure have been proposed [ELI72, PER92].

Finally, although they are not strictly post-processing procedures, we must point out that some authors such as Bagini-Bucci [BAG99] or Stipčević [STI04] use an analogue signal conditioning stage that can eliminate the bias of the generated sequences.

Statistical dependence reduction

Any unbiased sequence generated by a TRNG that presents a strong statistical dependence can be transformed into a perfectly random sequence if it is correctly shuffled (i.e., the order of the bits is randomly changed).

A common algorithm for generating a random permutation of a finite sequence (shuffling) is the Fisher-Yates shuffle, originally proposed by Ronald Fisher and Frank Yates and improved by Donald Knuth [KNU69]. Given a sequence $A = (a_0,\ a_1, \dots a_{n-1})$ this shuffling algorithm consists on performing *n*-1 iterations. In the ith iteration:

- Generate a random integer j such that $0 \leq j \leq i$
- Exchange a_j and a_i

The main drawback of this system is that it is necessary to use another random number generator to do the shuffling. If these random numbers were known, the process could be reversible so an attacker could obtain the raw sequence. Therefore, this sort of post-processing is rarely used. Instead, other post-processing techniques that simultaneously improve both the bias and the statistical independence of the sequences are commonly used.

Techniques to improve both bias and statistical dependence

One of the simplest methods to improve the statistical quality of a sequence generated by a TRNG, consists of XORing n bits of the sequence to generate 1 bit of the output sequence $(n > 1)$. Similarly to Von Neumann procedure, this method reduces the throughput although, in this case, the throughput reduction is constant.

Another common method is to XOR the sequence with another random sequence, typically generated by a PRNG. One of the most commonly used PRNGs for this purpose are the Linear Feedback Shift Registers (LFSRs) since they are very simple to implement and they present good statistical properties. In some cases, the sequence and the LFSR are combined so that the output random bits are used to feed back the LFSR [DIC07].

Finally, other compression methods such as hash functions or MAC functions can be used. The National Institute of Standards and Technology gives some guidelines about which of these functions can be used and as well as an entropy estimation in each case [TUR18].

2.4. Conclusions

In this chapter, the main concepts and definitions that will be used in the rest of this Thesis have been introduced. In Section 2.1, the main cryptographic primitives and their application in a secure communication protocol have been explained. Among them, our research has focused on the design of new ciphers (including only the encryption algorithm and the key generation process). From this Section, we can deduce that symmetric ciphers and, specially, stream ciphers, can offer higher encryption speeds using a smaller area. For this reason, in this Thesis we have focused on the design of this kind of ciphers.

In Section 2.2, the definition and properties of chaotic systems as well as a possible way of using them in secure communications have been presented. In this Thesis, we have decided to use this approach to design chaos-based stream ciphers trying to meet all the security requirements detailed in Section 2.2.3.

Finally, an introduction to TRNGs, explaining some of the quality tests that can be performed to assess their quality have been presented. By comparing these tests results for different random sequences (ideal and non-ideal) we have shown that, by analyzing the pattern distribution of bytes, the autocorrelations or the ASR, it is possible to determine if a given sequence present bias or statistical dependency. On the other hand, some post-processing techniques that can be used to improve their randomness have been included. Since the goal of this Thesis is to use a very simple post-processing stage, we will use the method consisting of combining the generated sequence with a simple pseudo-random sequence generated by an LFSR. Therefore, a good way to determine if the proposed TRNGs are good will be to see if they can pass the NIST test after applying this simple post-processing.

2.5. References

[ALV06] G. Alvarez, S. Li, “Some Basic Cryptographic Requirements for Chaos-Based Cryptosystems,” *International Journal of Bifurcation and Chaos*, vol. 16, no. 8, pp. 2129-2151, 2006.

[AMI09] M. Amin, O. S. Faragallah, A. A. Abd El-Latif, “Chaos-Based Hash Function (CBHF) for Cryptographic Applications,” *Chaos, Solitons & Fractals*, vol. 42, no. 2, pp. 767-772, 2009.

[AMI10] M. Amin, O. S. Faragallah, A. A. Abd El-Latif, “A Chaotic Block Cipher Algorithm for Image Cryptosystems,” *Communications in Nonlinear Science and Numerical Simulation*, vol. 15, no. 11, pp. 3484-3497, 2010.

[ARU07] G. Arumugam, V. L. Praba, S. Radhakrishnan, “Study of Chaos Functions for their Suitability in Generating Message Authentication Codes,” *Applied Soft Computing*, vol. 7, no. 3, pp. 1064-1071, 2007.

[AUM02] C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, J. P. Seifert, “Fault Attacks on RSA with CTR: Concrete Results and Practical Countermeasures,” *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 2523, pp. 260-275, 2002.

[BAG99] V. Gabini, M. Bucci, “A Design of Reliable True Random Number Generator for Cryptographic Applications,” *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems 1999 (CHES 1999)*, pp. 204-218, 1999.

[BAN92] J. Banks, J. Brooks, G. Cairns, G. Davis, P. Stacey, “On Devaney’s Definition of Chaos,” *The American Mathematical Monthly*, vol. 99, no. 4, pp. 332-334, 1992.

[BAR11] E. Barker, A. Roginsky, “Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths,” *NIST Special Publication 800-131 A*, 2011.

[BAT04] L. Batina, P. Buysschaert, E. De Mulder, N. Mentens, S. B. Ors, B. Preneel, G. Vandenbosch, I. Verbauwhede, “Side Channel Attacks and Fault Attacks on Cryptographic Algorithms,” *Revue HF Tijdschrift*, 3, pp. 36-45, 2004.

[BAS17] I. Bashir, “Mastering Blockchain,” *Packt Publishing*, 2017.

[BEL08] M. Bellare, C. Namprempre, “Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm,” *Journal of Cryptology*, vol. 21, no. 4, pp. 469-491, 2008.

[BER13] D. J. Bernstein, Y. Chang, C. Cheng, L. Chou, N. Heninger, T. Lange, N. van Someren, “Factoring RSA Keys from Certified Smart Cards: Coppersmith in the Wild,” *Advances in Cryptology – ASIACRYPT*, pp. 341-360, 2013.

[BUC17] W. J. Buchanan, “Cryptography (River Publishers Series in Information Science and Technology),” *River Publishers*, 2017.

[CHA13] K. Chain, W.-C. Kuo, “A New Digital Signature Scheme Based on Chaotic Maps,” *Nonlinear Dynamics*, vol. 74, no. 4, pp. 1003-1012, 2013.

[CLI09] Y. Cliff, C. Boyd, J. M. González, “How to Extract and Expand Randomness: a Summary and Explanation of Existing Results,” *Proceedings of the 7th International Conference on Applied Cryptography and Network Security*, 2009.

[COO71] S. Cook, “The Complexity of Theorem Proving Procedures,” *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pp. 151-158, 1971.

[DAE98] J. Daemen, V. Rijmen, "AES Proposal: Rijndael," *Addison-Wesley*, 1985.

[DEV85] R. L. Devaney, “An Introduction to Chaotic Dynamical Systems," *AES submission*, 1998.

[DIC07] M. Dichtl, “Bad and Good Ways of Post-Processing Biased Physical Random Numbers," *Proceedings of the Fast Software Encryption*, pp. 137-152, 2007.

[DIF76] W. Diffie, M. E. Hellman, “New Directions in Cryptography,” *IEEE Transactions on Information Theory*, vol. it-22, no. 6, pp. 644-654, 1976.

[DRU07] M. Drutarovsky, P. Galada, “A Robust Chaos-Based True Random Number Generator Embedded in Reconfigurable Switched-Capacitor Hardware,” *Proceedings of the 17th International Conference Radioelektronika*, pp. 1-6, 2007.

[DWO01] M. Dworkin, “Recommendation for Block Cipher Modes of Operation. Methods and Techniques,” *NIST Special Publication 800-38A,* 2001.

[ECU07] P. L.’Ecuyer, R. Simard, "TestU01: A C Library for Empirical Testing of Random Number Generators," *ACM Transactions on Mathematical Software*, vol. 33, no. 4, 2007.

[ELG85] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Transactions on Information Theory*, vol. IT-31, pp. 469-472, 1985.

[ENI14] ENISA, "Algorithms, Key Size and Parameters Report," Available online: www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-size-and-parameters-report-2014 (accessed on 25 February 2019), 2014.

[FOU14] J. S. A. E. Fouda, J. Y. Effa, S. L. Sabat, M. Ali, "A Fast Chaotic Block Cipher for Image Encryption," *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, no. 3, pp. 578-588, 2014.

[HAN08] H. Handschuh, B. Preneel, “Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms,” *Advances in Cryptology-CRYPTO 2008 volume 5157 of Lecture Notes in Computer Science*, pp. 144-161, 2008.

[KAN13] A. Kanso, M. Ghebleh, “A Fast and Efficient Chaos-Based Keyed Hash Function,” *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 1, pp. 109-123, 2013.

[KAT14] J. Katz, Y. Lindell, “Introduction to Modern Cryptography,” *Chapman & Hall/CRC*, 2014.

[KLE13] A. Klein, “Stream Ciphers,” *Springer-Verlag*, 2013.

[KNU69] D. E. Knuth, “Seminumerical algorithms. The Art of Computer Programming,” *Addison-Wesley*, pp. 139-140, 1969.

[KOC02] L. Kocarev, “Chaos-based cryptography: A brief overview,” *IEEE Circuits and Systems Magazine,* vol. 1, no. 3, pp. 6-21, 2002.

[KOC05] L. J. Kocarev, J. Makraduli, “Public-key Encryption Based on Chebyshev Polynomials,” *Circuits, Systems Signal Processing*, vol. 24, no. 5, pp. 497-517, 2005.

[KOC11] L. Kocarev, S. Lian, “Chaos-based cryptography: Theory, Algorithms and Applications,” *Springer, Berlin,* 2011.

[LIU14] D. Liu, Z. Cui, S. Xu, H. Liu, “An Empirical Study on the Performance of Hash Table,” *Proceedings of the 13th International Conference on Computer and Information Science (ICIS)*, 2014.

[MAL02] S. Malladi, J. Alves-Foss, R. B. Heckendorn, “On Preventing Replay Attacks on Secure Protocols,” *Proceedings of the 7th International Conference on Security and Management*, pp. 77-83, 2002.

[MAR95] G. Marsaglia, “The Marsaglia Random Number CD-ROM Including the Diehard Battery of Tests of Randomness,”, 1995.

[MEN97] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, “Handbook of Applied Cryptography,” *CRC Press*, 1997.

[MIR05] I. Mironov, “Hash functions: Theory, Attacks, and Applications,” *Microsoft Research, Silicon Valley Campus*, 2005.

[NAT77] National Bureau of Standards (U.S), Federal Information Processing Standards Publication 46, “Data Encryption Standard (DES),” 1977.

[RAI19] RainbowCrack Project, “List of Rainbow Tables,” 2019. Available at: https://project-rainbowcrack.com/table.htm.

[RIV78] R. Rivest, A. Shamir, L. Adleman, “A Method for Obtaining Digital Signatures and Public Key Cryptosystems,” *Communications of the ACM,* vol. 21, no. 2, pp. 120-126, 1978.

[RJA08] M. Rjasko, “Properties of Cryptographic Hash Functions,” *Cryptology ePrint Archive, Report 2008/527*, 2008.

[RUK10] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” *NIST Special Publication 800-22 Rev.1.a,* 2010.

[SHA49] C. E. Shannon, “Communication Theory of Secrecy Systems,” *Bell System Technical Journal*, vol. 28, pp. 656-715, 1949.

[STI04] M. Stipčević, “Fast Nondeterministic Random Bit Generator Based on Weakly Correlated Physical Events,” *Review of Scientific Instruments*, vol. 75, no. 4442, 2004.

[STI05] D. Stinson, “Cryptography: Theory and practice,” *CRC Press, Boca Raton, 3th edition*, 2005.

[TIL07] S. Tillich, C. Herbst, S. Mangard, “Protecting AES Software Implementations on 32-Bit Processors Against Power analysis*,” Applied Cryptography and Network Security*, pp. 141-157, 2007.

[TIR04] K. Tiri, I. Verbauwhede, “A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation,” *Proceedings of the Conference on Design, Automation and Test in Europe*, vol. 1, pp. 10246, 2004.

[TUR18] M. S. Turan, E. Barker, J. Kelsey, K. A. Mcay, M. L. Basish, M. Boile, “Recommendation for the Entropy Sources Used for Random Bit Generation,” *NIST Special Publication 800-90B*, 2018.

[VON51] J. Von Neumann, “Various Techniques used in Connection with Random Digits,” *National Bureau of Standards Applied Mathematical Series*, vol. 12, pp. 36-38, 1951.

[WEI02] E. W. Weisstein, “CRC Concise Encyclopedia of Mathematics,” *Chapman & Hall/CRC*, 2002.

[XIA05] D. Xiao, X. Liao, S. Deng, “One-Way Hash Function Construction Based on the Chaotic Map with Changeable-Parameter,” *Chaos, Solitons & Fractals*, vol. 24, no. 1, pp. 65-71, 2005.

3

Design of New Chaos-Based Stream Ciphers

In this chapter, several proposals of chaos-based stream ciphers are presented. In Section 3.1, the Logistic Map and the Skew Tent Map are introduced. Due to their implementation simplicity and good properties, all of the proposed stream ciphers have been based on one of these maps.

In Section 3.2, some of the problems that arise when a chaotic map is digitized are explained. In order to overcome these problems, several strategies have been proposed in the following sections: 3.3, 3.4, and 3.5. In each of these sections, one of these strategies is explained in detail and several stream ciphers based on this strategy are implemented in a Virtex 7 FPGA. Each of these Sections includes a subsection where the implementation results of the proposed algorithms are presented and a subsection where the NIST randomness tests results as well as other security-related aspects are discussed to determine if any of the proposed algorithms are secure.

In Section 3.6 the proposed stream ciphers implemented in the previous sections are compared in terms of throughput, FPGA resources and security. Based on this comparison, the best of the algorithms have been implemented in an ASIC, using a Taiwan Semiconductor Manufacturing Company (TSMC) 0.18-μm technology. The implementation results including the area and the power consumption are presented in this section.

3.1. Simple maps: Logistic Map and Skew Tent Map

As explained in Section 2.2.2, the most common model of chaos-based stream ciphers consists on a digitization of a chaotic map of the form shown in (2.9). Often, stream ciphers are used in applications that require a high throughput while using a relatively small amount of resources. In these cases, it is ideal to use very simple maps. In our work, we have restricted to one-dimensional chaotic maps that depend on a single control parameter (γ):

$$x_{i+1} = f(x_i, \gamma) \tag{3.1}$$

One of these systems that has been widely used in cryptography applications is the Logistic Map (LM), where the sequence is generated using the following equation:

$$x_{i+1} = f(x_i) = \gamma x_i(1 - x_i) \tag{3.2}$$

where $x_i \in [0, 1]$ and $\gamma \in [0, 4]$.

Unfortunately, this map along with other simple chaotic maps can often present periodic windows (i.e., certain initial conditions that lead to periodic orbits). As an example, Fig. 3.1 shows a bifurcation diagram of the LM, which represents several consecutive values of x_i as a function of the control parameter γ. As it can be seen, for small values of γ, the orbits stay at a fixed point. By increasing its value, orbits of period 2, 4, 8, … are generated until for values of $\gamma > 3.57$, the system starts to exhibit a chaotic behavior. However, even for values of $\gamma > 3.57$, it can be seen in Fig. 3.1 that there are still some white regions, which means that there are still some values of γ that lead to periodic (non-chaotic) orbits. Even in the regions that in the figure appear to be chaotic, it has been proven that there exist an infinite number of initial conditions that lead to periodic orbits [FEL12]. Although some methods have been proposed to find this initial conditions that lead to non-chaotic behavior, the whole set of parameters that lead to non-chaotic behavior is yet unknown [TUC09, GAL15]. This can be an important issue if an LM is directly used in a cryptosystem because the key space (that is given by the possible control parameters) can be difficult to define to avoid choosing unsecure keys (control parameters that lead to periodic windows).

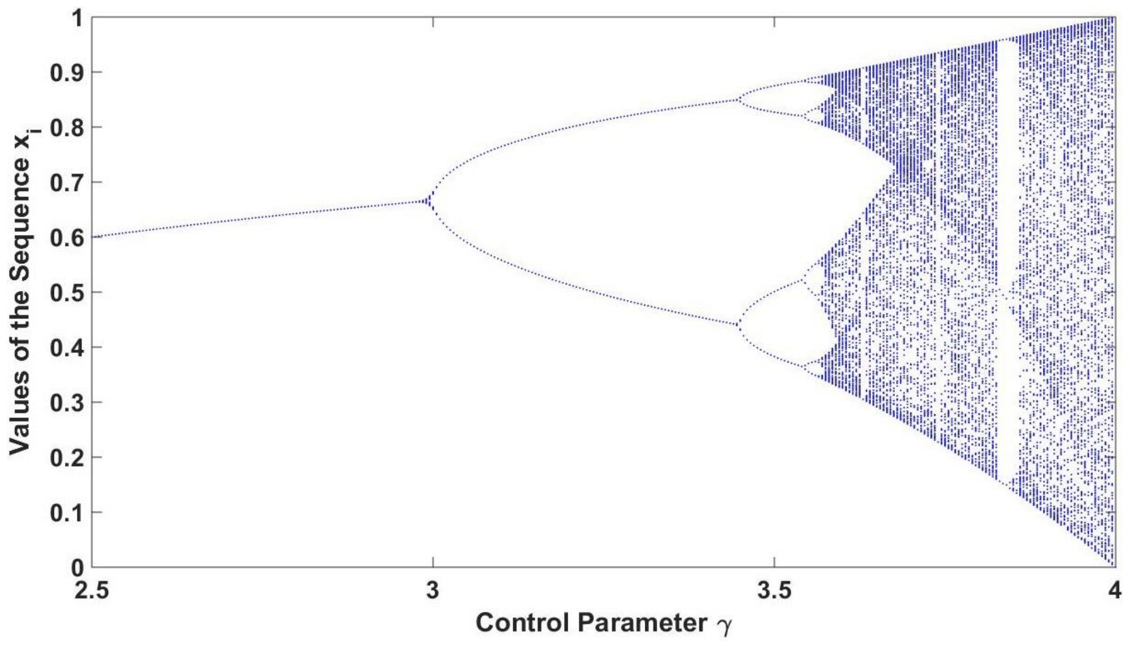


Fig. 3.1. Bifurcation diagram of the LM.

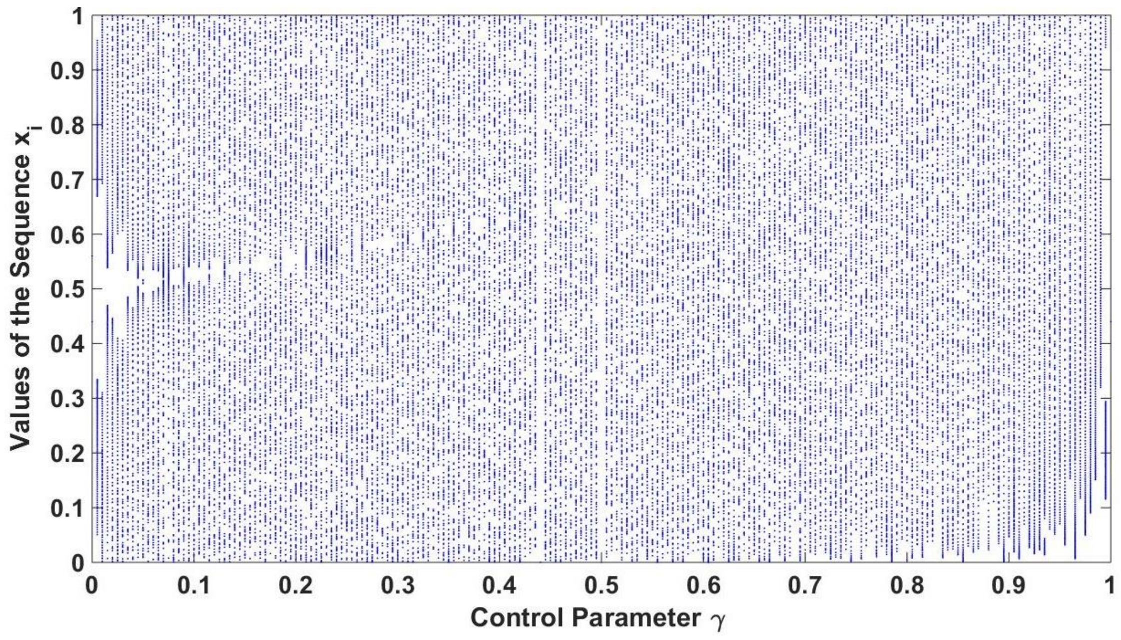


Fig. 3.2. Bifurcation diagram of the STM.

To avoid this problem, it is advisable to use chaotic maps where it is possible to find a continuous interval of initial parameters where any chosen value generates a chaotic orbit. This way, any key chosen in this range can be considered secure. One of these maps is the Skew Tent Map (STM), given by:

$$f(x_i) = x_{i+1} = \begin{cases} x_i/\gamma & x_i \in [0,\gamma] \\ (1-x_i)/(1-\gamma) & x_i \in (\gamma,1] \end{cases} \tag{3.3}$$

where $\gamma, x_0 \in (0,1)$. This map exhibits a chaotic behavior for any value of x_0 and γ [BAR95]. This can be seen by representing several consecutive values of x_i generated by different values of γ (Fig. 3.2). In opposition to Fig. 3.1, (LM) where a bifurcation diagram could be seen, in Fig. 3.2, for almost all values of γ, the orbits are chaotic and are approximately uniformly distributed in the interval $(0,1)$.

However, even if the orbits are not periodic, the STM along with other maps are often far from being ideal random functions and not all the possible values of x_i are generated with the exact same probability. As a consequence, statistical relationships between the ciphertext and the plaintext can be leaked (see the cryptanalyses in [ALV03, ALV04, LI07]).

3.2. Digital implementation of chaotic maps

To meet the throughput and resource requirements, apart from using simple chaotic systems they must be digitized using a precision of not too many bits. Under these conditions, due to round-off and truncation errors, the behavior of these systems differs from the ideal one [KOC06, OTE07, GAL13]. Although the shadowing lemma [BOW75] proofed that, in some chaotic systems, a pseudo-orbit (orbit generated using a finite precision) stays uniformly close to a true chaotic orbit [OTT93, PHA95], it has been shown later that the dynamics of a chaotic system are degraded to some degree [PER12, LI19].

The main consequence of the digitation is that, due to the fact that there is a finite number of different possible values for each element x_i and each x_i univocally determines the next element, x_{i+1}, the generated orbits have a maximum period of 2^n, where n is the precision used (number of bits used to represent each x_i). Furthermore, the typical periods of the generated orbits are much smaller than the maximum possible value. If f is a perfect random map, it was proofed in [HAR60] that the mean period of the generated orbits scales with the precision as $\bar{T}{\sim}2^{n/2}$. In 1988, a similar result was obtained assuming that f was an Ikeda map (chaotic): $\bar{T}{\sim}2^{nd/2}$, where d is the dimension of the map [GRE88]. Moreover, in both [HAR60] and [GRE88] it was shown that most of the periods are much shorter than $\bar{T}$. In general, although it has not been strictly proven, it can be assumed that, since the chaotic systems are considered to have random-like behaviors, their average periods will have a similar distribution as the one shown for an ideal random map in [HAR60].

As shown by these results, typical precisions used such as 32 or 64 bits cause that the generated sequences have short average periods. As a consequence, the sequences generated by these maps fail the randomness tests, regardless of the algorithm used. As an example to illustrate this, the NIST randomness tests for a sequence generated by a 32-bit STM using the LSB of each x_i is shown in Fig. 3.3. The list of applied tests with their numeration is detailed in Appendix A: Moreover, regardless of the randomness tests results, it is clear that, in order to be secure, the keystream generated by a stream cipher should not repeat itself. However, a sequence of period 2^{32} transmitted at a speed of 1 Gbps would repeat itself after only 4 seconds.

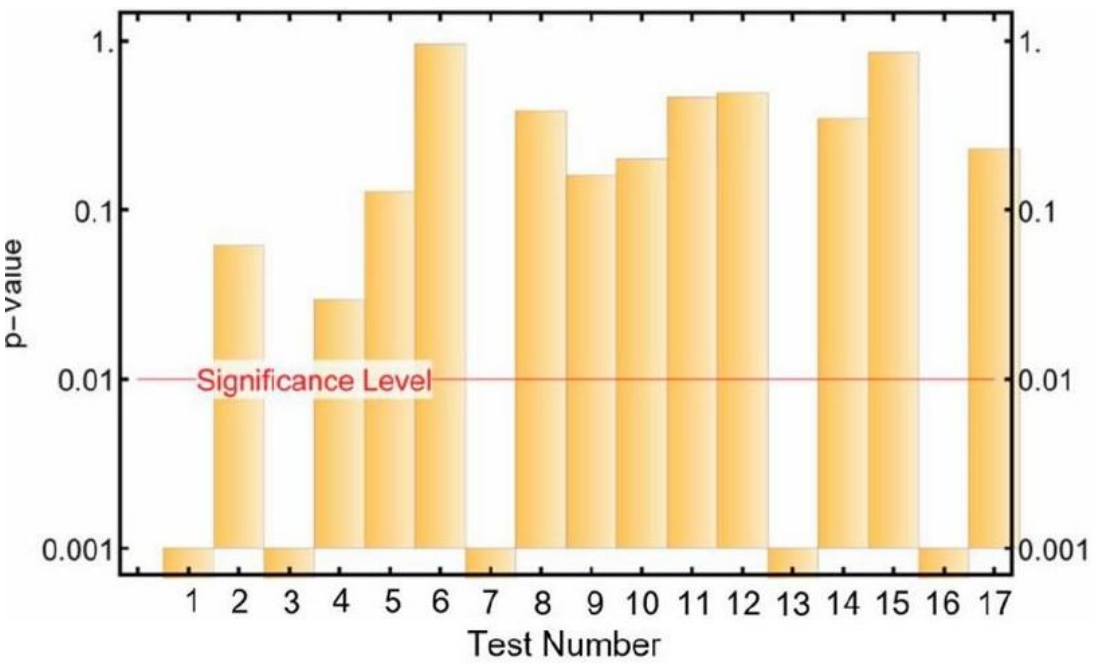


Fig. 3.3. NIST tests for a sequence generated by a 32-bit STM, extracting the LSB of each x_i.

A common approach to minimize this problem consists of using longer digital words. For example, [MAC17] uses a precision of 500 bits. However, this strategy supposes to increase the complexity of the system considerably and does not guarantee that all the generated sequences have long enough periods. Since, as explained before, it is our purpose to design stream ciphers that can achieve high encryption speeds using a small amount of resources, we have proposed several solutions that use different strategies to increase the periods and randomness of the sequences generated by simple chaotic algorithms: one strategy consists of combining several sequences using a Deficit Weighted Round Robin (DWRR) algorithm; other strategy consists of changing the control parameter γ dynamically; the last strategy consists of using an LFSR to perturb the chaotic orbits. In the next sections all these approaches will be explained and several stream ciphers based on these strategies will be proposed.

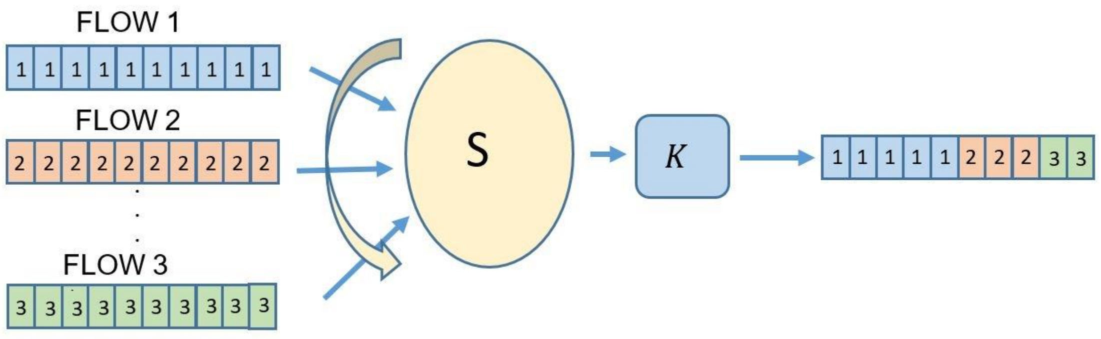
3.3. Using a multi-encryption scheme based on DWRR

This strategy consists of using a multi-encryption system where different parts of the same data stream are encrypted by different stream ciphers. A bandwidth percentage of the plaintext stream is assigned to different cipher modules, so that the final encrypted data stream is a mixture of different ciphertexts. This way, possible uncertainty about the nature of the encryption system is introduced to the attacker, making it more difficult to extract statistical information on the final ciphertext.

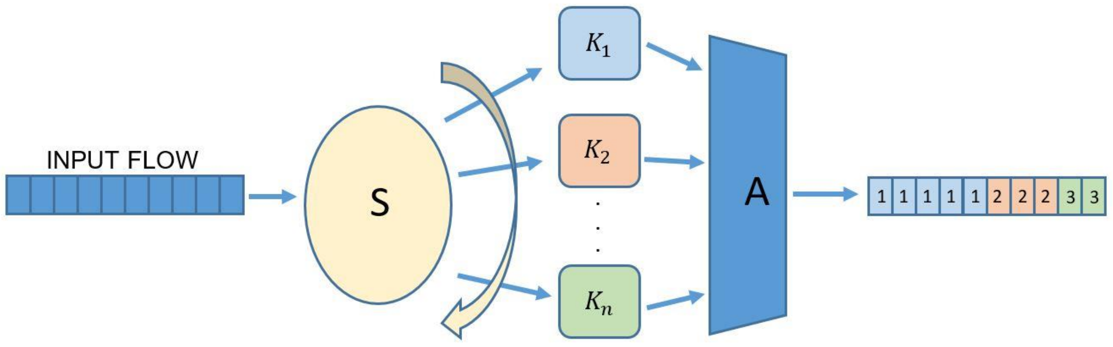
The final system consists of three different parts: a scheduling module, a bank of stream cipher modules and an aggregator of the streams generated by the ciphers (Fig. 3.4b). Traditionally, a scheduler is a generic structure that can be used to process several information flows in communication systems (Fig. 3.4a).

In this work, the scheduler module (S block in Fig. 3.4b) is responsible of distributing the information among each cipher according to the percentage of bandwidth that has been configured.

Both the transmitter and the receiver share the same information about bandwidth distribution and their allocation algorithm works with the same policy. The system is prepared for receiving a discontinuous stream of data, where data is grouped in frames of different bit size. According to the bandwidth distribution, frames are scheduled to the different cipher modules inside the cipher bank. This behavior could make this system well suited for encrypting purposes in packet switched networks.



(a)



(b)

Fig. 3.4. (a) Traditional use of a scheduler in a communication system. (b) Proposed multi-encryption system with *n* cipher modules (K_i), a scheduler and an aggregator block.

The allocation algorithm is based on a DWRR algorithm [SHR96] as it is suitable for distributing traffic bandwidth among several entities, in this case, cipher modules. For this purpose, a fixed quantum Q_i and a variable deficit counter DC_i are assigned to each module. The algorithm goes over each module sequentially. If the size of a frame, L_i, is smaller than DC_i, that frame is ciphered by the ith module and an amount L_i, is subtracted from its deficit counter ($DC_i' = DC_i - L_i$). Then, the size of the next frame is compared to the updated deficit counter DC_i' and the same procedure is repeated until the size of a new arrival frame is bigger than the ith deficit counter. When this condition happens, the deficit counter is increased by an amount Q_i ($DC_i' = DC_i + Q_i$) and the size of the frame is compared to the deficit counter of the next module, DC_{i+1}. This process is applied sequentially to each cipher module in a round-robin fashion.

The resulting bandwidth percentage assigned to each module, BW_i, will be proportional to its quantum value Q_i. Therefore, bandwidth can be distributed in terms of percentage depending on the quantum values assigned to each module by the scheduler.

After the scheduler has distributed different frames to the different stream ciphers in the cipher bank (K_1, K_2, … K_n, in Fig. 3.4b), and they have been encrypted, all the streams are combined by the aggregator (A block in Fig. 3.4b), giving as a result the final ciphertext.

At the receiver side, the scheduler works in the same way as in the transmitter, frames are distributed in the same way, and they are decrypted by the correct decipher. As bandwidth information must be known by the transmitter and the receiver, not only the keys of each stream cipher must be shared but also their quantum values.

3.3.1. Implementation results

In order to test the utility of the proposed multi-encryption scheme, a stream cipher composed by eight modules has been proposed and implemented in a Xilinx Virtex 7 FPGA [GAR17a, GAR18a]. Each module consists of a 32-bit STM-based stream cipher where only the least significant bit (LSB) of each x_i (x_i^0) is used to form the keystream (Fig. 3.6). Each module has different initial parameters γ, x_0. To obtain a better mixing, small frame sizes have been chosen and similar quantum values have been assigned to each module.

In this case, since all the ciphering modules use the same state update function (STM), part of the hardware can be reused, so that the required amount of area is not multiplied by the number of modules. For that purpose, the state and the key of each cipher are saved using memory resources, which in an FPGA can be RAM blocks or registers (Fig. 3.5). In our implementation, a RAM block configured

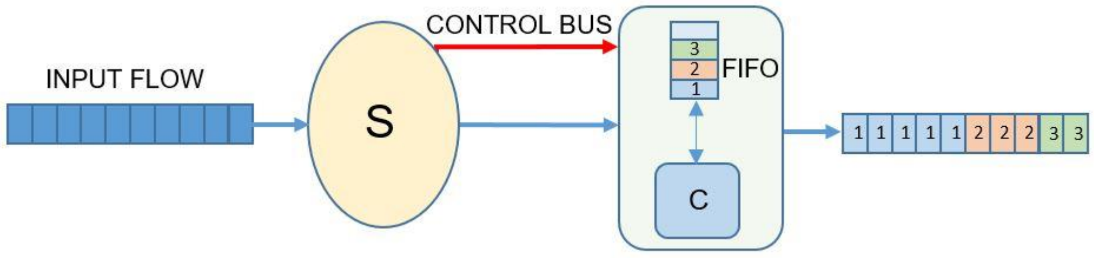


Fig. 3.5. Proposed dynamically configurable cipher structure when all the cipher modules use the same encryption algorithm.

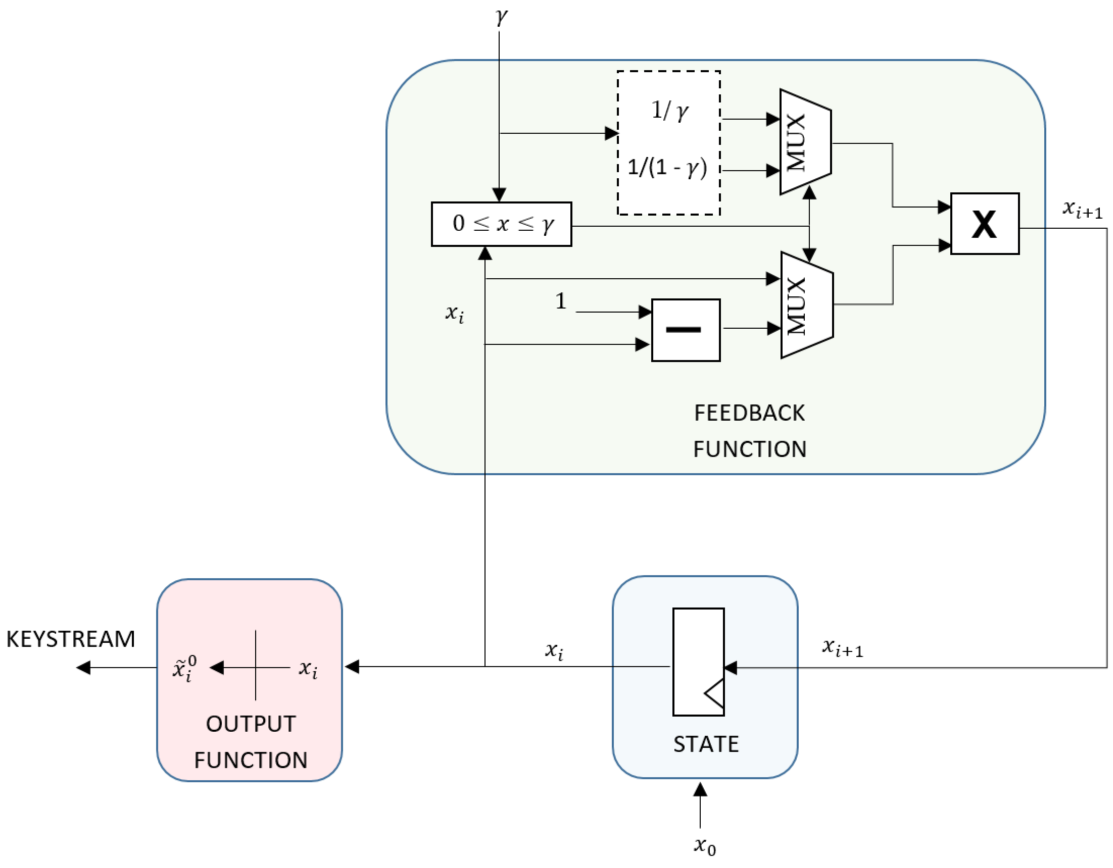


Fig. 3.6. Block diagram of a Skew Tent Map generator.

as First In First Out (FIFO) shift register was used in prevision of a substantial increment in the number of ciphers in future implementations. The general scheme of the optimized implementation of this system is shown in Fig. 3.7, where, in this case, the stream cipher feedback and output functions are the same as in Fig. 3.6.

As shown in Fig. 3.7, in the proposed structure it is possible to reload both registers (state and key) depending on which cipher is being used by the scheduler module. The ciphering profile for each cipher is composed of the pair state/key and is stored in a FIFO block that works as a circular buffer. When a scheduler needs to change the cipher module, load enable, read enable and write

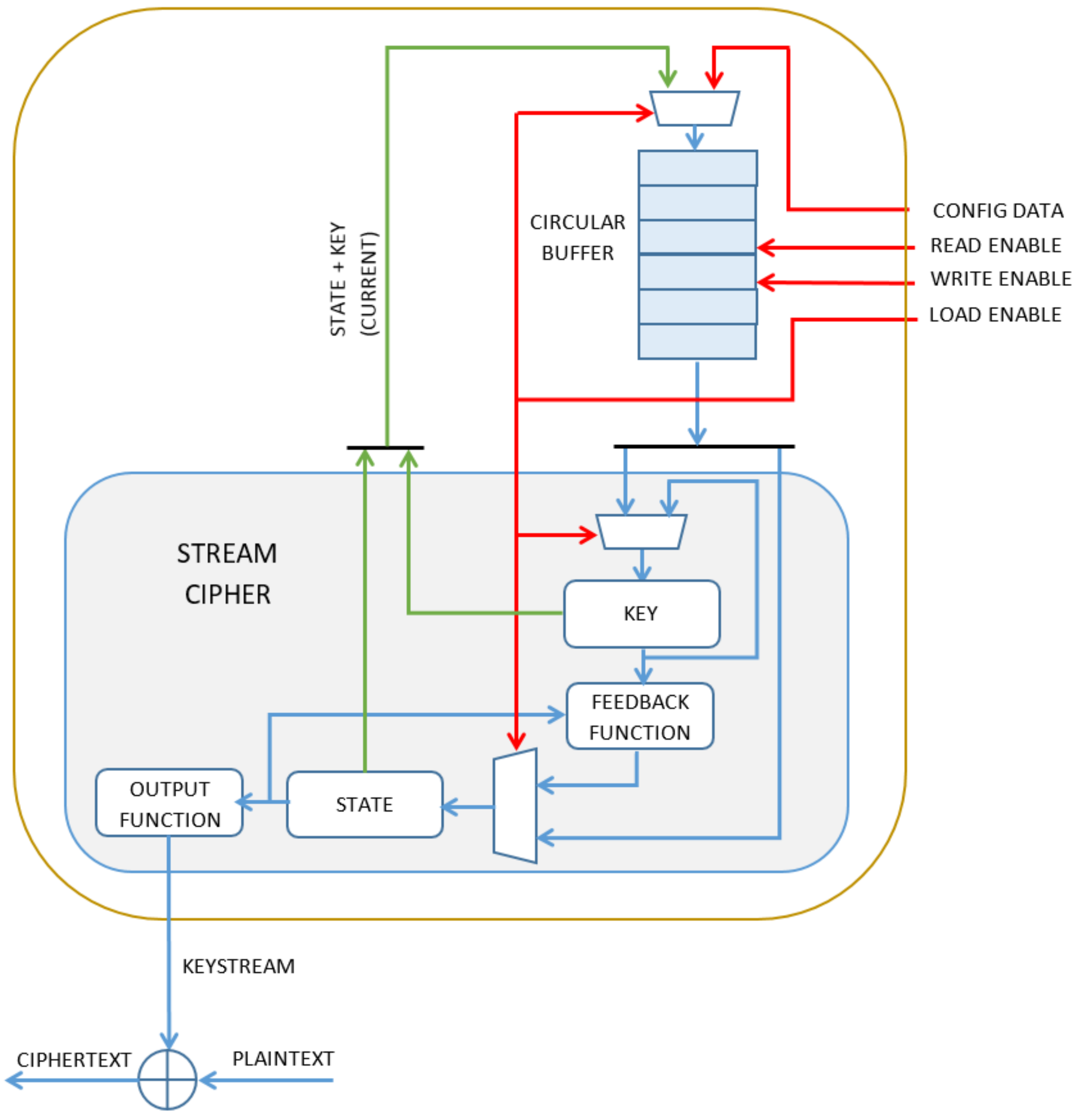


Fig. 3.7. Proposed configurable cipher structure.

enable inputs are asserted to load the new ciphering profile (state/key of the next cipher) and to store the current state/key of the last cipher in the circular buffer. This way, due to the fact that this cipher module can be dynamically configured, a single keystream is generated. Therefore, an aggregator block is not needed in this implementation, saving resources. In order to do the initial configuration of each cipher, ciphering profile FIFO can be written sequentially using *config data* word as input, asserting write enable and de-asserting load enable inputs. This action is taken by the scheduler when the user wants to change ciphering profiles.

Table 3.1 compares the resources used by the implementation of a single STM stream cipher, a cipher bank of eight STM generators and the proposed dynamically configurable STM generator. Moreover, the resources used by the scheduler separately are included in the last column. Since the purpose of the work published in [GAR17a, GAR18a] was to study the randomness enhancement of this strategy, the stream ciphers were not synthetized at the maximum possible frequency. In order to have a fair comparison with the rest of the proposed stream ciphers in the next sections, the same operating frequency

Table 3.1. Implementation results in a Virtex 7 FPGA

Configuration	**Single STM**	**8 STM**	**8 STM Configurable**	**Scheduler**
Number of stream ciphers	1	8	1	-
LUTs	369	2952	459	342
Registers	39	312	39	399
Slices*	98	777	119	136
Total DSPs	11	752	16	0
RAM blocks	0	0	1	1
Frequency (MHz)	134	134	134	134
Encryption rate (Mbps)	134	134	134	-
Mbps/slice	1.367	0.172	1.126	-

*The number of slices has been estimated from the number of registers and LUTs assuming unrelated logic of 134 MHz has been included in all the implementations. As it can be seen, by using one RAM block to save the states and the keys (x_i, γ) of each module, most of the hardware is reused, so there is only a small increase in the number of Lookup Tables (LUTs). While ciphering the last bit of the current frame, the configurable cipher updates its next state to be ready to cipher the next bit frame so that the throughput is the same. Unfortunately, the throughput of this stream cipher is not high enough for Gigabit Ethernet Communications.

3.3.2. Security analysis

With the proposed encryption system, 20 sequences of 1 million bits have been generated and have been subjected to the NIST randomness tests [RUK10] with a significance level of $\alpha = 0.01$. Ten of them have passed all these tests while the rest of them have failed one or more tests (see Fig. 3.8). In an ideal random sequence, the probability of passing each test would be 99% so the probability that a sequence passes all 17 test (assuming that the tests are independent) would be $0.99^{17} \approx 0.84$. Although the randomness improvement is considerable with respect to using a single 32-bit STM (Fig. 3.3), the percentage of sequences passing all the tests is a bit lower than this theoretical value. Moreover, by increasing the frame sizes, we have found that the percentage of sequences passing all the tests gradually decreases.

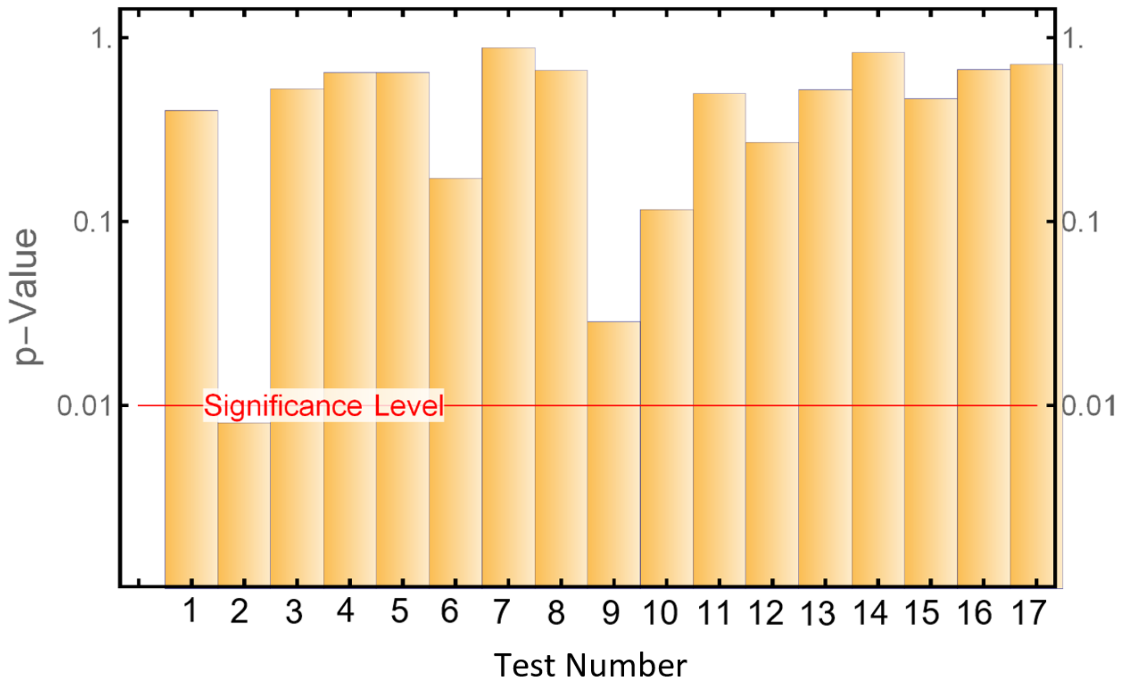


Fig. 3.8. NIST test results for a keystream generated by 8 STM modules combined with a DWRR algorithm.

The key size in this case is given by the total possible initial parameters x_o, control parameters γ and the quantums Q_i. In this case, where 8 modules have been used and all the parameters have been implemented using 32 bits, the size would be: $\kappa = 2^{32*8} \times 2^{32*8} \times 2^{32*8} = 2^{768}$. This size is much bigger than the minimum recommended size, 2^{112} [BAR11, ENI14].

Unfortunately, the sensitivity of the system on the key is not very high. Since the final keystream is composed by several independent keystreams, if only the parameters of one module are changed, only part of the keystream will change. Furthermore, if the quantum values were obtained by an attacker, it could be possible to determine which portion of the keystream has been encrypted by each module. Since each module independently is unsecure, the attacker could divide the problem into attacking 8 independent vulnerable modules.

In summary, the non-ideal randomness results, combined with the lack of high sensitivity on the key and the fact that the keystream could be separated into independent unsecure keystreams, makes us believe that, although this strategy undoubtedly improves the security of the systems, it does not guarantee full security.

3.4. Changing the chaotic control parameter dynamically

Let's consider that we have a stream cipher that, in order to generate the keystream, uses a chaotic map of the form (3.1). We propose to use, instead of a single value of γ, a set of *l* different values of γ: $\gamma_1, \gamma_2, \dots \gamma_l$. In order to generate the sequence $\{x_i\}$, the value of γ used is continuously changing in a circular way, according to a predefined sequence partition $\{s_i\}$. This way, the first s_1 elements of the sequence are generated as $x_i = f(x_{i-1}, \gamma_1)$, the next s_2 elements are obtained as $x_i = f(x_{i-1}, \gamma_2)$ and so on. When all γ_i have been used, and a total of $\sum_{i=1}^{l} s_i$ elements have been generated, the first value of γ, γ_1 is used again and the process keeps going in a circular fashion.

In the following subsections, we propose and analyze different approaches of choosing the values of *l* and $\{s_i\}$ in order to optimize the performance of the system.

3.4.1. Fixed value of sequence partition *s*

The first approach consists of generating the same number of elements using each γ_i, i. e., $s_1 = s_2 = \cdots s_l = s$. In the simplest case, where only a single value of γ is used, $l = 1$, we have that, for each value of x_i we always end up in one of the 2^n possible values of x_{i+1} so, as explained in Section 3.2, the maximum period is $T = 2^n$. However, if several values of γ are used, we can end up in *l* different values of x_{i+1} (one for each γ_i). This way, the theoretical maximum period of the generated sequences using this method is increased by a factor of *l*, $T(l) = l2^n$. Along this increment, it can be found experimentally that there is a considerable increment in the mean period length.

The main problem of this approach is that, by using always a fixed value of *s* and *l,* there is always a possibility that the sequences stay in a stable short period orbit. In order to avoid this problem, a possible alternative consists on changing constantly the values of s_i (Section 3.4.2).

3.4.2. Dynamically changed value of sequence partition *s*

In this case, a variable number of s_i is generated within a range $s_i \pm \Delta s_i$ and is used to encrypt the message. When all the values of γ_i have been used, a new value of s_1 is generated. This way, the sequences will conserve the characteristic properties of the chaotic map (ergodicity, high sensitivity to the initial conditions, …), but they will not stay at a stable orbit.

Since what is relevant in this method is to introduce small variations in the values of s_i, it is not necessary to use a perfect random number generator. Therefore, any random number generator, or even a simple algorithm that uses some parts of the plaintext or other parameters of the traffic could be used. For example, in a packed stream, the interframe gap or the packet size could be used.

3.4.3. Implementation results

To do a first test of this system, a 32-bit STM has been used as the state update function and only the LSB of each x_i (x_i^0) has been used to generate the keystream (Fig. 3.6) [GAR18b, GAR18c, GAR18d]. This map has been chosen because, as explained in Section 3.1, it is a simple system that presents a chaotic behavior for any initial value of x_0 and γ (i.e., it does not present periodic windows) and it can produce values of x_i that are uniformly distributed in the interval $(0,1)$. Therefore, after implementing this system on the FPGA, most of the problems of the generated sequences will be caused by the digitization of the system and not by the intrinsic properties of the chaotic map. Therefore, by proving that this method is capable of improving the randomness of the sequences generated by an STM-based algorithm, it is possible to extrapolate the results to other chaotic cryptosystems.

In order to test our proposed method, we have generated the sequences using different values of *l* (number of γ_i) and s_i (number of bits encrypted with each γ_i). A scheme of the complete implementation of the enhanced stream cipher that uses our randomness improvement technique and the STM generator is shown in Fig. 3.9.

First, we have started using fixed values of s_i ($s_1 = s_2 = \cdots = s$). In order to test the randomness of the sequences generated for each case, 100 sequences have been generated and have been subjected to the NIST randomness tests with a

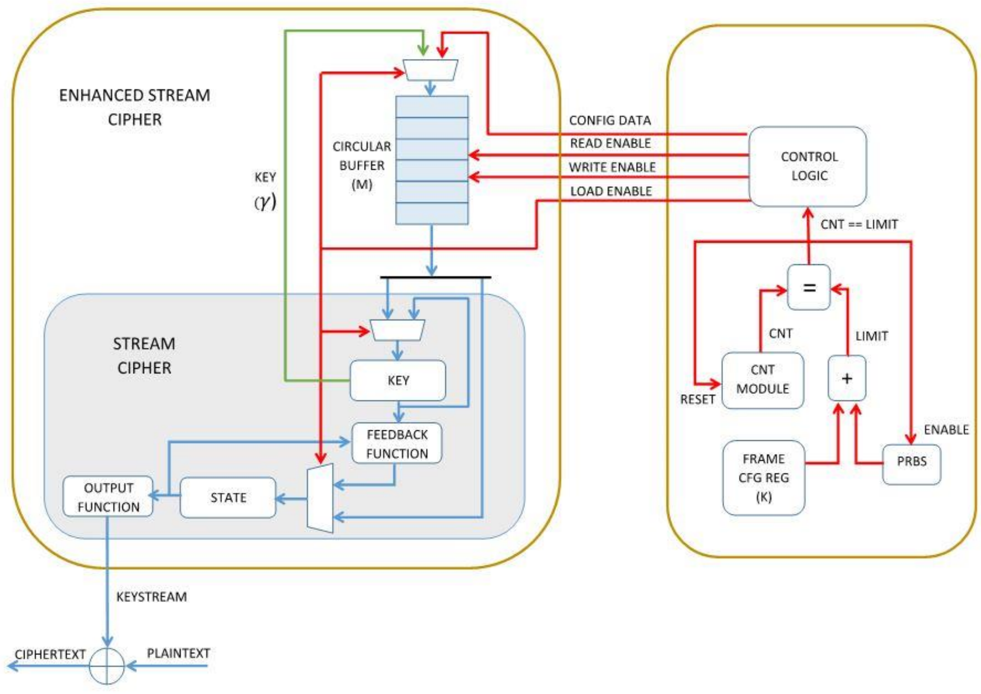


Fig. 3.9. Scheme of the proposed enhanced stream cipher. The values of γ_i are stored in a FIFO that works as a circular buffer. The control module performs the gamma change every time an amount of bits equal to the configured packet size plus a percentage is transmitted. This bit number limit is set thanks to the LIMIT value, which corresponds to the sum of the packet size value configured in FRAME CFG REG plus a pseudorandom value generated by the PRBS.

significance level of 0.01. To easily compare the results among different systems, the passing rates for each test (fraction of sequences that have passed each test) have been calculated and, after that, we have calculated the average of all of them. Results are shown in Table 3.2. For comparison purposes, the table also includes the results obtained by a system using only one value of γ ($l = 1$) but 32-bit (single) and 64-bit (double) precision.

As it can be seen, increasing *l* improves the randomness of the generated sequences. This can be explained by the fact that, by increasing the value of *l*, the maximum and the mean period of the sequences increases. This effect is especially noticeable for small values of *l,* since the effect of the increment of the period lengths of the sequences in the randomness test performance is bigger for small periods.

Table 3.2. NIST test results for fixed values of the partition sequence *s*

s	**10**	**100**	**1000**
l			
1 (32-bit)	0.317	0.317	0.317
1 (64-bit)	0.989	0.989	0.989
2	0.512	0.659	0.750
4	0.552	0.720	0.845
8	0.568	0.777	0.856
16	0.581	0.786	0.869

Table 3.3. NIST test results for variable values of the partition sequence *s*

s	**9-11**	**90-110**	**900-1100**
l			
1 (32-bit)	0.317	0.317	0.317
1 (64-bit)	0.989	0.989	0.989
2	0.938	0.935	0.857
4	0.971	0.932	0.910
8	0.964	0.944	0.929
16	0.955	0.937	0.901

On the other hand, by increasing the value of s, the randomness of the sequences also improves. This could be explained by the fact that, when a small value of s is used, groups of bits generated using the same value of γ are separated by a small distance inside the keystream. Therefore, the NIST tests might find some correlations among them. However, for bigger values of s, bits generated using the same γ are more separated so it becomes harder to find those correlations.

It can be noticed that, although the randomness improvement is significant, with the significance level of 0.01, 99% of the sequences should pass the randomness tests. Therefore, the sequences generated using this method are far from random. Although the results could be improved using bigger values of *l* and s, it would involve an increment of the hardware resources used.

In the second case, we have used the same values of γ_i as before but each s_i is a random integer generated within a certain range. To generate these random integers, a simple linear congruential algorithm has been used. Table 3.3 shows the results for different ranges. In this table, we have also included the results obtained by a system using only one value of γ ($l = 1$) but 32-bit (single) and 64-bit (double) precision.

As it can be seen, the randomness tests results are much better than for the fixed-s case since, as it has been mentioned before, this method helps the system to avoid stable orbits.

In the best case, for $l = 4$ and $s_i \in [9, 11]$, the randomness improvement is very similar to the improvement obtained by using a 64-bit precision. However, implementing the STM using a 64-bit precision requires far more hardware than implementing the 32-bit precision STM with the proposed randomness enhancement technique (Table 3.4).

Finally, to check that this randomness-improvement technique is valid in other chaotic maps, the same strategy has been tested but using the LM instead of the STM (also with a 32-bit precision and using only x_i^0 to generate the keystream) [GAR18e]. In this case, by using values of $l = 8$ and $s_i \in [9, 11]$ a NIST average passing rate of 0.989 has been obtained.

To summarize the results, a comparison in terms of resources and NIST average passing rate of stream ciphers based on the LM and the STM with and without the proposed strategy is presented in Table 3.4. Like as before, the main goal of the works published in [GAR18a, GAR18b, GAR18c, GAR18d, GAR18e] was to show the randomness improvement. Therefore, the operating frequencies presented in Table 3.4 have been adapted to 134 MHz to have a fair comparison with the other proposed stream ciphers. As it can be seen, the enhanced stream ciphers can obtain high security using a relatively small amount of resources. Unfortunately, the throughput is not high enough for Gigabit Ethernet communications. A possible way to improve the throughput could be to use a 64-bit implementation and transmit several bits of each x_i per cycle. In order to achieve a high throughput enough for 1 Gigabit Ethernet, the 8 LSBs of each x_i ($x_i^0, x_i^1, \dots x_i^7$) could be transmitted. However, we have checked by simulations that, with this approach, the generated sequences fail the NIST tests.

3.4.4. Security analysis

As explained in Section 3.4.3, using this proposed random-enhancement technique, the NIST average passing rate is considerably increased, from 0.317 to 0.971 in the case of the STM and from 0.252 to 0.989 in the case of the LM. Therefore, in both cases the randomness improvement is very high and the resulting enhanced stream cipher is able to generate almost ideal sequences.

In this case, the key space size would be given by all the possible values for the initial element of the sequence, x_0, all possible values of the chaotic parameters $\gamma_1, \gamma_2, \dots \gamma_l$ and, finally, all the possible sequence partitions, $n_{\{s_i\}}$. We do not provide a concrete value for $n_{\{s_i\}}$ since it may vary depending on the implementation. Therefore, the key space in this case, $\kappa_{Enhanced}$, would be given by: $\kappa_{Enhanced} = n_{\{s_i\}} \times 2^{32(l+1)} > 2^{32(l+1)}$.

It is clear that, by using several values of γ, the key space size is considerably increased. Providing that $l > 4$, the key space size would be $\kappa_{Enhanced} > 2^{112}$ which would be secure against brute-force attacks.

Table 3.4. Implementation results of the STM and the LM with and without the proposed randomness-enhancement technique

Configuration	**32-bit STM**	**64-bit STM**	**32-bit LM**	**64-bit LM**	**32-bit enhanced STM**	**32-bit enhanced LM**
LUTs	369	805	439	903	440	510
Registers	39	70	46	70	113	120
Slices*	98	210	116	235	139	143
DSPs	11	16	13	18	11	13
Frequency (MHz)	134	134	134	134	134	134
bits/cycle	1	1	1	1	1	1
Encryption rate (Mbps/s)	134	134	134	134	134	134
Mbps/slice	1.367	0.638	1.155	0.570	0.964	0.937
NIST average passing rate	0.317	0.989	0.252	0.979	0.971	0.989

*The number of slices has been estimated from the number of registers and LUTs assuming unrelated logic.

On the other hand, as long as the control parameters are chosen so that the system works in a chaotic regime, this system will present a high sensitivity in the key. It must be pointed out that, contrary to the strategy proposed in Section 3.3, with this strategy a change in a single parameter γ_i would affect the whole sequence since each element is obtained as a function of the previous one.

Finally, this strategy can be useful in chaotic maps where the key space is not well defined (the exact parameters that lead to chaotic behavior are not fully known). To prove this, let's call Pr_c the probability that a value of γ randomly chosen is "good" (i.e., an orbit generated by that γ exhibits a chaotic behavior) and $Pr_p = 1 - Pr_c$ the probability that the value of γ is "bad" (i.e., an orbit generated by that γ exhibits a periodic behavior). Usually, the probability of choosing an unsuitable value of γ will be very small (i.e., $Pr_p \ll Pr_c$). However, when this happens, the behavior of the cryptosystem will be very poor. Using this technique, since we are using *l* different values of γ, the probability of having *j* unsuitable values of γ among them will be:

$$Pr(l,j) = \binom{l}{j} Pr_p^j \left(1 - Pr_p\right)^{l-j} \approx \binom{l}{j} Pr_p^j \left(1 + (j - l) Pr_p\right) \qquad (3.4)$$

It can be seen that, by using this strategy, the probability of having at least one "bad" value of γ is increased.

$$\sum_{j=1}^{l} Pr(l,j) = 1 - Pr(l,0) = 1 - \left(1 - Pr_p\right)^l > Pr_p \qquad (3.5)$$

However, from (3.4) it can be seen that the probability of having *j* "bad" values of γ decreases very fast when increasing *j.* Therefore, even if one or some of the chosen values of γ are "bad", most of them will probably be good. Therefore, the overall behavior of the system will most likely not be as bad as in the case of using the chaotic map without this randomness enhancement technique.

In other words, in chaotic maps where the key space is not well defined, by using this technique, the probability of generating a non-ideal keystream (i.e., with at least one "bad" value of γ) would be higher, but the probability of generating a very poor keystream (i. e., with most of the chosen values of γ "bad") would be lower.

In conclusion, this strategy considerably increases the security of stream ciphers based on chaotic maps and, if the values of l and $\{s_i\}$ are chosen properly, it can be used to design perfectly secure cryptosystems. The best performing implemented stream ciphers (32-bit enhanced STM with $l = 4$, $s \in [9,11]$ and 32-bit enhanced LM with $l = 8$, $s \in [9,11]$) are capable of generating sequences with very good statistical properties and can be considered secure. However, as seen in Table 3.4, the LM requires a bit more resources to be implemented and, since not all possible values of γ are “good”, there is still a small possibility that all the chosen parameters of γ are bad so the resulting sequence is not random. Therefore, the 32-bit enhanced STM is a better solution.

3.5. Using an LFSR to perturb the chaotic orbits

This strategy consists of introducing small perturbations after each iteration of the map to increase the mean period and improve the randomness of the generated, sequences. After each iteration, the values of each x_i are changed ($x_i \rightarrow \tilde{x}_i$) and these modified values, $\tilde{x}_i$, are used to obtain the next values of the sequence as $x_{i+1} = f(\tilde{x}_i)$. In this proposal, the values of $\tilde{x}_i$ are obtained by XORing the least significant bit (LSB) of each x_i, x_i^0, with the least significant bit of an LFSR, y_i^0. Using this method, the period of the sequence can be set up to be above a given value using the following proposition, proven in [SAN98, KOC11].

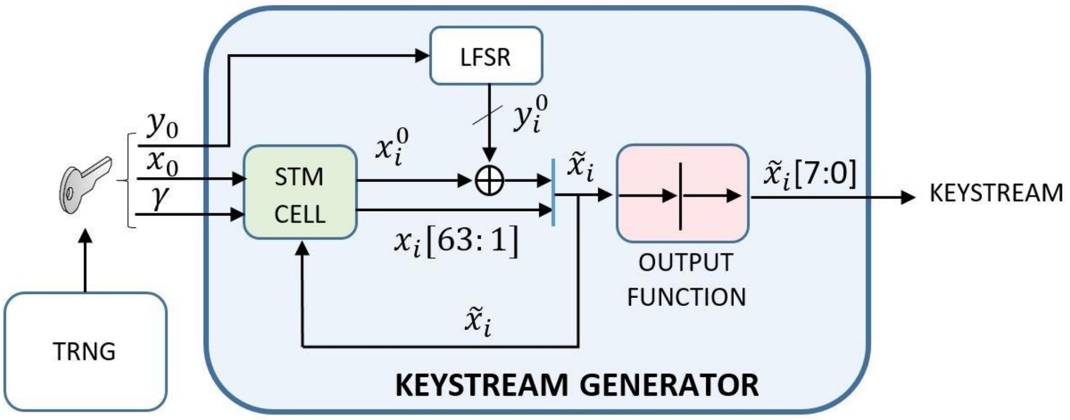
Proposition 3.1. Given the binary sequences $A_1(n)$, $A_2(n)$ and $A_3(n) = A_1(n) \oplus A_2(n)$, for $n \in \mathbb{N}$ and T_1, T_2 and T_3 their respective periods. If T_2 is prime, then $T_3 \geq lT_2$ with $l \geq 1$.

Therefore, if we consider $A_3(n)$ as the LSB of $\tilde{x}_i$, $\tilde{x}_i^0$, $A_1(n)$ as the LSB of x_i, x_i^0, and $A_2(n)$ as the output bits, y_i^0, of an LFSR with a prime period T_2, the period of the $\{\tilde{x}_i\}$ sequence will be $T_3 \geq T_2$. In our work, a 61-order LFSR has been used, which gives us a period of, at least, 2^{61}. This choice guarantees that the period is big enough for our purposes (a sequence with a period of 2^{61} transmitted at a speed of 1 Gbps would take more than 73 years to repeat itself).

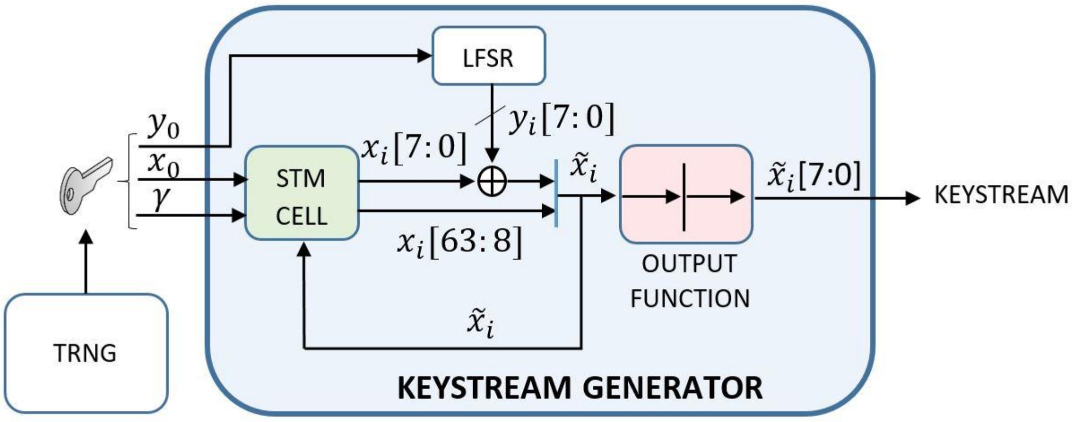
3.5.1. Implementation results

To test this strategy, 32-bit STM and LM functions have been implemented on a Virtex 7 FPGA and we have seen that, if only the LSB of each x_i, x_i^0, is used to form the sequences, the NIST tests are passed proving that the strategy considerably improves the randomness of the generated sequences [GAR16a, GAR16b, GAR17c, GAR17d].

Unfortunately, similarly to the previous stream ciphers, the obtained throughput is still too low to be used in Gigabit Ethernet communications. To obtain a higher throughput we have tried to combine several bits of each x_i with the LFSR and use several bits of each x_i to form the keystream but, unfortunately, in these cases the NIST test are failed. To be able to obtain a higher throughput, we propose to use a 64-bit STM and using the 8 least significant bits of each $\tilde{x}_i$ $(\tilde{x}_i^0, \tilde{x}_i^1, \dots, \tilde{x}_i^7)$ to form the keystream (Fig. 3.10a) [GAR17b].



(a)



(b)

Fig. 3.10. Proposed 64-bit STM-LFSR stream cipher. Version 1 (a) and version 2 (b).

Table 3.5. Implementation results of the 64-bit STM-LFSR

Algorithm	**64-bit STM-LFSR (V1)**	**64-bit STM-LFSR (V2)**
LUTs	810	817
Registers	135	135
Slices*	220	222
Total DSPs	16	16
RAM blocks	0	0
Frequency (MHz)	134	134
bits/cycle	8	8
Encryption rate (Mbps)	1072	1072
Mbps/slice	4.873	4.829
NIST average passing rate	0.985	0.989

This way, we have managed to pass the NIST test while obtaining a high throughput, as shown in Table 3.5. Since the NIST average passing rate is a bit lower than the theoretical value (0.99), an alternative version (64-bit STM-LFSR (V2)) has also been implemented where the last 8 LSBs of each x_i are XORed with the last 8 bits of the LFSR (Fig. 3.10b) [GAR19a, GAR19b]. As it can be seen, the NIST average passing rate is slightly better by just using a small amount of extra resources. It must be noticed that this approach (transmitting the 8 LSBs instead 1 and using a 64-bit implementation) has not worked with the previous strategies proposed in Sections 3.3 and 3.4 since their resulting sequences have failed the NIST tests.

3.5.2. Security analysis

As explained in Section 3.5.1, the proposed stream ciphers have been able to pass the NIST randomness, showing average passing rates very close to the ideal one. Therefore, the generated sequences are practically undistinguishable from truly random sequences.

The key space in this system is given by the total possible values of the control parameter γ and the initial states x_0 and y_0. Since a 64-bit implementation and a 61-order LFSR have been used, the total key space size is: $\kappa = 2^{64+64+61} = 2^{189}$.

Similarly as in the case of the stream ciphers proposed in previous Sections, the key size is also bigger than 2^{112}.

Finally, as explained in Section 3.1 the STM presents a chaotic behavior for any chosen value of x_0, γ. Furthermore, due to this chaotic behavior, by slightly changing the initial state of the LFSR the resulting orbits are completely different. Therefore, we can conclude that the system presents a high sensitivity on the key.

All the proposed stream ciphers proposed in the previous sections have been compared with this one in terms of resources, throughput and security. A comparison of the best stream ciphers implemented using each strategy is presented in Table 3.6. The 32-bit and 64-bit STM implementations are also included in the table to highlight the improvements that the proposed strategies have achieved. The security has been qualitatively addressed as “high”, “medium”, or “low” based on the NIST tests results along with the other considerations made in the “Security analysis” sections.

As it can be seen, the stream cipher implemented using this last method can clearly achieve a higher encryption rate per slice (Mbps/slice) while maintaining a high level of security.

Table 3.6. Comparison among several implemented stream ciphers based on different strategies

Configuration	**32-bit**	**64-bit**	**32-bit**	**32-bit**	**64-bit**
	STM	**STM**	**STM-Scheduler**	**Enhanced-STM**	**STM-LFSR (V2)**
LUTs	369	805	459	440	817
Registers	39	70	39	113	135
Slices*	98	210	119	139	220
DSPs	11	16	16	11	16
Frequency (MHz)	134	134	134	134	134
bits/cycle	1	1	1	1	1
Throughput (Mbps)	134	134	134	134	1072
Mbps/slice	1.367	0.638	1.126	0.964	4.829
Security	Low	Medium	Medium	High	High

*The number of slices has been estimated from the number of registers and LUTs assuming unrelated logic.

3.6. Final implementation

Finally, after determining that the last stream cipher offered the best performance in terms of security and throughput per slice, several implementation strategies have been tried in Vivado to obtain the best possible performance. The final implementation results are presented in Table 3.7 along with the implementation results of other chaotic stream ciphers found in the literature [PER19]. To have a fair comparison the configuration registers and LUTs have not been considered as part of the encryption algorithm, obtaining a considerable reduction with respect to the implementation shown in Section 3.5.1.

Table 3.7. Implementation results and comparison with other chaotic stream ciphers

Reference	**[DAB11]**	**[DAB12]**	**[PAN13]**	**[FRA17]**	**This work**
Main chaotic algorithm	LM	Bernoulli Map	LM	Hénon map	STM
LUTs	129	575	643	1600	734
Registers	64	108	160	64	65
Slices*	41	342	181	408	192
Total DSPs	16	9	16	16	16
Frequency (MHz)	26.9	36.9	93	25.7	174
Bits/cycle	1	0.2	16	1	8
Encryption rate (Mbps)	26.9	7.38	1488	25.7	1392
Mbps/slice	0.656	0.022	8.22	0.063	7.25

*The number of slices has been estimated from the number of registers and LUTs assuming unrelated logic.

As it can be seen, the proposed stream cipher clearly achieves a better performance in terms of throughput per slice than the ones presented in [DAB11, DAB12, FRA17]. Although the work presented in [PAN13] achieves a slightly higher throughput per slice, it transmits 16 bits of the internal state each cycle, which can result in a lower level of security.

Table 3.8. ASIC implementation of the 64-bit STM-LFSR algorithm

Algorithm	**64-bit STM-LFSR (V2)**
Technology (nm)	180
Area (mm^2)	0.19682
Gate equivalent (2-NAND)	19,682
Frequency (MHz)	125
bits/cycle	8
Encryption rate (Mbps)	1000
kbps/gate	50.8
Power at 125 MHz (mW)	24.1
Security	high

Finally, the proposed stream cipher has been implemented in a 0.18 μm CMOS technology with six metal layers provided by TSMC fed at 1.8 V (core) and 3.3 V (I/O). Some major constraints of the design were a maximum clock period of 14 ns, load ranging from 0.01 to 1.0 pf at outputs and drive up to 0.4 kΩ at inputs. This set of constraints have proven to be enough for this technology to satisfactorily implement the cipher, achieving an encryption speed of 1 Gbps using a total area of 0.197 mm^2 or ($\sim$20,000-NAND equivalent gates using an equivalence of 10 μm per 2-NAND [ART01]). At the maximum frequency of operation, its power consumption has been 24.1 mW (24.1 pJ/bit). This result is lower than typical implementations of AES such as the ones presented in [FEL05] and [HAM06]. The full implementation results are shown in Table 3.8.

These implementation results have been compared to other ASIC implementations of previously proposed chaotic stream ciphers in Table 3.9. Since other works do not provide information about some of the parameters in Table 3.8 (e.g., power consumption or area), Table 3.9 only focuses on 2-NAND equivalent gates and throughput. As it can be seen, the proposed algorithm performs slightly better than the works presented in [CHE10] and [CHA12] in terms of throughput/gate and achieved higher encryption speeds. Although a work improving the algorithm proposed in [CHA12] was presented in [LI12], this result has not been included in the table since it has not been implemented.

Table 3.9. Comparison with other chaotic stream ciphers implemented in ASIC

System	[CHE10]	[CHA12]	This Work
Technology (nm)	180	180	180
Gate equivalent (2-NAND)	9,622	10,218	19,682
Encryption rate (Mbps)	200	250	1000
kbps/gate	20.8	24.5	50.8

To characterize experimentally this chip, a development board Zybo has been used to feed the chip with the clock and reset signals through Pmod ports (Fig. 3.11) and a DSAV334A Infiium V-Series Oscilloscope has been used to capture the generated signals. As an example, a sample signal obtained at a clock frequency of 250 kHz is shown in Fig. 3.12. The captured signals have been later post-processed to extract the binary sequences and we have checked that, for different initial parameters (x_0, γ, y_0) and different clock frequencies, the obtained sequences match with the theoretical ones.

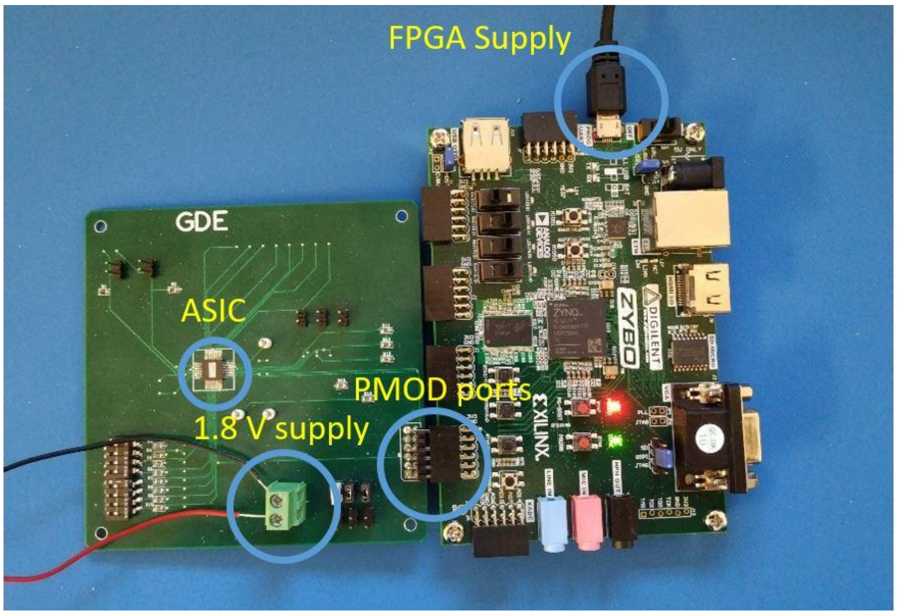


Fig. 3.11. PCB for the ASIC test setup connected to the Zybo board used to feed the clock and the reset signals. PMOD ports have been used for the connection.

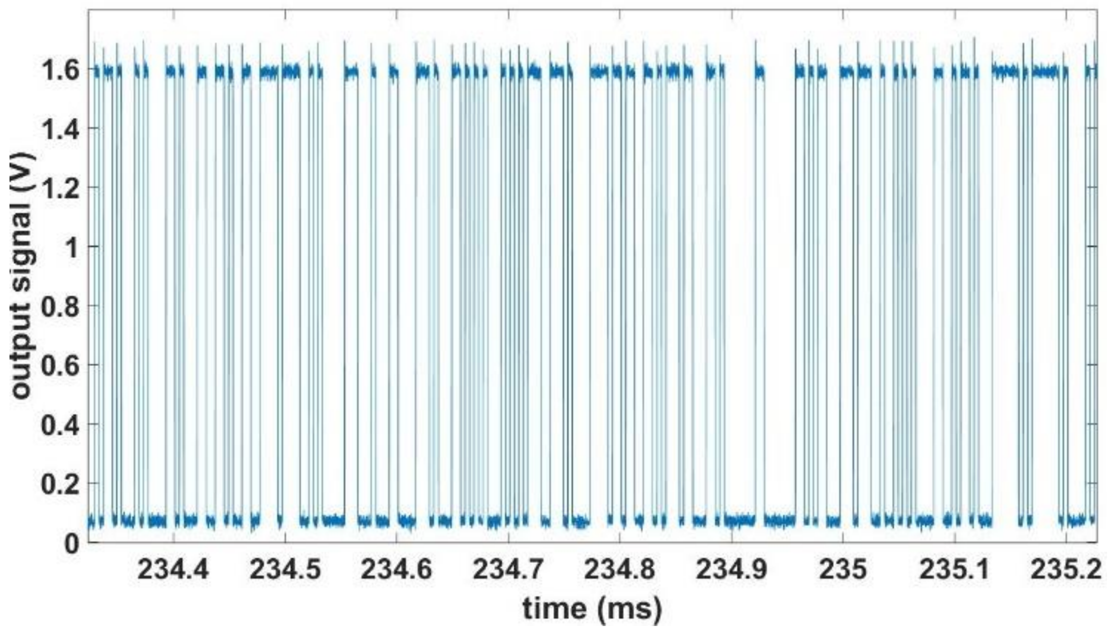


Fig. 3.12. Output signal at a clock frequency of 250 kHz.

This way, we have checked that the system works properly at all the measured frequencies up to 125 MHz, which is the maximum clock frequency that the FPGA was capable of supplying. Since with the chosen algorithm 8 bits are transmitted per cycle, the throughput is at least 1 Gbps and, therefore, the system could be suitable for 1 Gigabit Ethernet.

3.7. Conclusions

In this chapter, several stream ciphers based on chaotic maps have been proposed, implemented and analyzed in terms of throughput and security. These stream ciphers can be divided into three different groups depending on the strategy used to mitigate the digitization effects: using a multi-encryption scheme based on a DWRR, changing the chaotic parameter dynamically and using an LFSR to perturb the orbits. Among them, the stream ciphers that used this last strategy achieved better results. In particular, by implementing a 64-bit STM with a 61-order LFSR and extracting the 8 LSBs after each iteration, a high throughput and high security has been achieved.

This last algorithm has been implemented in a Virtex 7 FPGA achieving a throughput of 1392 Mbps (7.25 Mbps/slice), which is a considerable improvement with respect to other chaos-based stream ciphers implemented in FPGAs.

Furthermore, an ASIC implementation using a 0.18 μm CMOS technology has been made and tested. The system has worked perfectly at all clock frequencies up to 125 MHz (1 Gbps throughput), which is the maximum clock frequency that we could supply with the used FPGA board (Zybo board). Again, a comparison with other chaotic stream ciphers implemented in ASIC show that the proposed algorithm is a considerable improvement with respect to the state of the art.

3.8. References

[ALV03] G. Alvarez, F. Montoya, M. Romera, G. Pastor, “Cryptanalysis of an Ergodic Chaotic Cipher,” *Physics Letters A*, vol. 311, pp. 172-179, 2003.

[ALV04] G. Alvarez, F. Montoya, M. Romera, G. Pastor, “Breaking Parameter Modulated Chaotic Secure Communication Systems,” *Chaos, Solitons & Fractals*, vol. 21, no. 4, pp. 793-797, 2004.

[ART01] Artisan Components, “TSMC 0.18 μm Process 1.8-Volt SAGE-X^{TM} Standard Cell Library Databook,” *Artisan Components Inc*, 2001.

[BAR95] A. Baranovsky, D. Daems, “Design on One-Dimensional Chaotic Maps with Prescribed Statistical Properties,” *International Journal of Bifurcations and Chaos*, vol. 16, pp. 1585-1598, 1995.

[BAR11] E. Barker, A. Roginsky, “Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths,” *NIST Special Publication 800-131 A*, 2011.

[BOW75] R. Bowen, “Equilibrium States and the Ergodic Theory of Anosov Diffeomorphims,” *Lecture Notes in Mathematics*, vol. 470, 1975.

[DAB11] P. Dabal, R. Pelka, “A Chaos-Based Pseudo-Random Bit Generator Implemented in FPGA Device,” *Proceedings of the 14th International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pp. 151-154, 2011.

[DAB11] P. Dabal, R. Pelka, “FPGA Implementation of Chaotic Pseudo-Random Bit Generators,” *Proceedings of the 19th International Conference Mixed Design of Integrated Circuits and Systems (MIXDES)*, 2012.

[ENI14] ENISA, "Algorithms, Key Size and Parameters Report," Available online: www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-size-and-parameters-report-2014 (accessed on 25 February 2019), 2014.

[FEL05] M. Feldhofer, J. Wokerstorfer, V. Rijmen, “AES implementation on a grain of sand,” *IEE Proceedings – Information Security*, vol. 152, no. 1, pp. 13-20, 2005.

[FEL12] D. P. Feldman, “Chaos and Fractals. An Elementary Introduction,” *Oxford University Press*, 2012.

[FRA17] L. De la Fraga, E. Torres-Pérez, E. Tielo-Cuautle, C. Mancillas-López, “Hardware Implementation of Pseudo-Random Number Generators Based on Chaotic Maps,” *Nonlinear Dynamics*, vol. 90, no. 2, pp. 1661-1670, 2017.

[GAL13] Z. Galias, “The Dangers of Rounding Errors for Simulations and Analysis of Nonlinear Circuits and Systems-and how to Avoid them,” *IEEE Circuits and Systems Magazine*, vol. 13, no. 3, pp. 35-52, 2013.

[GAL15] Z. Galias, B. Garda, “Detection of All Low-Period Windows for the Logistic Map,” *Proceedings of the 2015 IEEE International Symposium on Circuits and Systems (ISCAS 2015)*, 2015.

[GAR16a] M. Garcia-Bosque, C. Sánchez-Azqueta, G. Royo, S. Celma, “Lightweight Ciphers Based on Chaotic Map-LFSR Architectures,” *Proceedings of the 12th Conference on Ph.D. Research in Microelectronics (PRIME)*, pp. 1-4, 2016.

[GAR16b] M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, “Secure Communication System Based on a Logistic Map and a Linear Feedback Shift Register,” *Proceedings of the 2016 IEEE International Symposium on Circuits and Systems (ISCAS 2016)*, pp. 1-4, 2016.

[GAR17a] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, S. Celma, “A New Multiple Ciphering Scheme for Improving Randomness,” *Proceedings of the 2017 European Conference on Circuit Theory and Design (ECCTD)*, pp. 1170-1173, 2017.

[GAR17b] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, S. Celma, “Application of a MEMS-Based TRNG in a Chaotic Stream Cipher,” *MDPI Sensors*, vol. 17, no. 3, pp. 1-15, 2017.

[GAR17c] M. Garcia-Bosque, A. Pérez, C. Sánchez-Azqueta, S. Celma “Fast and Secure Chaotic Stream Cipher with a MEMS-Based Seed Generator,” *Proceedings of the 2017 IEEE International Instrumentation and Measurement Technology Conference*, 2017.

[GAR17d] M. Garcia-Bosque, A. Pérez, C. Sánchez-Azqueta, G. Royo, S. Celma “MEMS-Based Seed Generator Applied to a Chaotic Stream Cipher,” *Proceedings of SPIE Microtechnologies*, 2017.

[GAR18a] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, S. Celma, “A Multi-Encryption Scheme and its Application to Improve the Random Properties of a Chaos-Based Stream Cipher,” *Proceedings of the 2018 International Symposium on Nonlinear Theory and its Applications (NOLTA 2018)*, 2018.

[GAR18b] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, S. Celma, “A New Randomness-Enhancement Method for Chaos-Based Cryptosystem,” *2018 IEEE 9th Latin American Symposium on Circuits and Systems (LASCAS)*, pp. 1-4, 2018.

[GAR18c] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, S. Celma, “A New Technique For Improving the Security of Chaos Based Cryptosystems,” *Proceedings of 2018 International Symposium on Circuits and Systems*, pp. 1-5, 2018.

[GAR18d] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, S. Celma, “A New Simple Technique for Improving the Random Properties of Chaos-Based Cryptosystems,” *AIP Advances*, vol. 8, no. 3, pp. 035004-1:035004-10, 2018.

[GAR18e] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, S. Celma, "Chaos-Based Bitwise Dynamical Pseudorandom Number Generator On FPGA," *IEEE Transactions on Instrumentation and Measurements*, vol. 68, no. 1, pp. 291-293, 2018.

[GAR19a] M. Garcia-Bosque, G. Díez-Señorans, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, S. Celma, "A 1 Gbps Chaos-Based Stream Cipher Implemented in 0.18μm CMOS Technology," *MDPI Electronics*, vol. 8, no. 6, pp. 1-10, 2019.

[GAR19b] M. Garcia-Bosque, G. Díez-Señorans, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, S. Celma, "A New Lightweight CSPRNG Implemented in a 0.18 □m CMOS Technology," *Proceedings of the 15th Conference on PhD Research in Microelectronics and Electronics (PRIME 2019),* 2019.

[GRE88] C. Grebogy, E. Ott, J. A. Yorke, "Roundoff-Induced Periodicity and the Correlation Dimension of Chaotic Attractors," *Physical Review A,* vol. 38, no. 7, pp. 3688-3692, 1998.

[HAM06] P. Hämäläinen, T. Alho, M. Hämäläinen, T. D. Hämäläinen, "Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core," *Proceedings of the 9th EUROMICRO Conference on Digital System Design (DSD'06)*, pp. 577-583, 2006.

[HAR60] B. Harris, "Probability Distributions Related to Random Mappings," *Annals of Mathematical Statistics,* vol. 31, pp. 1045-1062, 1960.

[KOC06] L. J. Kocarev, J. Szczepanski, J. M. Amigó, I. Tomovski, "Discrete Chaos-I: Theory," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 6, pp. 1300-1309, 2006.

[KOC11] L. J. Kocarev, S. Lian, "Chaos-Based Cryptography: Theory, Algorithms and Applications," *Springer-Verlag*, Berlin Heidelberg, 2011.

[LI07] C. Li, S. Li, G. Alvarez, G. Chen, K.-T. Lo, "Cryptanalysis of Two Chaotic Encryption Schemes Based on Circular Bit Shift and XOR Operations," *Physics Letters A*, vol. 369, pp. 23-30, 2007.

[LI19] C. Li, B. Feng, S. Li, J. Kurths, G. Chen, "Dynamic Analysis of Digital Chaotic Maps via State-Mapping Networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1-14, 2019 (early access).

[MAC17] J. Machicao, O. M. Bruno, "Improving the Pseudo-Randomness Properties of Chaotic Maps Using Deep-Zoom," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, pp. 053116-1 053116-14, 2017.

[OTE07] J. A. Oteo, J. Ros, "Double Precision Errors in the Logistic Map: Statistical Study and Dynamical Interpretation," *Physical Review E*, vol. 76, no. 3, pp. 036214, 2007.

[OTT93] E. Ott, “Chaos in Dynamical Systems,” *Cambridge University Press*, pp. 18-19, 1993.

[PAN13] A. Pandle, J. Zambreno, “A Chaotic Encryption Scheme for Real-Time Embedded Systems: Design and Implementation,” *Telecommunication Systems*, no. 52, pp. 515-561, 2013.

[PER12] K. J. Persohn, R. Povinelli, “Analyzing Logistic Map Pseudorandom Number Generators for Periodicity Induced by Finite Precision Floating-Point Representation,” *Chaos, Solitons Fractals*, vol. 45, no. 3, pp. 238-245, 2012.

[PER19] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, “Chaotic Encryption Applied to Optical Ethernet in Industrial Control Systems,” *IEEE Transactions on Instrumentation and Measurements*, 2019 (early access).

[PHA95] S. C. Phatak, S. S. Rao, “Logistic Map: A Possible Random-Number Generator,” *Phys. Rev. E,* vol. 51, no. 4, pp. 3670-3678, 1995.

[RUK10] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” *NIST Special Publication 800-22 Rev.1.a,* 2010.

[SAN98] T. Sang, R. Wang, Y. Yan, “Perturbance-Based Algorithm to Expand Cycle Length of Chaotic Key Stream,” *Electronics Letters*, vol. 34, pp. 873-874, 1998.

[SHR96] M. Shreedhar, G. Varghese, “Efficient Fair Queuing Using Deficit Round Robin,” *IEEE Transactions on Networking*, vol. 4, no. 3, pp. 375-385, 1996.

[TUC09] W. Tucer, D. Wilczak, “A Rigorous Lower Bound for the Stability Regions of the Quadratic Map,” *Physica D.*, vol. 238, no. 18, pp. 1923-1936, 2009.

4

Design of New TRNGs

As explained in Section 1.3.2, there are two alternatives to ASIC TRNGs that have not been studied as much. The first alternative, consists of using the noise generated by a sensor (temperature, accelerometer, pressure, …) to generate random numbers. The other alternative consists of designing TRNGs for PLDs, using the thermal noise, jitter or metastability as random sources. In this chapter, these two alternatives have been studied. In Section 4.1, a proposal and analysis of a TRNG based on MEMs accelerometer is presented. In Section 4.2, three different TRNGs based on Digital Nonlinear Oscillators (DNOs) have been implemented, analyzed and compared. The first one, was based on a ring oscillator, the second one was an architecture proposed by Golić in [GOL06] and the third one is a new proposed architecture that have been presented in [ADD18 and ADD19]. Finally, in Section 4.3, a comparison among the two different approaches (using sensor-based or DNOs-based TRNGs) is presented, stating the strong and weak points of each option.

4.1. TRNG based on a MEMS accelerometer

Although the idea of using a sensor as a seed generator is not novel [VOR11, HON15, WAL16, REV17], as far as we know, there are not commercially available sensor-based TRNGs. Among all the possible sensors, in this work we have studied the usage of a MEMS accelerometer as a source of noise. The main reason is that these sensors are cheap and are ubiquitously present in many wireless devices (such as smartphones, wearable devices, laptops, drones, etc.) so, in many cases, it can be possible to reuse a component that is already available in the device.

4.1.1. Noise signal analysis

To generate random numbers, the noise signal produced by means of the evaluation board EVAL-ADXL335Z has been used. This board contains a small low-energy three-axis accelerometer, ADXL335 [ANA09]. This sensor is composed by a polysilicon surface-micromachined capacitive sensor and a conditioning electronic stage that implements an open-loop measurement architecture. Acceleration deflects the moving mass and unbalances the differential capacitor resulting in a sensor output whose voltage amplitude is proportional to acceleration. The phase-sensitive demodulation technique is then used to determine the magnitude and direction of the acceleration. Some of its typical applications, such as mobile systems or sports and health devices, are strongly related with Internet of Things (IoT).

The evaluation board has been supplied with batteries to avoid coupling 50 Hz or 60 Hz harmonics from the power supply network. AC signal of the measurements

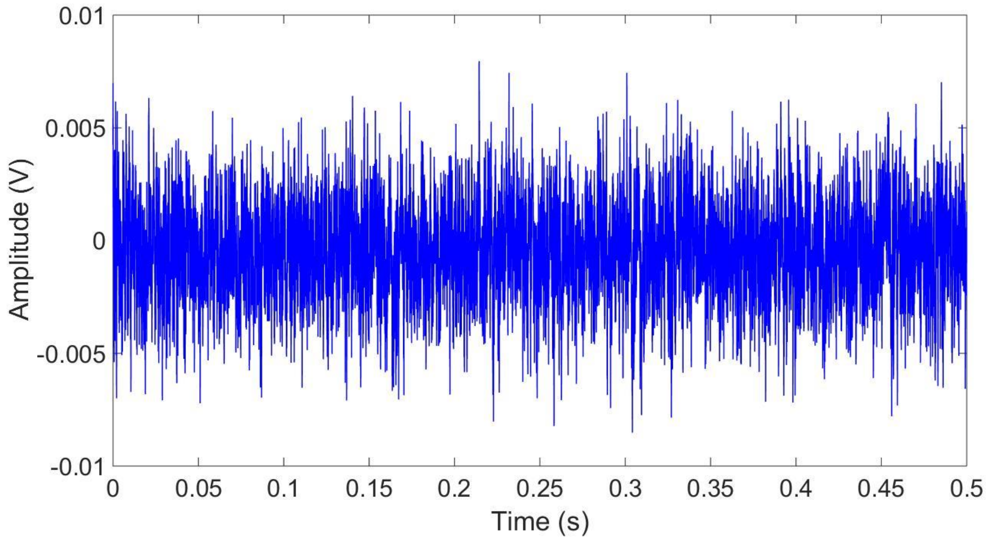


Fig. 4.1. Sample random data signal generated by the accelerometer at rest, measured using a sampling frequency of 25 kHz.

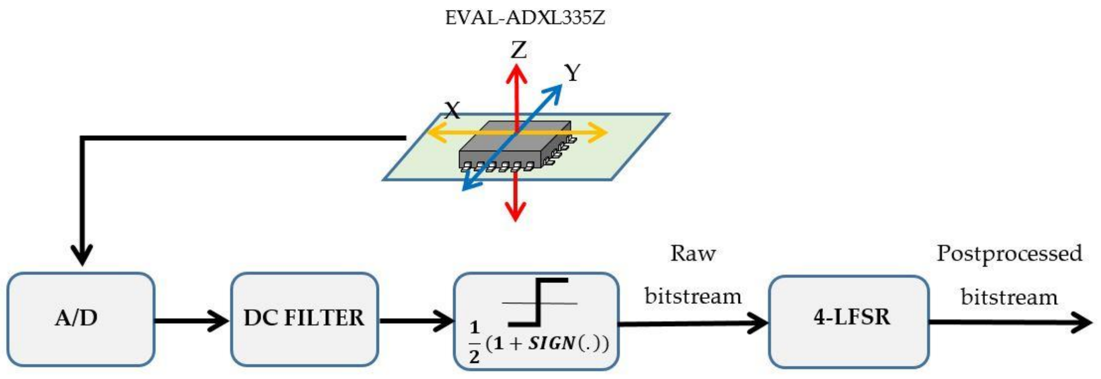


Fig. 4.2. Conceptual block diagram of the processing performed

of the acceleration in the X axis have been acquired by an oscilloscope at several sampling frequencies, 250, 25, 2.5, 0.25 and 0.1 kHz using a dynamic range of $\pm$8 mV and 8-bit resolution. An example of noise signal, obtained using a sampling frequency of 25 kHz is shown in Fig. 4.1. To minimize the correlated common noise, signals generated by the X and Y sensors have been subtracted.

4.1.2. Bitstream generation and post-processing

After sampling the signal noise, the DC level has been eliminated and a sign detection has been applied in order to generate a raw bitstream (Fig. 4.2). Once the raw bitstream for key generation has been obtained, its viability as a source of random numbers has been analyzed. For this purpose, several bitstreams of

1 million bits obtained with different sampling frequencies have been subjected to the statistical tests explained in Section 2.3.1: pattern distribution of bytes, correlation coefficients and Average Shannon Redundancy (Figs. 4.3, 4.4, 4.5).

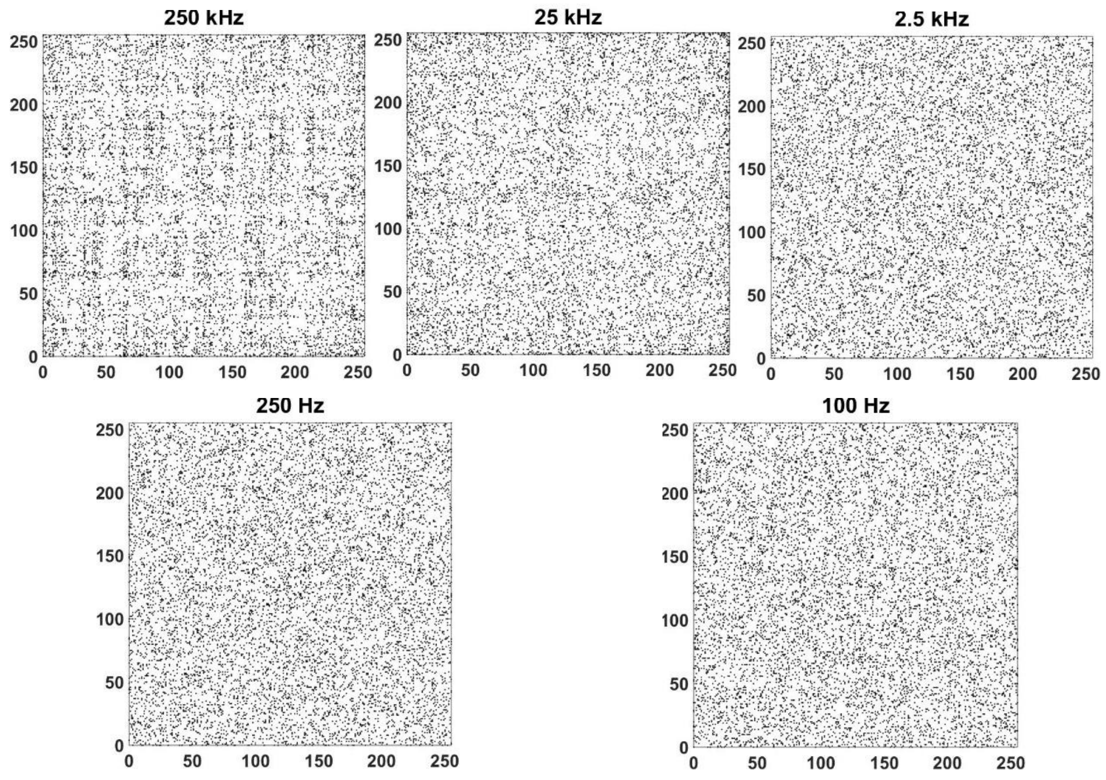


Fig. 4.3. Pattern distribution of bytes at sampling frequencies of 250 kHz, 25 kHz, 2.5 kHz, 250 Hz and 100 Hz. Some areas with a too big or too low concentration of dots can be found in the figures obtained 250 kHz and 25 kHz, indicating that those sequences are not random.

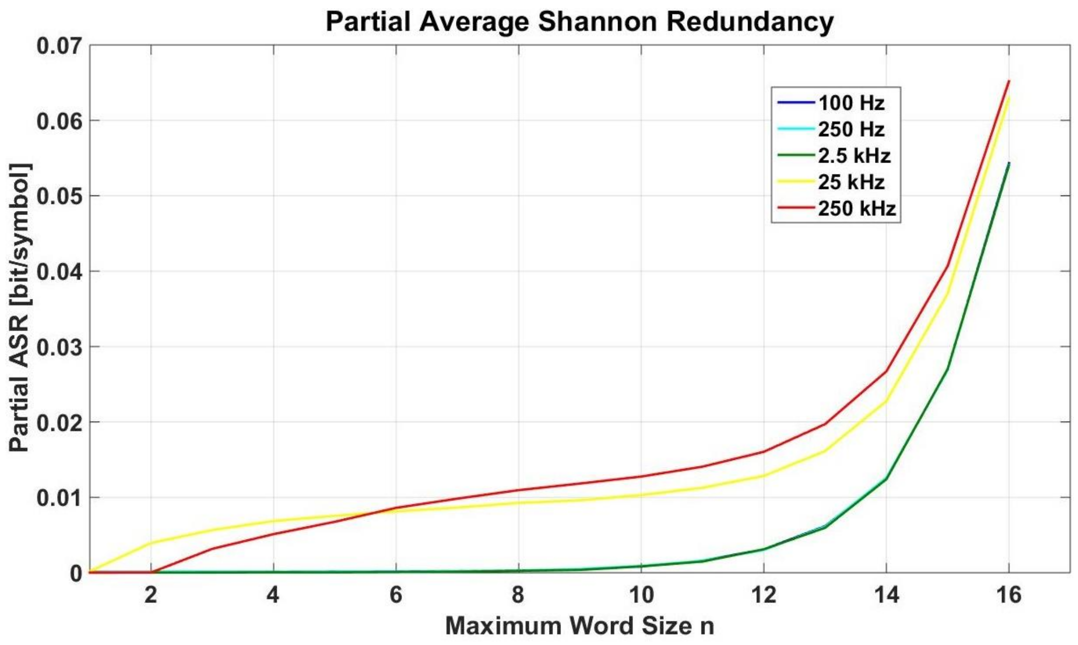


Fig. 4.4. Partial Average Shannon Redundancy at all sampling frequencies. Lines at 2.5 kHz, 250 Hz and 100 Hz are overlapped.

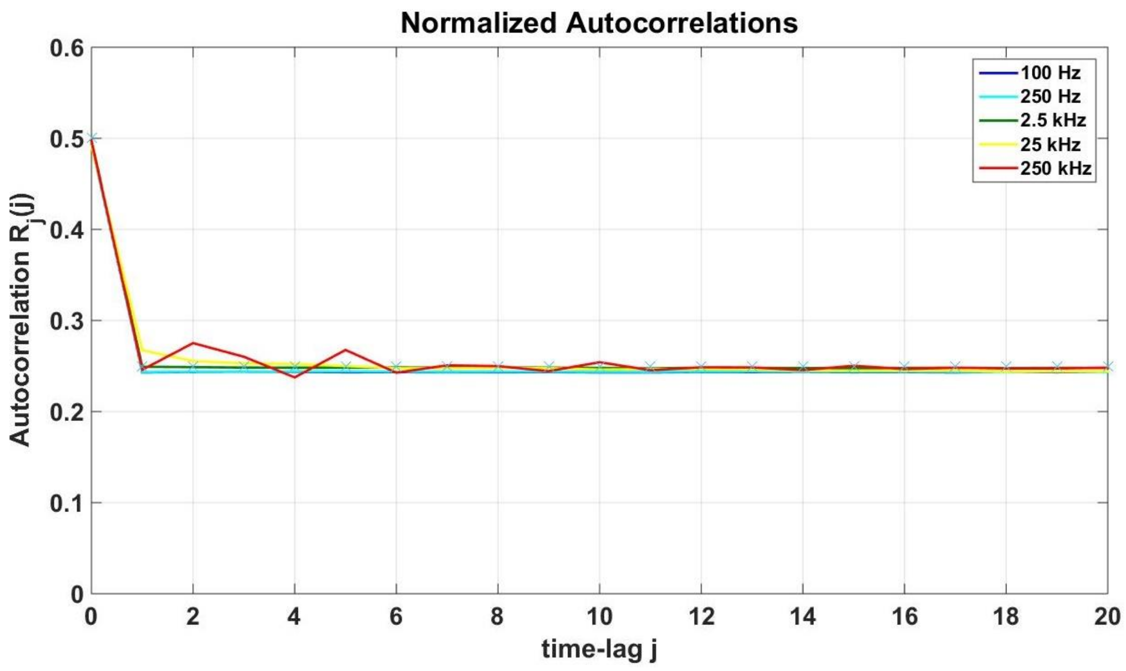


Fig. 4.5. Autocorrelation coefficients at all sampling frequencies. Lines at 2.5 kHz, 250 Hz and 100 Hz are almost the same.

As it can be seen from Fig. 4.5, the first correlation coefficient in all cases is very close to 0.5, indicating that the sequences present almost no bias. On the other hand, from Fig. 4.3 and Fig. 4.4, It can be seen that, although the samples acquired at 250 kHz and 25 kHz are clearly not random, the sequences obtained at 2.5, 0.25 and 0.1 kHz all present very similar results and they look like truly random sequences.

To check if, indeed, these sequences are random, they have been subjected to the NIST randomness test. The results of these tests are shown in Table 4.1. The list of applied tests with their numeration is the same as the one used in Fig. 3.3 in Section 3.2.

As it can be seen, none of these sequences pass all the NIST tests proving the high sensitivity of these tests. On the other hand, it can be seen that the number of tests passed increases by decreasing the sampling frequency. This result is consistent with the theory explained in Section 2.3 since, by decreasing the sampling frequency, the sequences present a lower statistical dependency.

From these results it can be seen that, to pass all the NIST tests, it is necessary to post-process the sequences. Among all the possible post-processing techniques, the Secure Hash Algorithm 512 (SHA-512) [DWO15] algorithm was

Table 4.1. NIST test results for the raw bitstreams (10^6 bits) obtained at different sampling frequencies

Sampling Frequency (kHz)	**250**	**25**	**2.5**	**0.25**	**0.1**
NIST Test					
1	x	x	x	x	x
2	x	x	√	x	√
3	x	x	x	√	√
4	x	x	√	√	√
5	√	√	√	√	√
6	x	x	√	√	√
7	x	x	x	√	x
8	x	x	√	√	√
9	x	x	√	√	√
10	√	x	√	√	√
11	x	x	√	√	√
12	x	x	√	x	√
13	x	x	x	x	x
14	√	√	√	√	√
15	√	x	√	√	√
16	x	x	x	x	x
17	x	x	√	√	√

originally used in the work presented in [GAR17]. The main reason for choosing this algorithm was that, according to NIST recommendations [TUR18], it is a valid post-processing function. Furthermore, since it is a commonly used function, there are multiple available implementations in software libraries or hardware IPs that can be easily accessed. A problem of using this function is that its implementation requires more resources than other simpler post-processing functions. Furthermore, almost any sequence post-processed by SHA-512 will pass the NIST tests so that it is not ideal to compare among different generators. For this reason, for this Thesis we have post-processed those sequences by using a simpler algorithm, consisting on XORing the sequences with a sequence generated by a 4-order LFSR. This post-processing algorithm has been also used in the TRNG proposed in the next section so that both generators can be easily compared. The NIST tests results for the sequences post-processed using this method are shown in Table 4.2. As it can be seen, the sequences obtained at a sampling frequencies of 250 kHz and 25 kHz, fail a few NIST tests. However, for

Table 4.2. NIST test results for the post-processed bitstreams (10^6 bits) obtained at different sampling frequencies

Sampling Frequency (kHz)	**250**	**25**	**2.5**	**0.25**	**0.1**
NIST Test					
1	√	√	√	√	√
2	√	√	√	√	√
3	√	x	√	√	√
4	√	√	√	√	√
5	√	√	√	√	√
6	√	√	√	√	√
7	√	√	√	√	√
8	√	√	√	√	√
9	√	√	√	√	√
10	√	√	√	√	√
11	x	x	√	√	√
12	x	x	√	√	√
13	√	√	√	√	√
14	√	√	√	√	√
15	√	√	√	√	√
16	√	√	√	√	√
17	√	x	√	√	√

all the smaller sampling frequencies, the tests are passed so, with this simple post-processing, it is possible to generate good random sequences at frequencies up to 2.5 kHz.

4.1.3. Mechanical response of the accelerometer

Although all the measurements have been made with the accelerometer at rest, the mechanical response of the accelerometer in an IoT device can often surpass the noise signal. However, this problem could be solved using some of the following methods:

- First, since it is not necessary to generate the keys constantly, the system can generate keys only when the accelerometer is at rest. Currently, some algorithms exist that can detect when the accelerometer is at rest. Although these algorithms are usually used to put the device in sleep mode, they could also be used to activate the key generation process [TUC10].

- Second, signal acquisition in inertial MEMS (such as an accelerometer) is usually based on the synchronous demodulation technique, which requires an output low-pass filter to demodulate the output signal and reduce the high frequency noise. Similarly, another path with a high-pass filter could be included. This way, when the device was in a normal operation mode, the low-pass filter path would be used while the system would switch to the patch with the high-pass filter in order to generate the keys.
- Third, most of the accelerometers include a self-test pin that, when set at a certain voltage, the accelerometer presents a null response. Therefore, this feature could be used to stop the accelerometer from generating the keys. For example, capacitive accelerometers use the force-feedback technique to force the proof mass back to its rest position.
- Finally, a combination of these solutions could be used. Further study is needed in order to determine the best solution for each application.

4.2. TRNG based on Digital Nonlinear Oscillators

The design of new TRNGs for Programmable Logic Devices (PLDs) has been a hot topic in the last years [TSO03, DIT07, SUN07, WOL08, DAN09, GÜL10, HAT12, MAR15, RAI15]. Compared to ASICs the PLDs present a huge advantage in terms of cost and versatility. Unfortunately, PLD-TRNGs still suffer from a lack of trust in the information security communities, mainly due to some major cryptographic weaknesses found in solutions based on ring oscillators [BAU11, AMA13, RAI15, YAN16].

In this Thesis, we have focus on the design of TRNGs for PLDs, in particular, TRNGs based on Digital Nonlinear Oscillators. To introduce this topic, we provide first an informal definition of Digital Nonlinear Oscillators.

Definition 2.3. A Digital Nonlinear Oscillator (DNO) is a network of electronic circuits, each one originally designed to behave as a digital logic gate, implementing an autonomous nonlinear dynamical system exhibiting oscillations in the continuous domain.

The term nonlinear in this case refers to the intrinsic nonlinear nature of electronic circuits implementing digital logic gates, using transistors as switches, at a “large signal” operation scale, providing the saturation of their output voltages toward either the power supply or ground levels. The behavior of a DNO may have little to share with the logic operation on its original gates since it is an analog system, even if implemented in a digital device.

In presence of electronic noise, DNOs may be affected by phase noise and jitter, which can be used as a source of randomness [ACO17]. Some examples of this kind of systems are the ring oscillators [MAN10], the circuits presented by Golić in [GOL06, DIC07] and a novel class of systems proposed in this Thesis [ADD18, ADD19]. In this work, these three approaches have been implemented, analyzed and compared.

4.2.1. Ring oscillator

A ring oscillator is composed by an odd number of cascaded inverters connected in a close loop chain (Fig. 4.6). Ideally, its output oscillates between two levels (high and low) with a period that depends on the number of inverters as well as the propagation delay in each stage. If the number of inverters is n and the delay of each inverter is τ_d, this period is given by: $T = 2\,\tau_d n$. However, in presence of electronic noise, this period variates in a random manner (jitter). This jitter accumulates over time and, therefore, by using a large sampling period, $\Delta T > T$, the measured value will be unpredictable so this system can be used as a TRNG. Several techniques to improve the performance of these TRNGs such as combining several ring oscillators have been proposed in the literature [FIS08, SUN06].

It must be pointed out that, typically, the jitter can depend on the temperature and, when implemented in a PLD, it can also depend in the routing so it can be difficult to assess the security of these TRNGs.

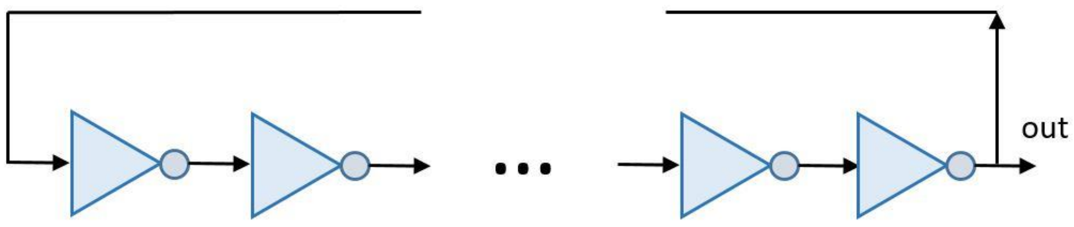


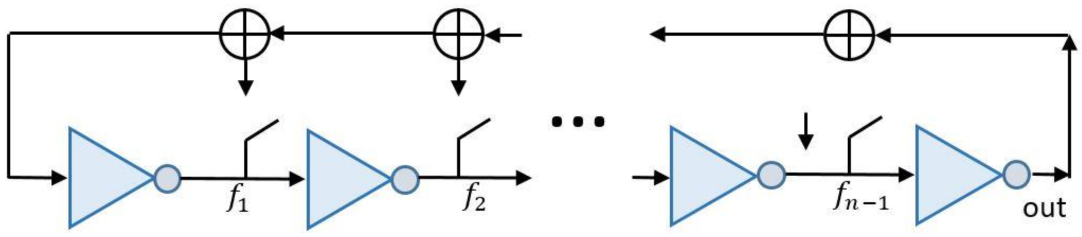
Fig. 4.6. Scheme of a ring oscillator. The number of inverters must be odd.

4.2.2. Golić system

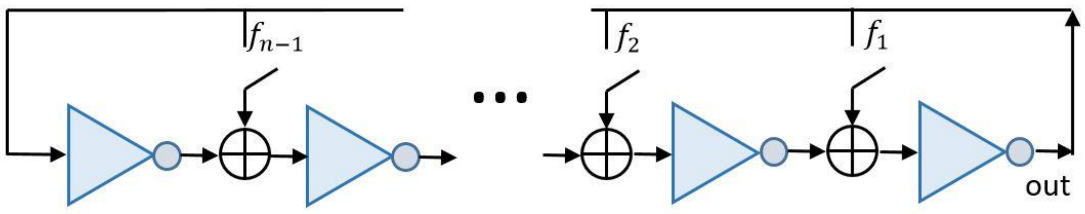
In 2006, J. D. Golić proposed a novel class of TRNGs based on the structure of ring oscillators but, instead of using a single circular feedback, they use a more complex feedback incorporating XOR gates in an analogous way to the Fibonacci and Galois configurations of an LFSR (Fig. 4.7). The idea behind this proposal is to combine the pseudo-randomness properties of the LFSRs with the true randomness properties of ring oscillators due to oscillation jitter. For both Fibonacci and Galois configurations, the feedback connections are specified by the binary coefficients f_i and, therefore, the configuration can be defined using a binary polynomial $f(x) = \sum_{i=0}^{n} f_i x^i$, $f_0 = f_1 = 1$ called the feedback polynomial.

Golić analyzed these structures as synchronous finite state machines, identifying theoretical conditions to avoid fixed points and to achieve maximum period systems. This way, only certain feedback polynomials were allowed. However, although the analysis of logical fixed points could be useful, the discrete algebra analysis cannot say much about the dynamics of the implemented circuit since it is far from being a finite state machine.

In this work, we have focused on the TRNGs based on a Galois configuration. Its structure is similar to the structure of a ring oscillator (as shown in Fig. 4.6) but it includes some extra feedback connections. The inverters as well as the feedback coefficients are indexed from right to left. The input to the first inverter is always directly defined by the feedback function. If $f_i = 0$, the input of the ith inverter is directly defined by the output of the $(i+1)$th inverter and, if $f_i = 1$, the input of the ith inverter is formed by XORing the output of the $(i+1)$th inverter with the feedback signal. This structure is shown in Fig. 4.7b were the possible feedbacks are determined by switches that are closed if $f_i = 1$ and open if $f_i = 0$ (in practice, in this case, the XOR gate is not present). One advantage of using Galois configurations over Fibonacci configurations is that, given a primitive polynomial of order n, it is possible to implement it in an FPGA using exactly n LUTs. If coefficient $f_i = 1$, the implemented function in the ith LUT is a NOT operation and, if $f_i = 0$, the implemented function in the ith LUT is an XNOR operation. According to Xilinx specifications [XIL16], the LUT propagation delay does not depend on the function implemented so the total time-delay will only depend on the order of the primitive polynomial.



(a)



(b)

Fig. 4.7. Scheme of a (a) Fibonacci ring oscillator and (b) Galois ring oscillator.

4.2.3. A novel class of DNOs

Starting with the hypothesis that the fundamental principle of Golić's system is that, by adding some feedbacks to an oscillator, it can present much complex dynamics, we have proposed to implement a system similar to Golić's Galois Ring Oscillator, but using delays instead of inverters. As long as the system is oscillating, similar complex dynamics can be expected. To avoid having any stable points, the system is fed with a 3-inverters ring oscillator. The scheme of this structures is shown in Fig. 4.8. Analogously to the previous case, the feedback connections can be specified by a feedback polynomial and the number of LUTs (and thus the total delay) only depends on the order of this polynomial (when $f_i = 1$ the function implemented in the ith LUT is an XOR and, when $f_i = 0$ the implemented function in the ith LUT is an identity operation $(x \to x)$.

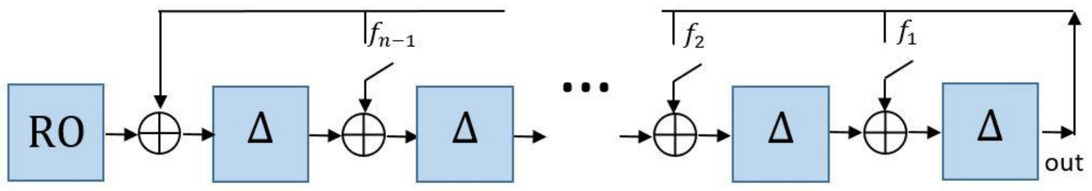
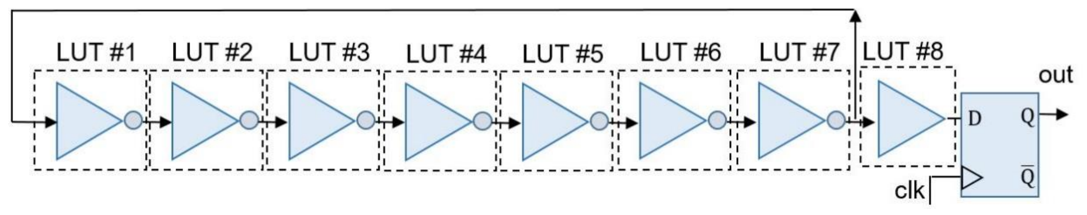


Fig. 4.8. Scheme of the proposed novel class of DNOs. Δ symbol represents a delay operation (the implemented function is $x \rightarrow x$).

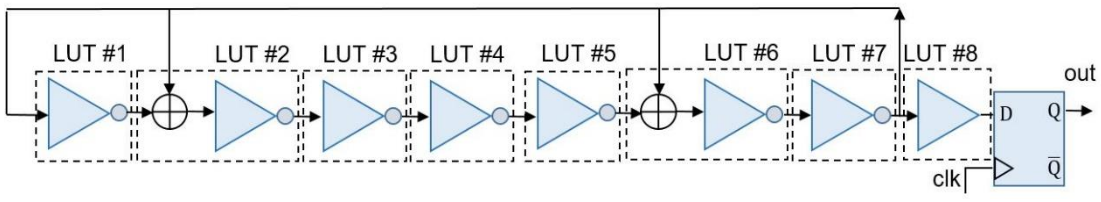
4.2.4. Comparison among the three different structures

To compare among these structures, we have implemented 3 different TRNGs, all of them using the same number of LUTs implemented in the exact same location within the FPGA but each of them using a different structure. To implement these kind of systems, special directives from Xilinx to avoid combinational loop errors or the deletion of some design blocks have been used. Furthermore, to have a fair comparison, we have put additional constraints in the RTL code so that the ith LUT of each structure is implemented in the exact same position within the FPGA.

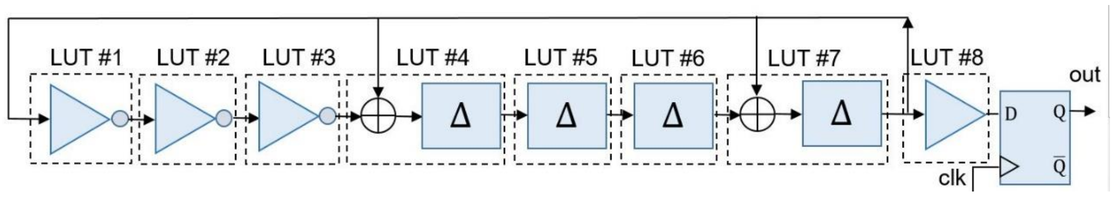
The implemented structures are shown in Fig. 4.9. To measure the random bits, an extra inverter (for avoiding possible period couplings) followed by a D-Flip Flop has been used. The flip-flop is fed with a clock signal with different frequencies, which allow as to study the behavior of the systems at different sampling frequencies. The collected data is sent to a computer and stored in binary files using a Labview project. The scheme of the whole measuring system is shown in Fig. 4.10. As seen in the figure, a controller block chooses a sampling frequency to collect the random bits. The collected bits are stored in a shift register and, when a full byte is stored, it is sent to the controller along a “ready” signal. The controller then selects a RAM address and writes the collected byte in a RAM block. This process is repeated until 125000 bytes are collected. After that, the controller sends a signal to the transmitter block which starts to send all the bytes to a computer, where they are saved in a binary file using a Labview project. When all the bytes have been transmitted, the controller sends a signal to the frequency selector to select a different sampling frequency.



(a)



(b)



(c)

Fig. 4.9. Implemented TRNGs. (a) Ring oscillator, (b) Golić's structure, (c) Proposed structure.

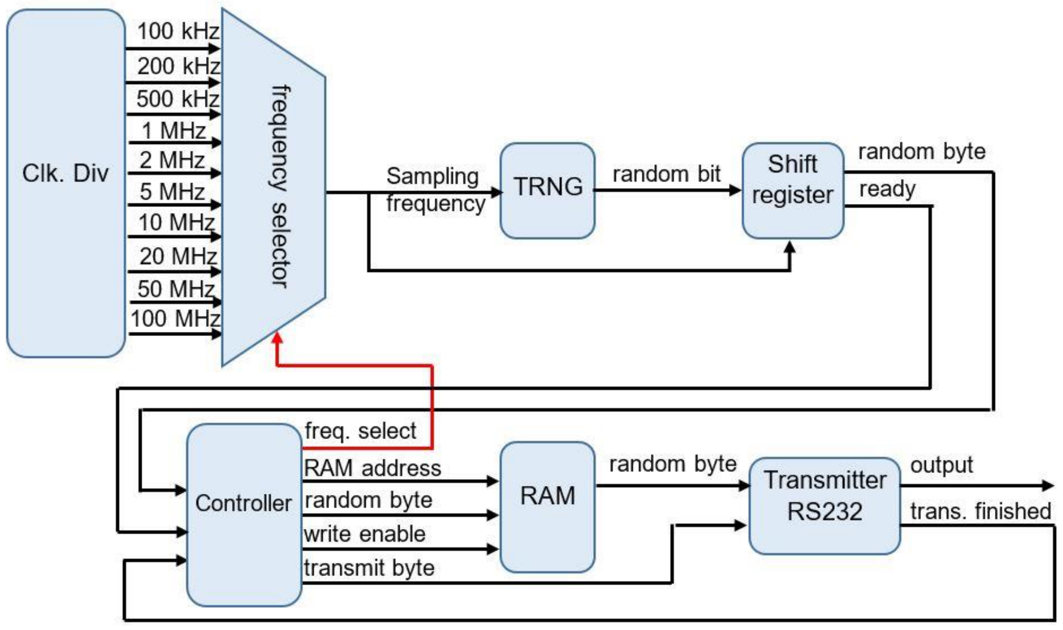


Fig. 4.10. Scheme of the measuring system.

The random sequences generated by these systems at different sampling frequencies have been subjected to several statistical tests. The most important tests, are summarized in Fig. 4.11, Fig. 4.12 and Fig. 4.13. As it can be seen, the proposed DNO outperforms the DNO proposed by Golić approximately in one order of magnitude and the ring oscillator in 2 orders of magnitude in the sampling frequency.

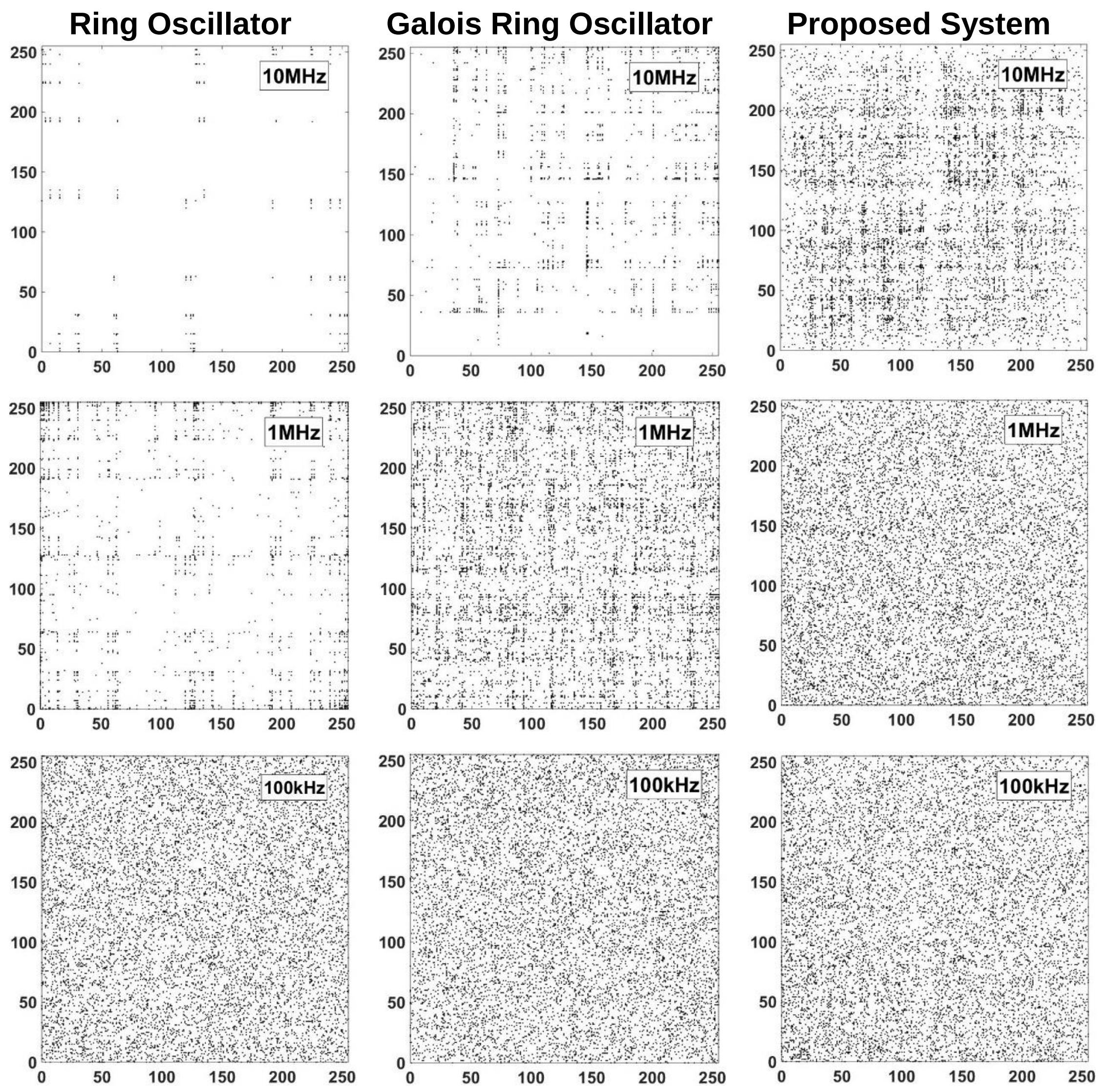


Fig. 4.11. Pattern distributions of successive generated bytes on the plane for the three DNOs considered in this work, for sampling frequencies of 10 MHz, 1MHz and 100 kHz.

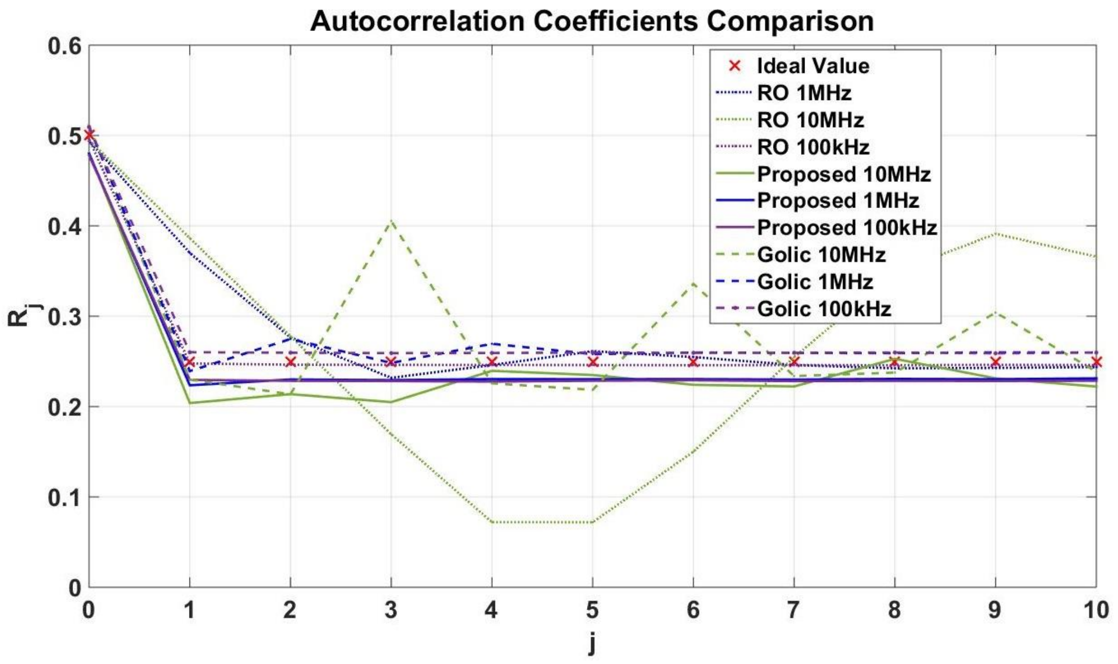


Fig. 4.12. Autocorrelation coefficients for the three DNOs considered in this work, for sampling frequencies of 10 MHz, 1MHz and 100 kHz.

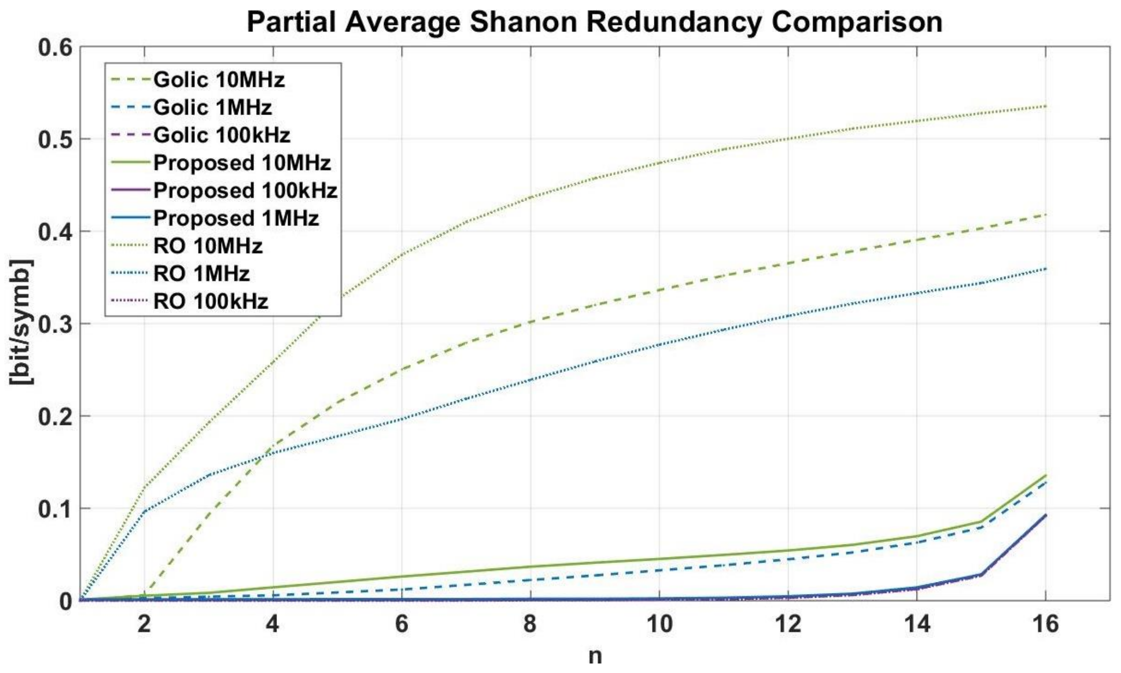


Fig. 4.13. Partial Average Shannon Redundancy for the three DNOs considered in this work, for sampling frequencies of 10 MHz, 1MHz and 100 kHz.

This figures, however, can be misleading since previous works such as [DIC15] have shown that the random behavior of the Golić's systems can strongly depend on the chosen polynomial as well as the location where the system is implemented. A similar result has been observed in our experiments for all

TRNGs, especially Golić's TRNGs as well as the novel proposed TRNGs. Therefore, by choosing different polynomials or locations, Golić's system could outperform the proposed system. To take this in consideration, a statistical analysis implementing all three DNO's structures in several locations and in different FPGA's is being made. The preliminary results have been published in [ADD19] and they show that, on average, the proposed system is quite better than Golić's system and much better than the ring oscillator.

4.2.5. Post processing

Finally, by using the same post-processing as proposed in Section 4.1.2 (XORing the sequences with a sequence generated by a 4-order LFSR), we have managed to pass the NIST tests (using a sequence of 10^6 bits) at a sampling frequency of 2 MHz. Therefore, the proposed TRNG based on a DNO is clearly capable of outperforming the proposed TRNG based on a MEMS-accelerometer in terms of speed.

An alternative implementation XORing two identical 7-nodes DNOs has also been testing. By using the same post-processing, has been able to pass the severe NIST tests (100 sequences of 10^6 bits) at a sampling frequency of 5 MHz. Thus, by using just 21 LUTs (8 for each TRNG 1 for the XOR and 4 for the LFSR), the system is capable of generating true random sequences at a very high speed.

4.3. Analysis and comparison of the proposed systems

As we have seen, both systems (MEMS-based and DNO-based TRNGs) have proven to be good sources of entropy although, in order to pass the NIST tests, they need a post-processing stage. Since, in each case, different sampling frequencies have been used, it is not possible to make a straight up comparison. However, it can be clearly seen that the TRNG based on the proposed DNO can generate higher entropies at higher sampling frequencies. For example, in Fig. 4.14, it can be seen that, in terms of randomness, the sequence obtained with the MEMS-based TRNG at 250 kHz looks "more random" than the sequence obtained by DNO-TRNG at 10 MHz but "less random" than the sequence obtained at 5 MHz. On the other hand, by post-processing the sequences using the 4-order LFSR to pass the NIST test, in the case of the MEMS-based TRNG a sampling frequency of 2.5 kHz or lower is needed while in the DNO-based

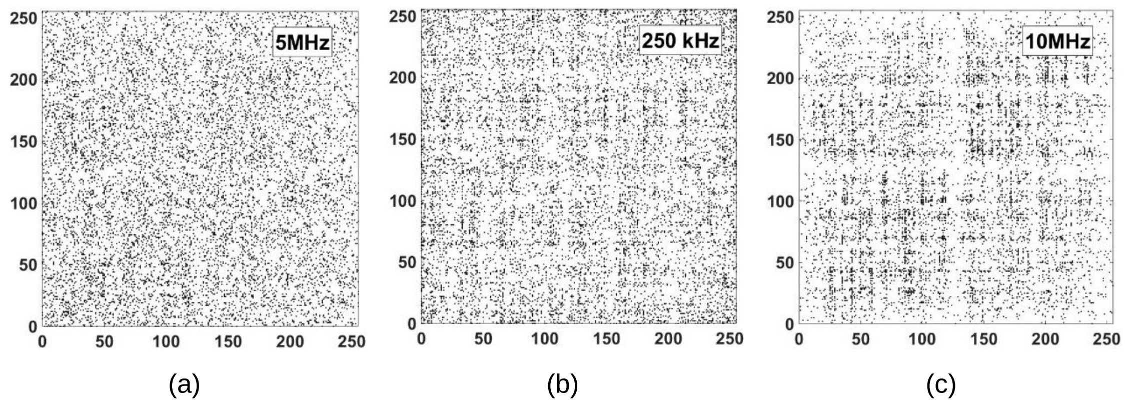


Fig. 4.14. Pattern distributions of successive generated bytes on the plane for (a) the proposed DNO-based TRNG at 5 MHz (b) the MEMS-based TRNG at 250 kHz and (c) the proposed DNO-based TRNG at a 10 MHz.

TRNG sampling frequencies up to 2 MHz can be used. Therefore, we can conclude that the DNO-based TRNG is much more efficient. However, it must be pointed out that the speed of the TRNG is not often a key factor in several applications such as keys generation. Furthermore, as explained before, using a MEMS accelerometer can have an added value of reusing a sensor that is present on the device. Therefore, depending on the application it could be a more suitable option.

4.4. Conclusions

In this chapter, two different approaches to create new TRNGs have been studied. The first approach consisted of using the noise generated by a sensor as a source of entropy while the second one consisted of using the jitter that is present in DNOs. Both approaches have been tested experimentally by comparing the output random sequences. To compare the quality of all the collected sequences, the statistical tests explained in Section 2.3.1 have been used.

In the first case, a TRNG that samples the signal generated by a MEMS accelerometer at rest have been tested. For this purpose, sequences of 1 million bits have been collected at sampling frequencies of 250 kHz, 25 kHz, 2.5 kHz, 250 Hz and 100 Hz. The statistical tests have shown that, for these last three sampling frequencies, the graphs obtained (pattern distribution of bytes, correlation coefficients and ASR) were very similar to the graphs generated by an ideal truly random sequence. Although these sequences were not able to pass all the NIST randomness tests, by applying a simple post-processing technique consisting of XORing the sequences with a sequence generated by a 4-order LFSR, all the NIST tests were passed (for sampling frequencies of 2.5 kHz, 250 Hz and 100 Hz).

In the second case, three different DNO structures have been implemented in an FPGA, each of them using the same number of logic blocks in the exact same location within the FPGA. The first structure was based on a ring oscillator, the second structure was based on chain of inverters with XOR feedback loops (Golić's system) while the first structure was a novel proposed structure consisting of a chain of delays with XOR feedback loops. In each case, 1 million sequences have been obtained by sampling the systems at different frequencies. By comparing the implemented systems, we have concluded that the sequences generated by the proposed system offered better results in terms of randomness. Among the implemented systems, we have seen that, for the same sampling frequencies, the sequences generated by the proposed structure present the best statistical properties. By using the same post-processing technique as the one used in the previous approach, the proposed system has been capable of passing the NIST test using a sampling frequency of 2 MHz.

In summary, with both approaches, we have been capable of generating true random sequences capable of passing the NIST test using a simple post-processing. The DNO-based TRNGs (specially the new proposed structure) have generated good random sequences at higher sampling frequencies than the MEMS-based TRNG although the MEMS-based TRNG present some advantages such as the possibility of reusing a sensor that is present on the device.

4.5. References

[ACO17] A. Acosta, T. Addabbo, E. Tena-Sanchez, “Embedded Electronic Circuits for Cryptography, Hardware Security and True Random Number Generation: an Overview,” *International Journal of Circuit Theory and Applications*, vol. 45, no. 2, pp. 145-169, 2017.

[ADD18] T. Addabbo, A. Fort, M. Mugnaini, V. Vignoli, M. Garcia-Bosque “Digital Nonlinear Oscillators in PLDs: Pitfalls and Open Perspectives for a Novel Class of True Random Number Generators,” *Proceedings of the 2019 International Symposium on Circuits and Systems (ISCAS 2018)*, pp. 1-5, 2018.

[ADD19] T. Addabbo, A. Fort, R. Moretti, M. Mugnaini, V. Vignoli, M. Garcia-Bosque “Lightweight True Random Bit Generators in PLDs: Figures of Merit and Performance Comparison,” *Proceedings of the 2019 International Symposium on Circuits and Systems (ISCAS 2019)*, pp. 1-5, 2019.

[AMA13] T. Amaki, M. Hashimoto, Y. Mitsuyama, T. Onoye, “A Worst-Case-Aware Design Methodology for Noise-Tolerant Oscillator-Based True Random Number Generator with Stochastic Behavior Modeling,” *IEEE Transactions on Information Forensics and Security,* vol. 8, no. 8, pp. 1331-1342, 2013.

[ANA09] Analog Devices, “ADXL35,” Available online: https://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf (accessed 19 May 2019), 2009.

[BAU11] M. Baudet, D. Lubicz, J. Micolod, A. Tassiaux, “On the Security of Oscillator-Based Random Number Generators,” *Journal of Cryptology*, vol. 24, no. 2, pp. 398-425, 2011.

[DAN09] J. L. Danger, S. Guilley, P. Hoogvorst, “High Speed True Random Number Generator Based on Open Loop Structures in FPGAs,” *Microelectronics Journal*, vol. 40, no. 11, pp. 1650-1656, 2009.

[DIC07] M. Dichtl, J. D. Golić, “High-Speed True Random Number Generation with Logic Gates Only,” *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 07)*, pp. 45-62, 2007.

[DIC15] M. Dichtl, “Fibonacci Ring Oscillators as True Random Number Generators – A Security Risk,” *IACR Cryptology ePrint Archive*, pp. 270, 2015.

[DWO15] M. Dworkin, “SHA-3 Standard: Permutation-Based Hash and Extendable Output Functions,” *NIST FIPS-202*, 2018.

[GAR17] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, S. Celma, “Application of a MEMS-Based TRNG in a Chaotic Stream Cipher,” *MDPI Sensors*, vol. 17, no. 3, pp. 1-15, 2017.

[GOL06] J. D. Golić, “New Methods for Digital Generation and Postprocessing of Random Data,” *IEEE Transactions on Computers*, vol. 55, no. 10, pp. 1217-1229, 2006.

[GÜL10] U. Güller, S. Ergün, G. Dündar, “A Digital IC Random Number Generator with Logic Gates Only,” *2010 17th IEEE International Conference on Electronics Circuits and Systems*, pp. 239-242, 2010.

[HAT12] H. Hata, S. Ichikawa, "FPGA Implementation of Metastability-Based True Random Number Generators," *IEICE Transactions on Information and Systems*, vol. E95.D, no. 2, pp. 426-436, 2012.

[HON15] S. L. Hong, C. Liu, "Sensor-Based Random Number Generator Seeding," *IEEE Access*, vol. 3, pp. 562-568, 2015.

[MAN10] M. K. Mandal, B. C. Sarkar, “Ring Oscillators: Characteristics and Applications,” *Indian Journal of Pure & Applied Physics*, vol. 48, pp. 136-145, 2010.

[MAR15] H. Martn, T. Korak, E. S. Milln and M. Hutter, “Fault Attacks on STRNGs: Impact of Glitches, Temperature and Underpowering on Randomness,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 266-277, 2015.

[RAI15] M. Raitza, M. Vogt, C. Hochberger, T. Pionteck, “Raw 2014: Random Number Generators on FPGAs,” *ACM Transactions on Reconfigurable Technolology and Systems (TRETS),* vol. 9, no. 2, pp. 15:1-15:21, 2015.

[REV17] G. Revadigar, C. Javali, W. Xu, A. V. Vasilakos, W. Hu, S. Jha, “Accelerometer and Fuzzy Vault-Based Secure Group Key Generation and Sharing Protocol for Smart Wearables,” *IEEE Transactions on Information Forensics and Security,* vol. 12, no. 10, pp. 2467-2482, 2017.

[SUN07] B. Sunar, W. J. Martin, D. R. Stinson, “A Provably Secure True Random Number Generator with Built-in Tolerance to Active Attacks,” *IEEE Transactions on Copmuters,* vol. 56, no. 1, pp. 109-119, 2007.

[TSO03] K. H. Tsoi, K. H. Leung, P. H. W. Leong, “Compact FPGA-Based True and Pseudo Random Number Generators,” *Proceedings of the 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2003),* pp. 51-61, 2003.

[TUC10] K. Tuck, “Low Power Modes and Auto-Wake/Sleep Using the MMA8450Q,” Freescale Semiconductor Inc. Available online: https://www.nxp.com/assets/documents/data/en/application-notes/AN3921.pdf (accessed 19 May 2019), 2010.

[TUR18] M. S. Turan, E. Barker, J. Kelsey, K. A. Mcay, M. L. Basish, M. Boile, “Recommendation for the Entropy Sources Used for Random Bit Generation,” *NIST Special Publication 800-90B*, 2018.

[VOR11] J. Voris, D. Saxena, T. Halevi, “Accelerometers and Randomness: Perfect together,” *Proceedings of the 4th ACM Conference on Wireless Network Security,* pp. 115-126, 2011.

[WAL16] K. Wallace, K. Moran, E. Novak, G. Zhou, K. Sun, “Toward Sensor-Based Random Number Generation for Mobile and IoT Devices,” *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1189-1201, 2016.

[WOL08] K. Wold, C. H. Tan, “Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillators Rings,” *2008 International Conference on Reconfigurable Computing and FPGAs*, pp. 385-390, 2008.

[XIL16] Xilinx, “7 Series FPGAs Configurable Logic Block User Guide,” Available online: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf (accesed 29 May 2019), 2016.

[YAN16] J. Yang, Y. Ma, T. Chen, J. Lin, J. Jing, “Extracting More Entropy for TRNGs Based on Coherent Sampling,” *International Conference on Security and Privacy in Communication Systems*, pp. 694-709, 2016.

5

Applications for Ethernet Communications

In this chapter, two new encryption methods for the physical layer (PHY) for 1 Gb and 10 Gb Optical Gigabit Ethernet communications have been proposed and implemented in an FPGA. We have named them PHYsec and 10G-PHYsec respectively. In Section 5.1, a general introduction to the physical layer will be presented. Then, in Section 5.2, the PHYsec encryption method will be presented while, in Section 5.3, the 10G-PHYsec will be presented. In both cases, the overall structure of the physical layer used in each standard (1000Base-X for 1 Gb Ethernet and 10GBase-R for 10 Gb Ethernet) will be introduced along with the proposed modifications to perform the encryption. Then, the encryption algorithms compatible with each standard will be presented in each case. Finally, in both cases the implementation results as well as the security analysis will be presented.

5.1. Introduction to the physical layer

The physical layer (PHY) is responsible for carrying out the lower level functions in the transmission. It defines the electrical, mechanical and functional specifications to activate, maintain and deactivate the physical link. In the case of Ethernet standards, the physical layer is usually divided into three sublayers with different functionalities: Physical Coding Sublayer (PCS), Physical Medium Attachment (PMA), and Physical Medium Dependent (PMD). Among them, the PCS layer performs functions such as autonegotiation, link establishment, clock rate adaptation, symbol synchronization and data encoding.

Usually, in high speed baseband serial data standards such as Ethernet, the clock signal is not sent in a separate line. Instead, the receiver must use Clock and Data Recovery (CDR) circuits to extract the clock signal from the data bitstream itself, using its bit transitions.

To facilitate the work of the CDR circuits, the transmitted bitstreams must be encoded in a way such that DC-balance, high transition density and short run length are achieved. Block line encoders group input bits into n-bit blocks and map them into n'-bit blocks, where $n' > n$. Thanks to this redundancy introduced in the data, the encoders can achieve these properties [SMI04].

In this Thesis, we have proposed and implemented two encryption systems (one for 1 Gb and one for 10 Gb) that perform the encryption at the physical layer. An obvious challenge of these encryption systems is that, to guarantee the previously mentioned properties (DC-balance, high transition density and short run length), they must preserve the coding properties. On the other hand, encryption in the physical layer can present some advantages against other layer encryption standards that operate at upper layers such as MACsec [LAZ17] or IPsec [SAU11b]:

- Both IPsec and MACsec encryption standards need to introduce additional overhead to secure each packet, which results in an increase of the network latency and a decrease in the effective throughput [XEN06]. As an example, according to [TRO05], the overhead introduced during IPsec encryption reduces the effective throughput between 20% and 90% of the maximum achievable, depending on the packet size. For example, for a 64-byte packet size, IPsec can result in up to 40% wasted bandwidth and incremental latencies on the order of 125 ms. For larger packet sizes, the throughput increases but the latency also increases approaching values of the order of 350 ms for 1420-byte packet sizes [MAR13, MIC16]. By performing the encryption at the physical layer, it is possible to avoid introducing overhead, achieving a line rate and minimum latency.
- In the physical layer, extra control bytes or ordered sets such as packet start, packet end or idle of bytes are introduced to facilitate the communication process. However, with this information, an attacker can know if a communication is taking place. By performing the encryption in the physical layer, it is possible to encrypt those bytes, making them undistinguishable from the information bytes so that an attacker cannot obtain any information about the state of the communication. This is not possible to achieve by performing the encryption in upper layers since they only work with information bytes.

Finally, it must be noticed that, by performing the encryption preserving the coding properties and maintaining the rest of physical layer features untouched, the physical layer encryption maintains compatibility with the subsequent hardware elements or medium-dependent circuitry. For example, commercial optical modules such as Small Form-Factor Pluggable (SFPs) and electronic circuits such as CDR or Serializer/Deserializer (SERDES) for 1000Base-X standard are also compatible with the proposed encryption methods.

5.2. 1 Gb Optical Ethernet encryption

In 1 Gb Optical Ethernet, a 1000Base-X standard, which implements an 8b/10b encoding in the physical layer, is used. In this coding, each input symbol consists of 9 bits: 8 of them determine the value of the symbol while the other one (flag) indicates weather it is a data symbol or a control symbol. This symbol is then mapped to a 10-bit symbol to achieve the properties of DC-balance, high transition density and short run length. As explained in Section 5.1, these properties are necessary to facilitate the operation of the remote Clock and Data Recovery (CDR) circuit. The receiver then transforms each 10-bit symbol into the original 9-bit symbol. In this coding, the data set is limited: while the data symbols can take any of the 256 possible values, the control symbols can only take 12 possible values. One of these control symbols is not used for standard data communication [IEE18] so it will not be considered in this Thesis. Therefore, out of the 512 existing 9-bit words, only 267 constitute a valid symbol, which adds some difficulty in the encryption process.

Indeed, if the encryption is performed before the encoding, the encrypted chain of symbols should only contain these 267 possible symbols to perform the coding with no errors. Therefore, to encrypt a symbol, it is not possible to directly XOR it with a random 9-bit keystream since it could generate a non-valid symbol. On the other hand, if the encryption process is performed after the encoding, XOR-ing the 10-bit symbols with random 10-bit keystreams would not guarantee that the resulting chain of bits meets the intended requirements of the 8b/10b coding (DC-balance, high transition density and short run length). Therefore, it would also be necessary to assure that each encrypted value is a valid 10-bit symbol. However, even in this case, an additional issue would arise: in the 8b/10b encoding, for each input 9-bit symbol, there are two possible 10-bit output symbols, one is the negated of the other one. The standard choses one of them to achieve a perfect balance of the transmitted chain. If the encryption was performed after the encoding, by randomly mapping each 10-bit symbol to any valid 10-bit symbol, this perfect balance would not be guaranteed. The only possibility would be to design an encryption algorithm that mapped each symbol to another symbol that shared the same parity. Due to these issues, in this Thesis, we have decided to perform the encryption before the encoding.

5.2.1. Overall structure of the encryption system

As explained in Section 5.1, the physical layer is usually divided into three other sublayers with different functionalities: Physical Coding Sublayer (PCS), Physical Medium Attachment (PMA), and Physical Medium Dependent (PMD). The proposed encryption and decryption will also be performed in the PCS sublayer.

The basic structure of the PCS sublayer, including the encryption/decryption blocks, and its interface with the Medium Access Control (MAC) layer (layer 2) and the PMA sublayer are shown in Fig. 5.1. The coding function is separated in Encoder and Decoder blocks while the rest of the functions of the PCS layer reside in the RX_PCS_CTRL and TX_PCS_CTRL modules.

Unlike other encryption mechanisms such as IPsec or MACsec, where the data packets are encrypted individually, in this work, the 8b/10b symbol flow is directly and continuously encrypted. This encryption method has been named PHYsec.

The encryption/decryption infrastructure in Fig. 5.1 is shown with more detail in Fig. 5.2. The MANAGEMENT module implements the initial synchronization and collects the alarms relative to the synchronization status. This way, in case of a mismatch between the encryption status (e.g., one PCS is encrypting but the other one is not decrypting) or a bad synchronization status (misaligned keystream generators), several alarms are triggered by the MANAGEMENT module and reported to the used thanks to the FPGA debug system. After this

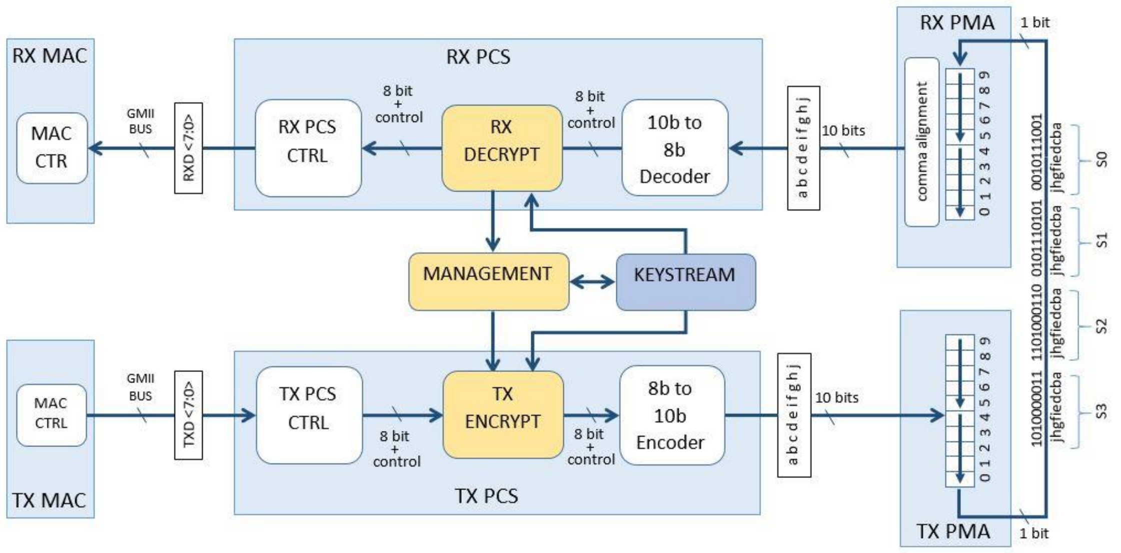


Fig. 5.1. PCS structure with the proposed encryption function PHYsec included.

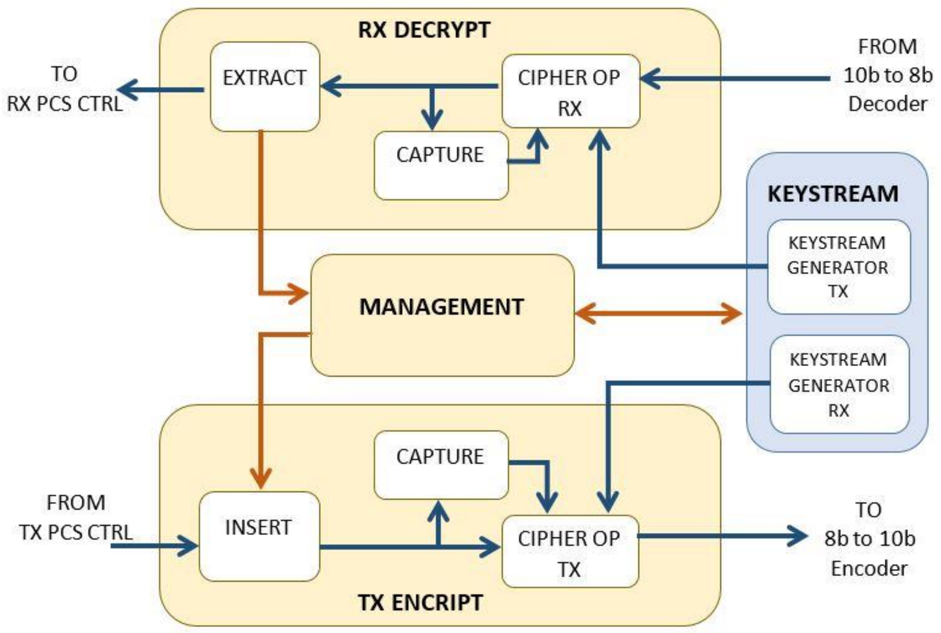
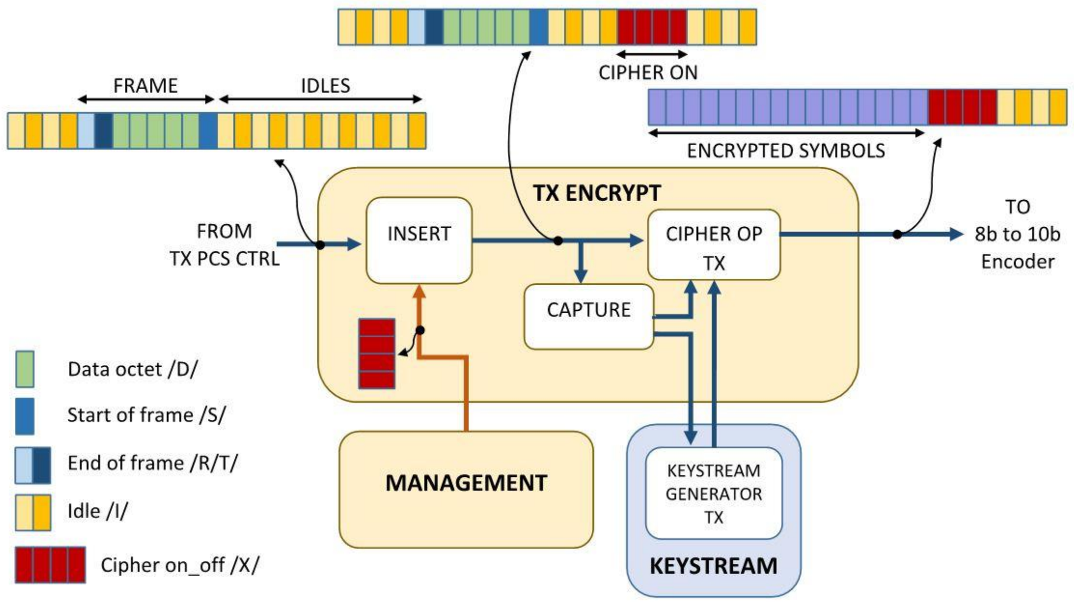


Fig. 5.2. Encryption infrastructure for PHYsec function.

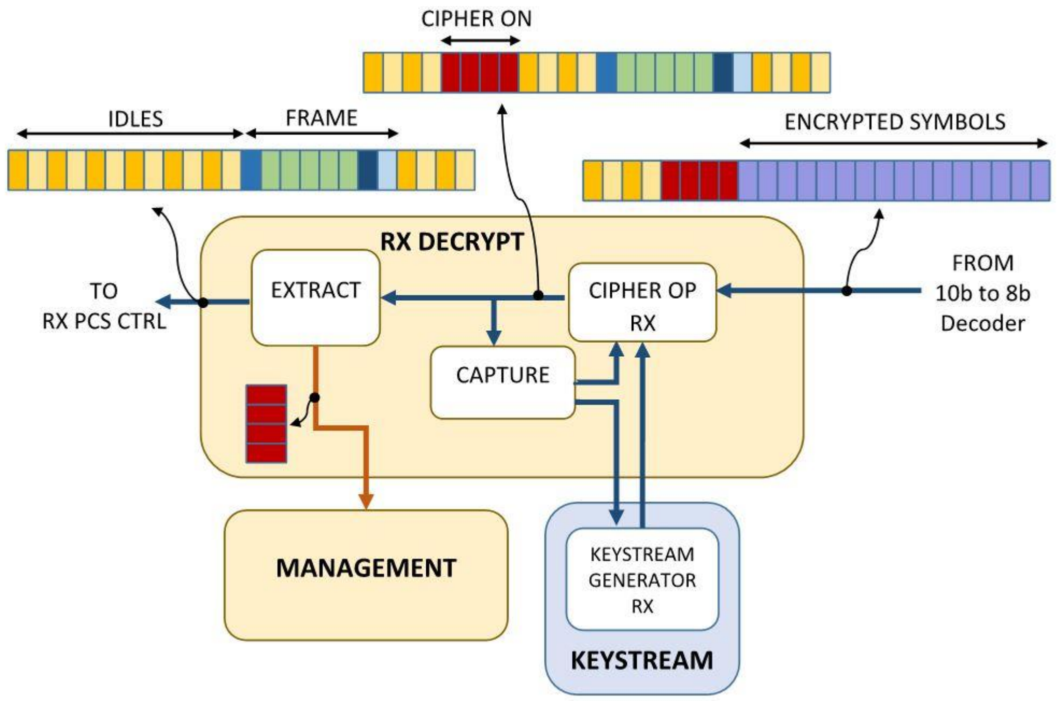
notification, the user can stop the keystream generators and restart the encryption synchronization procedure.

To achieve encryption synchronization with a stream cipher it is necessary that the keystream sequence generated at the receiver (RX) side is exactly aligned with the incoming keystream sequence embedded into the ciphertext and generated at the transmitter (TX) side. For that purpose, MANAGEMENT module in Fig. 5.2 performs the insertion a control sequence into the 8b/10b symbol stream to indicate when to start or stop the encryption and the decryption processes.

The basic operation of this procedure is shown in Fig. 5.3. In the transmitter (Fig. 5.3a), the CIPHER_OP_TX module, responsible for performing the encryption operation, is initially disabled, allowing the 8b/10b symbols to be transparently passed from the TX_PCS_CTRL to the 8b/10b encoder. In order to start encrypting the 8b/10b symbol stream, the MANAGEMENT module acts on the INSERT block to introduce a /X/ encryption start message in the 8b/10b symbol flow. This message is formed by a set of four special symbols and are introduced in the symbol stream by replacing four idle symbols (in 1000Base-X standard, when no communication is happening, idle symbols are continuously being transmitted). Before passing through the CIPHER_OP_TX module, /X/ message is detected by the CAPTURE module. Then, the CAPTURE module enables the CIPHER_OP_TX module and the keystream generator module (KEYSTREAM_GENERATOR TX), which initiates the encryption process after /X/ has been transmitted. As shown in Fig. 5.3a, at the output of the CIPHER_OP_TX module, every 8b/10b symbol after /X/ is encrypted.



(a)



(b)

Fig. 5.3. Initial synchronization procedure, based on the proposed /X/ ordered set, (a) transmission, (b) reception.

At the receiver (Fig. 5.3b), the module CIPHER_OP_RX is in charge of performing the decryption operation. Just like the transmitter, it is initially inactive, enabling 8b/10b symbols to be transparently passed from the 8b/10b decoder to the controller RX_PCS_CTRL. When the CAPTURE module detects the /X/ message, the decipher module (CIPHER_OP_RX) and the keystream generator (KEYSTREAM_GENERATOR RX) are enabled, starting to decrypt the 8b/10b stream after the /X/ set. Subsequently, the control message /X/ is extracted from the data stream in the EXTRACT module, which replaces it with idle ordered sets in the 8b/10b symbol flow (reverse operation of the INSERT module).

To disable the encryption and the decryption, the MANAGEMENT module introduces again a control message /X/ in the symbol flow, replacing four 8b/10b idle symbols. In the transmitter, the CAPTURE module detects the /X/ message and disables the encryption. In the receiver, when the CAPTURE module detects the /X/ message, the decryption is disabled and the extract block replaces the /X/ message with idle symbols.

5.2.2. Encryption algorithm

To encrypt the symbols while preserving the coding properties, each 9-bit symbol is first mapped to an integer value in the range of 0 to 266. After the mapping, the stream cipher operation is performed. This operation consists of a modulo-267 addition between the mapped symbols and the keystream, which also takes values uniformly distributed between 0 and 266. Once the cipher operation is done, the resulting values are reverse-mapped to the corresponding new 9-bit symbols. For both the mapping and the de-mapping, a (public) bijective (one-to-one) map needs to have been defined first. In the decryption process, the same keystream is subtracted from the encrypted mapped symbol and the modulo-267 operation is performed (for negative numbers, if $n \in [-266, -1]$, $n \pmod{267} = 267 + n$). Then, this integer value is reverse-mapped to the corresponding 9-bit symbol. This approach for preserving the coding properties has been proposed and presented in [PER18a, PER18b, PER19a, PER19b]. The structure of the ciphering/deciphering modulo (CIPHER_OP_TX/ CIPHER_OP_RX) is shown in Fig. 5.4.

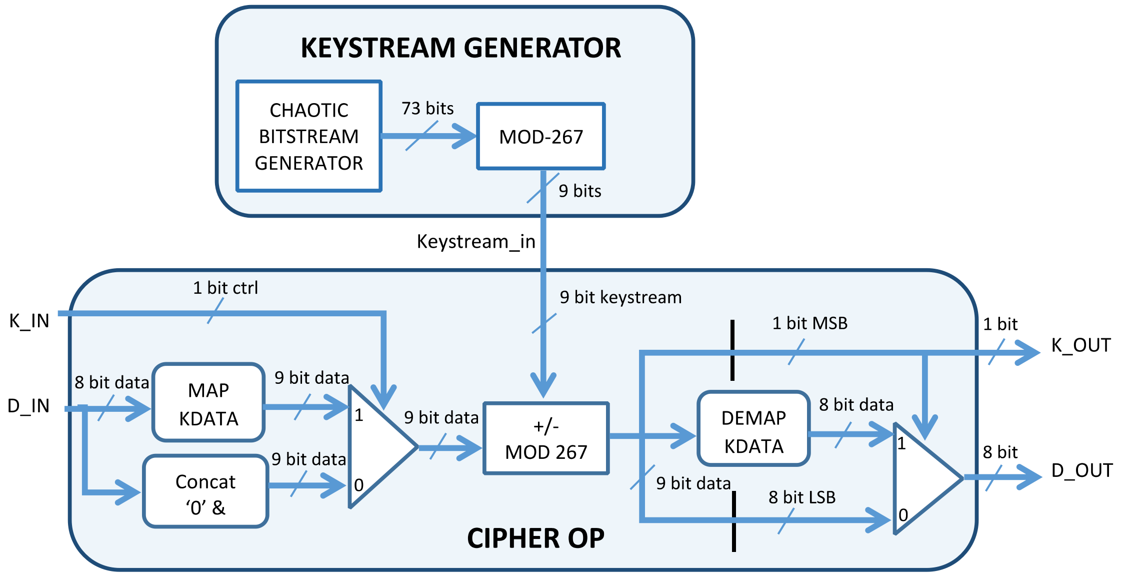


Fig. 5.4. Stream cipher operation performed in CIPHER_OP module next to one of the keystream generators (RX or TX).

To generate a keystream that takes a value between 0 and 266, it is possible to generate a pseudorandom bitstream (with any of the algorithms proposed in Section 3) with a width of 9 or more bits, and perform the modulo-267 operation. Unfortunately, when doing this operation, even if the original words are perfectly random, a bias is introduced in the resulting keystreams. According to the NIST recommendations [BAR15], in order to make this bias negligible, the input width of the modulo-267 operation shall be at least 64 bits longer than the output. Since in our case 9-bit numbers are necessary, the input of the modulo-267 operation generated by the pseudorandom bitstream generator must have a size of at least 73 bits.

The implementation of the modulo-267 operation has been based on [BUT11], which presents a high-speed hardware implementation for a generic operation '$x \; mod \; z$' that can be fragmented into a pipeline of $n - m + 1$ stages, where x is represented by n bits and z by m bits. In our case, z has been taken as a constant value equal to 267. According to this, the resulting hardware structure should have 65 stages as shown in Fig. 5.5.

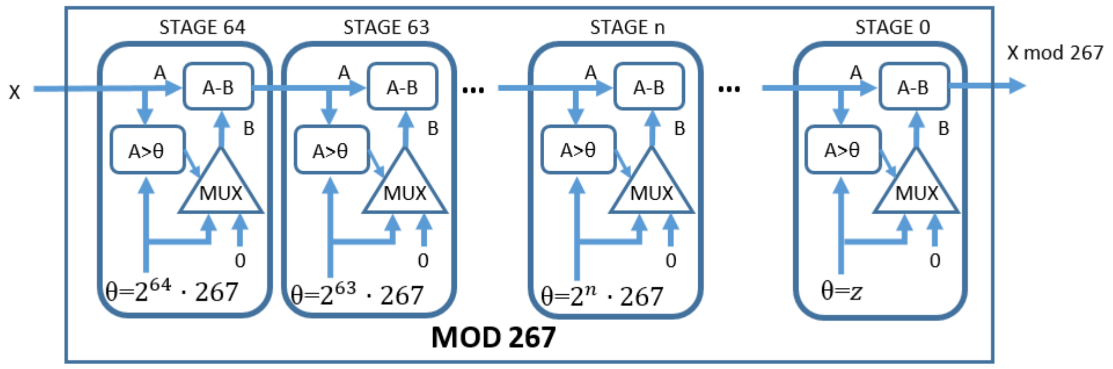


Fig. 5.5. Modulo-267 hardware.

As the pseudorandom bitstream generator, we have chosen the algorithm proposed in Section 3.5 (64 bit STM-LFSR V2), consisting of a 64-bits STM, where each x_i is perturbed by combining the 8 LSBs with the last 8 bits of a 61 order LFSR and those 8 bits are used to form the output bitstream (Fig. 3.10b).The reason for this choice is that, as it has been previously proven, this algorithm is capable of generating bitstreams at a high speed (more than 1 Gbps) while maintaining a high level of security. Since, in this application, a total of 73 random bits must be generated for each modulo-267 operation, a bank of bitstream generators has been built. The bank consists of eight 8-bit width generators (each of these identical as the one shown in Fig. 3.10b) and one 9-bit width generator. Their outputs are concatenated to give a 73-bit output. The final structure for the keystream generator is shown in Fig. 5.6.

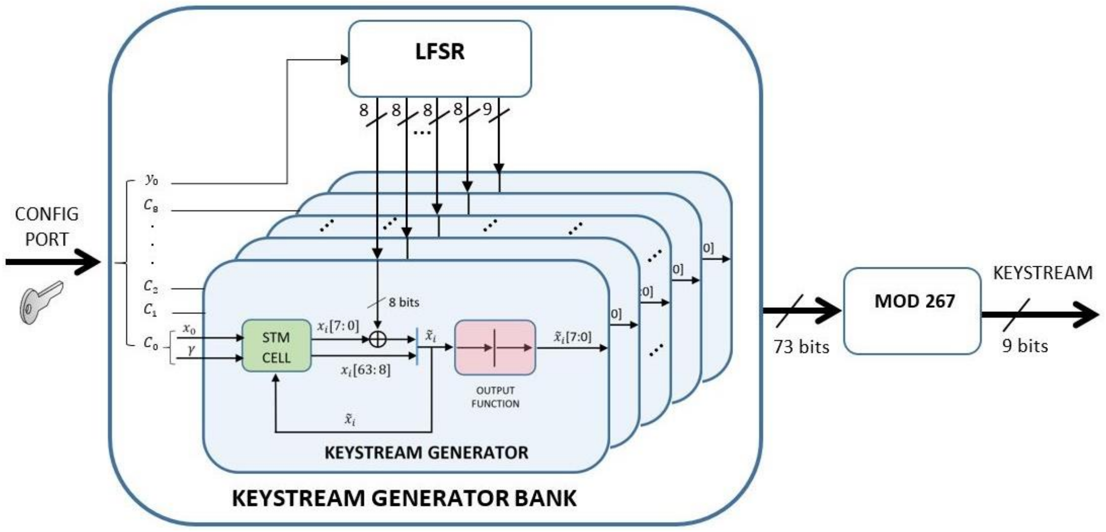


Fig. 5.6. Keystream generator.

5.2.3. Implementation results

The complete system has been implemented in a Xilinx Virtex 7 FPGA [PER18a, PER18b, PER19a]. Regarding the resources used by this encryption system, the main contribution for each block inside each keystream generator (KEYSTREAM GEN) is shown in Table 5.1. As it can be seen, modulo-267 (MOD-267) operation and keystream generator bank (STM_BANK) take up the largest amount of FPGA resources.

In the setup for test, the FPGA has been connected to two Small Form-Factor Pluggable (SFP) modules capable of transmitting at a rate up to 10.3125 Gbps at 850 nm over multimode fiber. In this case, these modules have been configured to work at a rate of 1.25 Gbps which is the necessary rate to achieve a net data transmission of 1 Gbps. The FPGA design consists of two Ethernet interfaces with PHYsec function and two Ethernet frame generator modules connected to them (Fig. 5.7). Each Ethernet interface contains the MAC module and the PHY (PCS and PMA blocks) including the encryption system explained in this work. In Fig. 5.8, a photo of the test setup is also shown.

Table 5.1. FPGA resources used in the keystream generator submodules

MODULE	Slice LUTs	Slice Registers	DSPs
KEYSTREAM_GEN*	10943	3629	144
LFSR	210	195	0
STM_BANK	6606	589	144
MOD-267	4127	2845	0

*KEYSTREAM_GEN refers to the hardware resources of the Keystream Generator, including LFSR, STM_BANK and MOD-267 operation.

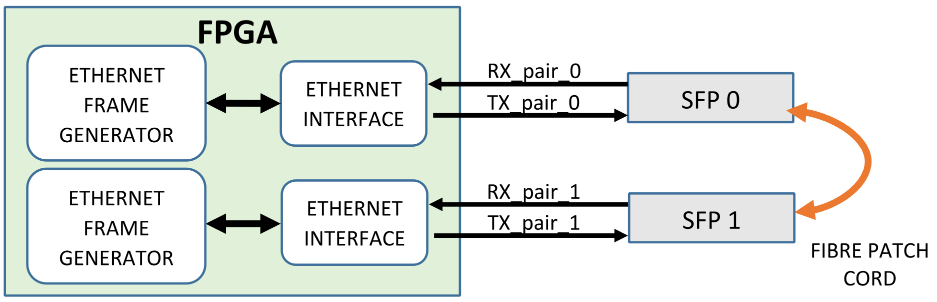


Fig. 5.7. Test setup scheme.

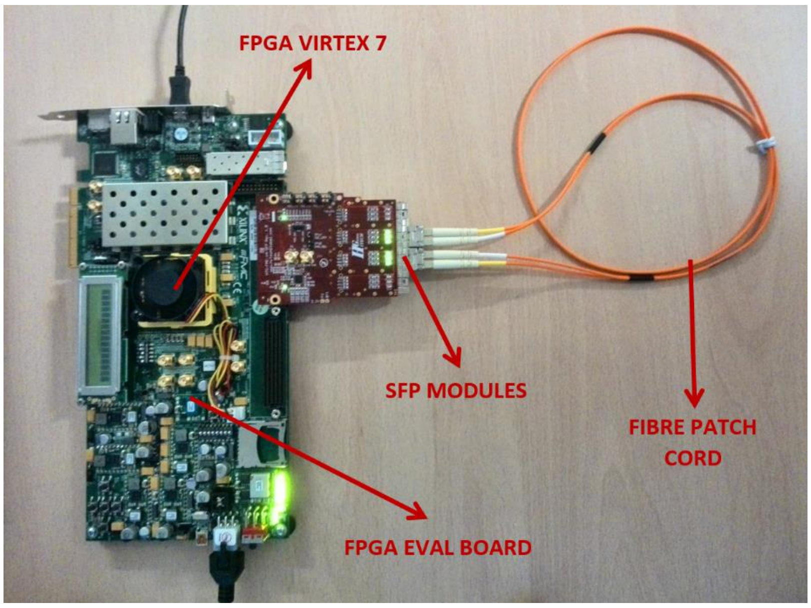


Fig. 5.8. Test setup photo.

The PHY is connected directly to the SFP modules thanks to the FPGA Serializer-Deserializer (SERDES) circuit. In the MAC side, the Ethernet Interface is connected to the Ethernet frame generator to test the encrypted link with real traffic and verify that no frames are lost and no Cyclic Redundancy Check (CRC) errors are produced during encryption.

With this test setup, we have checked that the encryption and decryption work correctly and synchronously without harming data traffic or link establishment between Ethernet interfaces. Thanks to the frame and CRC counters inside the Ethernet frame generators, we have checked that the maximum data rate is achieved without frame losses or CRC errors. In particular, several traffic bursts of 10^6 frames, each of them with a 1500-byte length have been tested. The throughput of the traffic has been configured between 10% and 98%[1] of the maximum line rate.

It is important to remark that the initial PCS structure of this work (without the encryption mechanism) parts from an implementation compatible with the standard. The PHYsec functionality has been developed and incorporated to this initial PCS sublayer.

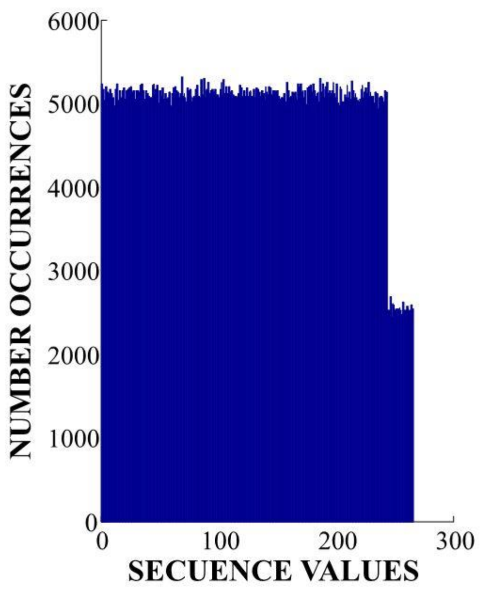
[1] A 100% of the line rate is not possible since, between packages there must be some idle symbols.

Moreover, the final PCS structure, including PHYsec functionality, introduces an extra latency in the 8b/10b TX datapath of 192 ns, with respect to the baseline implementation, and approximately the same in the RX datapath. This extra latency is due to the new hardware modules added in the PCS sublayer as shown in Fig. 5.2. For example, in the TX direction, 8b/10b symbols have to go through the INSERT and CIPHER_OP_TX modules before being encoded. In the INSERT module, a latency of 18 clock cycles is introduced while, in the CIPHER_OP_TX module, the operations (mapping, modulo-267 addition and de-mapping) and delays take 6 clock cycles. Therefore, the total latency introduced in the TX datapath is 24 cycles. Since the frequency of operation of the system is 125 MHz (8 ns period) the total latency in the TX datapath is 192 ns.

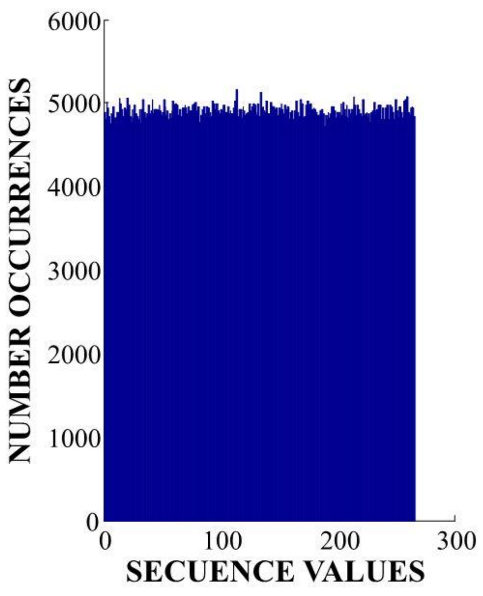
5.2.4. Security analysis

As explained before, by performing the encryption at the physical layer, we intend to achieve two objectives: encrypt the information data (like in any other encryption system) and hide the traffic pattern, so that an attacker cannot know if a communication is taking place.

Regarding the data encryption, although the encryption algorithm based on a STM and an LFSR was proved to be secure in Section 3.5, since the modulo-operation has been added, new statistical tests must be made to check that the sequences are still secure. In order to see that the generated keystreams are uniformly distributed between 0 and 266, some histograms for different sequences have been obtained. In Fig. 5.9, the histograms for two sequences of 1,310,720 values are shown. In one of them (Fig. 5.9a), only a 9-bit width input has been used in the module operation while, in the other one (Fig. 5.9b), a 73-bit width input has been used in the module operation following the NIST recommendation [BAR15]. As it can be seen, a strong bias can be clearly noticed in the first case while, in the second case, which is the one used in this work, this bias seems negligible. To check that, indeed, this keystream is uniformly distributed between 0 and 266, a Chi-Square Goodness Fit [BAL13] test has been applied. This test has not rejected this hypothesis with a 5% significance level. However, it must be noticed that this kind of test does not guarantee the randomness for the final keystream, so other randomness tests must be applied.



(a)



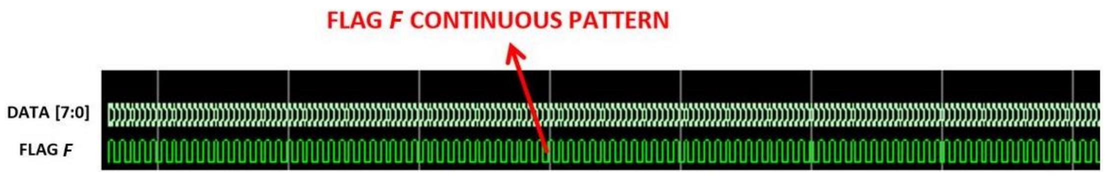
(b)

Fig. 5.9. Histogram of the output keystream (without applying the NIST recommendation and (b) applying the NIST recommendation.

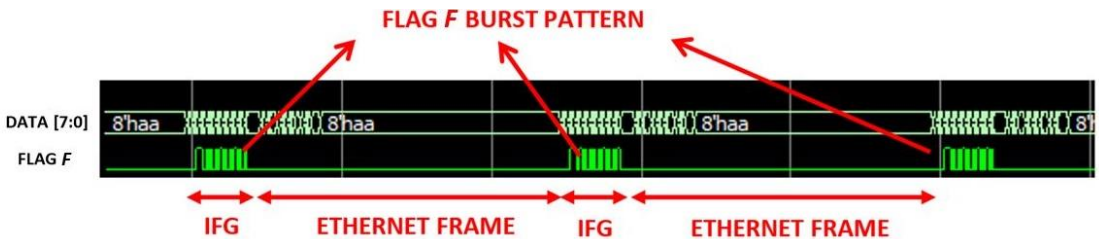
Since the NIST battery of tests [RUK10] is not applicable to non-binary data that are not a power of two, some other randomness tests suitable for non-binary sources described in [KNU81] have been used: frequency test, serial test and poker test. For the frequency and serial test, the Chi-Square Goodness of fit test was successfully passed to sequences of 3 and 15 millions of tuples respectively. Regarding the poker test, also the Chi-Square Goodness of fit test was carried out by using the five categories described by Knuth in [KNU81].

Finally, by doing a similar analysis to the one presented in Section 3.5.2, it is easy to conclude that this algorithm also meets the other security requirements such as: key space size, sensitivity on the key and robustness against reconstruction attacks.

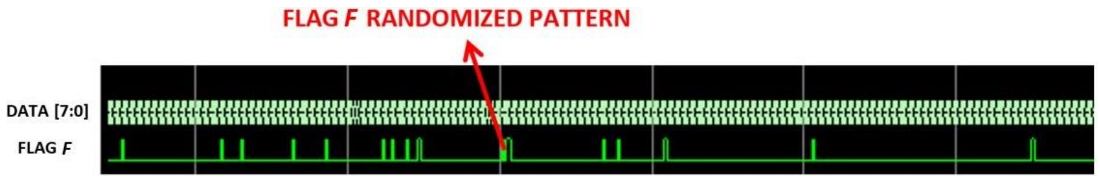
Regarding the second objective, as explained before, each symbol has a flag bit (F) that determines if the symbol is a control symbol or a data symbol. Therefore, the control flag F can give information about the transmission state. This can be seen in Fig. 5.10. When no frames are being transmitted (Fig. 5.10a), idle ordered sets composed by a control symbol and a data symbol are continuously being transmitted. In this case, the F pattern is a signal that is continuously switching between ‘0’ and ‘1’. On the other hand, when transmitting Ethernet frames



(a)



(b)



(c)

Fig. 5.10. (a) F flag pattern before encryption when transmitting an Ethenet frame burst, (b) F pattern before encryption when no Ethernet frame is transmitted and (c) F pattern after encryption regardless of the transmission or nontransmission of Ethernet frames.

(Fig. 5.10b), F follows a burst pattern, as idle transmission only occurs in the Interframe Gap (IFG) periods and only data symbols are transmitted between frame boundaries (setting F to zero). Therefore, if no encryption is used or if the encryption is performed at higher layers (e. g., MACsec or IPsec) it is possible to know if there is Ethernet traffic being transmitted. However with the proposed encryption system at the physical layer, F flag pattern seems completely random in both situations, with or without Ethernet traffic being transmitted (Fig. 5.10c). Therefore, the data traffic pattern is completely hidden.

5.3. 10 Gb Optical Ethernet encryption

In 10 Gb Optical Ethernet communications, the 10GBase-R standard is commonly used. In this coding, a 64-bit input block is converted into a 66-bit block composed by a 2-bit synchronization header and a 64-bit payload. The synchronization header can only take two possible values: ‘10’ in case of a control block and ‘01’ in case of a data block. The purpose of this header is to distinguish between data and control blocks but also to assure that there is at least a bit transition every 66 bits to guarantee the 66-bit block alignment and to limit the

Table 5.2. 64b/66b block formats in 10GBase-R standard

Input Data	Sync	Block Payload								
Bit Position: / Data Block Format:	0 1	2								65
D_0 D_1 D_2 D_3/D_4 D_5 D_6 D_7	01	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
Control Block Formats:		Block Type								
C_0 C_1 C_2 C_3/C_4 C_5 C_6 C_7	10	0x1e	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_0 C_1 C_2 C_3/O_4 D_5 D_6 D_7	10	0x2d	C_0	C_1	C_2	C_3	O_4	D_5	D_6	D_7
C_0 C_1 C_2 C_3/S_4 D_5 D_6 D_7	10	0x33	C_0	C_1	C_2	C_3	▥	D_5	D_6	D_7
O_0 D_1 D_2 D_3/S_4 D_5 D_6 D_7	10	0x66	D_1	D_2	D_3	O_0	▥	D_5	D_6	D_7
O_0 D_1 D_2 D_3/O_4 D_5 D_6 D_7	10	0x55	D_1	D_2	D_3	O_0	O_4	D_5	D_6	D_7
S_0 D_1 D_2 D_3/D_4 D_5 D_6 D_7	10	0x78	D_1	D_2	D_3	D_4		D_5	D_6	D_7
O_0 D_1 D_2 D_3/C_4 C_5 C_6 C_7	10	0x4b	D_1	D_2	D_3	O_0	C_4	C_5	C_6	C_7
T_0 C_1 C_2 C_3/C_4 C_5 C_6 C_7	10	0x87	▥	C_1	C_2	C_3	C_4	C_5	C_6	C_7
D_0 T_1 C_2 C_3/C_4 C_5 C_6 C_7	10	0x99	D_0	▥	C_2	C_3	C_4	C_5	C_6	C_7
D_0 D_1 T_2 C_3/C_4 C_5 C_6 C_7	10	0xaa	D_0	D_1	▥	C_3	C_4	C_5	C_6	C_7
D_0 D_1 D_2 T_3/C_4 C_5 C_6 C_7	10	0xb4	D_0	D_1	D_2	▥	C_4	C_5	C_6	C_7
D_0 D_1 D_2 D_3/T_4 C_5 C_6 C_7	10	0xcc	D_0	D_1	D_2	D_3	▥	C_5	C_6	C_7
D_0 D_1 D_2 D_3/D_4 T_5 C_6 C_7	10	0xd2	D_0	D_1	D_2	D_3	D_4	▥	C_6	C_7
D_0 D_1 D_2 D_3/D_4 D_5 T_6 C_7	10	0xe1	D_0	D_1	D_2	D_3	D_4	D_5	▥	C_7
D_0 D_1 D_2 D_3/D_4 D_5 D_6 T_7	10	0xff	D_0	D_1	D_2	D_3	D_4	D_5	D_6	

maximum run length to 66 bits. In case of the control bytes, some of the bits of the payload determine the block type (Table 5.2).

While in the 1000BaseX standard, the coding achieves the properties of DC-balance, high transition density and short run length, the 64b/66b coding used in 10GBase-R standard does not achieve these properties. To statistically achieve them, a scrambler that randomize the 64-bit payload is used after the coding. It must be noticed that the only purpose of this scrambler is to randomize the bitstream so it does not provide any security.

In a similar manner as in the 1 Gb case, other security protocols such as MACsec only encrypt the data blocks so that, by seeing the headers pattern, it is possible to know if a communication is taking place. In the proposed encryption function,

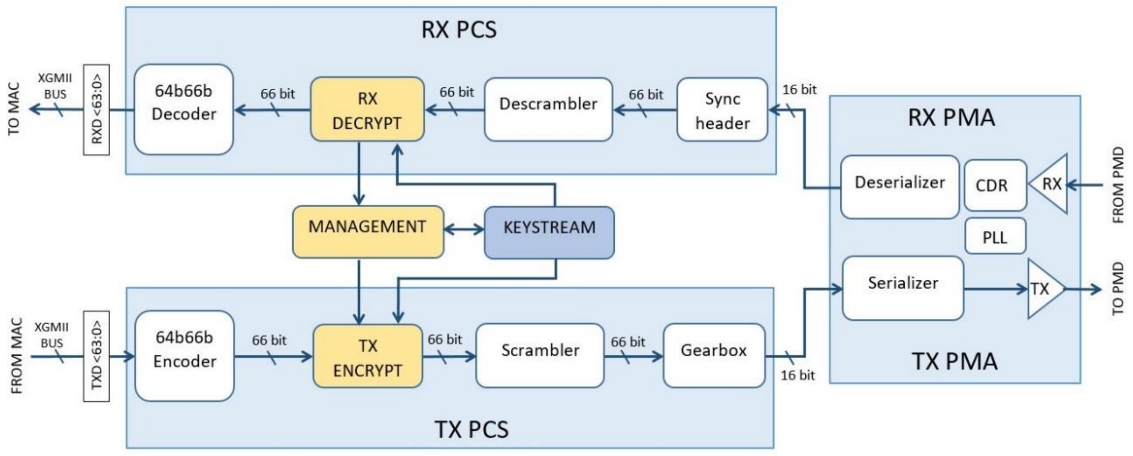


Fig. 5.11. PCS structure in 10GBase-R standard including the proposed encryption function.

10G-PHYsec, by encrypting the headers as well as the payload, it is possible to hide the data traffic pattern [PER18c, PER19d]. Furthermore, like the PHYsec, the 10G-PHYsec achieves minimum latency and zero overhead.

5.3.1. Overall structure of the encryption system

The basic structure of the PCS sublayer in 10GBase-R standard, including the proposed encryption/decryption function (10G-PHYsec) is shown in Fig. 5.11. In the proposed scheme, the encryption has been performed between the encoding and the scrambling while the decryption has been performed between the descrambling and the decoding.

The encryption/decryption infrastructure is shown with more detail in Fig. 5.12. The MANAGEMENT module implements the initial synchronization by introducing two control sequences into the 64b/66b block stream. One of these sequences is used for the encryption activation and the other one for deactivation.

In order to maintain the concordance with the standard, the format of the implemented control sequence is equivalent to that of an ordered set. Among the available 64b/66b block types, the one selected for this work is 0x55. As shown in Table 5.2, inside this block type two consecutive ordered sets, each formed by a control header byte and 3 content data bytes, can be transmitted. The possible values for the content of the ordered sets established in the standard are between values 0x00 and 0x03 and they are used for link fault signaling. Since values above 0x03 are out of use and are reserved for future standardization, in this work, values of 0x04 and 0x05 have been used for activate and deactivate the encryption and encryption blocks (Cipher On and Cipher Off in Table 5.3).

The final encryption sequence for enabling the encryption consists of the 0x55 block type filled with two consecutive ‘Cipher ON’ ordered sets. In this work this block is called ‘Cipher_ON block’. For disabling encryption the sequence is the same but using two ‘Cipher OFF’ ordered sets and the resulting block is called ‘Cipher_OFF block’.

Table 5.3. Link fault signaling ordered sets and new proposed ordered sets.

LANE 0	LANE 1	LANE 2	LANE 3	DESCRIPTION
Sequence	0x00	0x00	0x00	Reserved
Sequence	0x00	0x00	0x01	Local Fault
Sequence	0x00	0x00	0x02	Remote Fault
Sequence	0x00	0x00	0x03	Link Interruption
Sequence	**0x00**	**0x00**	**0x04**	**Cipher ON**
Sequence	**0x00**	**0x00**	**0x05**	**Cipher OFF**
Sequence	≥0x00	≥0x00	≥0x06	Reserved

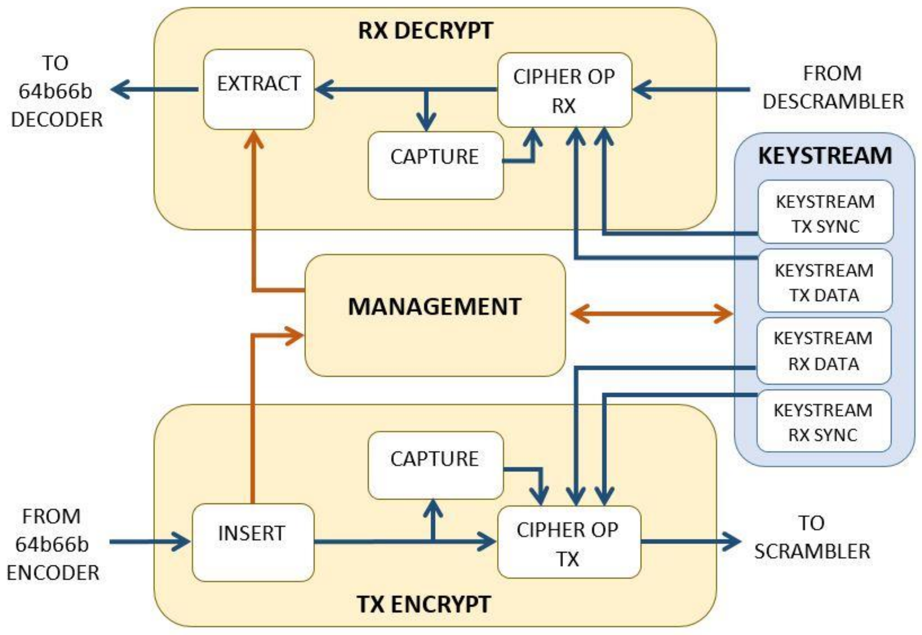


Fig. 5.12. Encryption infrastructure for 10G-PHYsec function.

For carrying out the insertion of these new control sequences the structure shown in Fig. 5.12 is used. The CIPHER_OP_TX module is responsible for implementing the encryption operation. Initially, it is disabled, allowing the 64b/66b blocks to be transparently passed from the encoder to the scrambler. To start the encryption of the 64b/66b block stream, the MANAGEMENT module acts on the INSERT module to send the encryption start message. This message is introduced into the 64b/66b block stream by replacing one 0x1e type block filled with IDLE control characters with the new Cipher_ON block. When CAPTURE module detects the presence of a Cipher_ON block it enables the CIPHER_OP_TX module and TX keystream generators (KEYSTREAM TX SYNC and KEYSTREAM TX DATA in Fig. 5.12), which starts the encryption process after Cipher_ON block has been transmitted.

In the receiver, the module CIPHER_OP_RX is in charge of performing the decryption operation. Like the transmitter, it is initially inactive, enabling 64b/66b blocks to be transparently passed from the descrambler to the decoder. When the CAPTURE module receives the Cipher_ON block, CIPHER_OP_RX module and RX keystream generator modules (KEYSTREAM RX SYNC and KEYSTREAM RX DATA in Fig. 5.12) are enabled, starting to decrypt the 64b/66b data flow after Cipher_ON block. Subsequently, the control sequence Cipher_ON is extracted from the data stream in the EXTRACT module, which replaces it with a 0x1e type block filled with control characters (reverse operation of the INSERT module). Thanks to this procedure, the keystream generators in TX and RX are synchronized and data can be deciphered correctly.

In order to disable the encryption, the same process is used, but sending Cipher_OFF blocks instead of Cipher_ON.

5.3.2. Encryption algorithm

As explained before, the encoded blocks are composed by a 2-bit synchronization header plus a 64-bit payload, forming a 66-bit block. The synchronization header always has a bit transition (‘01’ or ‘10’) to guarantee the 66-bit block alignment and limiting the run length to 66 bits. To keep these properties, it is necessary that the encryption also preserves a bit transition.

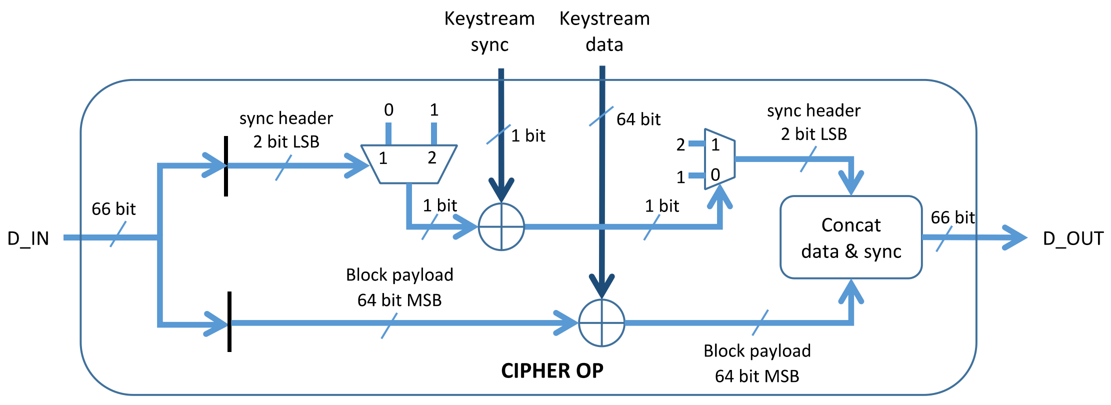


Fig. 5.13. Stream cipher operation.

The stream cipher operation is shown in Fig. 5.13. Both modules CIPHER_OP_TX and CIPHER_OP_RX perform the same operation. Firstly, the 66-bit block, D_IN, is split in two parts, the synchronization header and the block payload. The block payload is directly encrypted by performing an XOR operation between its 64 bits and a 64-bit keystream data sequence, called 'keystream data' in Fig. 5.13. The 2-bit synchronization header is mapped to values '0' or '1' depending, respectively, on whether it is equal to '01' or '10'. Then, the mapped value is encrypted performing an XOR operation with a 1-bit keystream sequence called 'keystream sync'. After encryption, this new value is reverse mapped resulting in a new header '01' or '10' depending on whether it is '0' or '1'. In this way, the transition between '0' and '1' every 66 bits is guaranteed. Finally, the new synchronization header is concatenated with the encrypted block payload resulting in the output D_OUT, which is sent to the scrambler when encrypting or to the 64b/66b decoder when decrypting.

As the keystream generator, we have chosen the algorithm proposed in Section 3.5 and used in Section 5.2.2 but, in this case, the 16 LSBs of each x_i have been used to form the keystream (Fig. 5.14). The reason for using the 16 LSBs (instead of 8) was to double the throughput of each chaotic cell, although it could affect the security of the system. Since, for the payload encryption, a 64-bit keystream sequence is necessary, a bank consisting of four 16-bit generators whose outputs are concatenated to give a 64-bit output has been built (Fig. 5.15).

For the synchronization header encryption, since only a 1-bit keystream sequence is needed, the basic STM cell in Fig. 5.14 has been used, but taking only the LSB of each x_i as output.

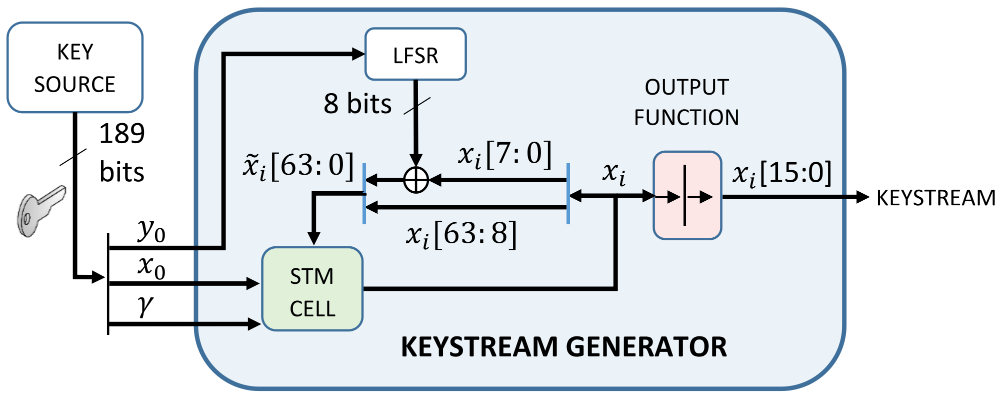


Fig. 5.14. Proposed keystream generator. The 189-bit key is the concatenation of the three parameters (y_0, x_0, γ).

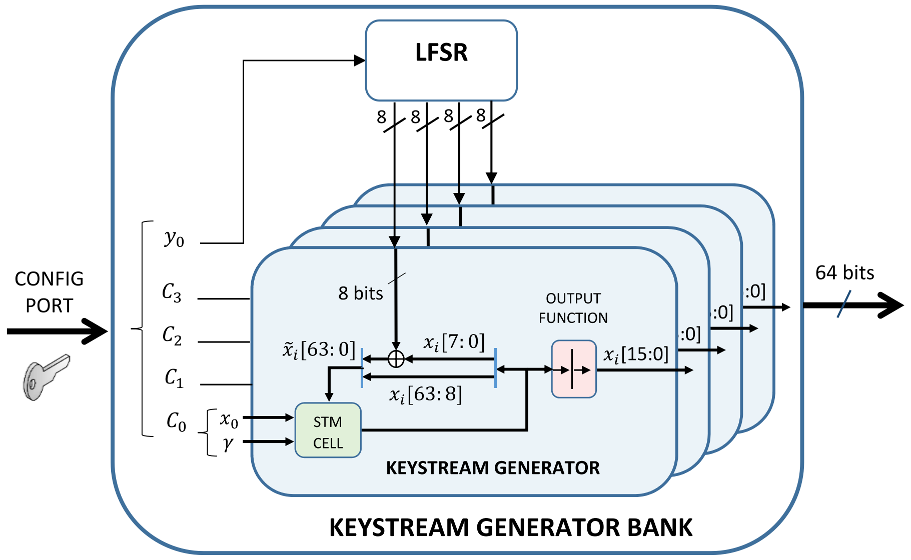


Fig. 5.15. 64-bit keystream generator for the 64-bit payload.

5.3.3. Implementation results

The complete system has been implemented in a Xilinx Virtex 7 FPGA [PER18c, PER19d]. Regarding the resources used by this encryption system, the main contribution for each block inside each keystream generator (KEYSTREAM GEN) is shown in Table 5.4. As it can be seen, the keystream generator bank used for the payload encryption (STM_BANK) takes up the largest amount of FPGA resources.

The test set up is similar as the one presented in the 1 Gb case (Fig. 5.7 and Fig. 5.8). The PHY, including the proposed 10G-PHYsec function is connected directly to the SFP+ modules thanks to the FPGA SERDES circuit while, in the MAC side, Ethernet Interface is connected to the Ethernet Frame Generator to

Table 5.4. FPGA resources in the keystream generator submodules

MODULE	Slice LUTs	Slice Registers	DSPs
KEYSTREAM_GEN*	4680	2271	80
LFSR	202	186	0
STM_BANK	3534	1569	64
STM_1BIT	948	516	16

*KEYSTREAM_GEN refers to the hardware resources of the Keystream Generator, including LFSR, STM_BANK and STM_1BIT.

test the encrypted link with real traffic. The PHY is connected directly to the SFP modules thanks to the FPGA SERDES circuit. In the MAC side, the Ethernet Interface is connected to the Ethernet frame generator to test the encrypted link with real traffic and verify that no frames are lost and no Cyclic Redundancy Check (CRC) errors are produced during encryption.

With this setup, we have checked that the system is capable of encrypting and decrypting Ethernet frames properly, without harming the data traffic or link establishment between the transmitter and receiver interfaces. For this purpose, several traffic bursts of 10^6 frames, each of them with a 1024-byte length, have been tested. In this case, the throughput has also been configured between 10% and 98% of the maximum line rate. Thanks to the CRC counters we have checked that there are no frame losses or CRC errors.

5.3.4. Security analysis

As explained before, by performing the encryption at the physical layer, we intend to achieve two objectives: encrypt the information data (like in any other encryption system) and hide the traffic pattern, so that an attacker cannot know if a communication is taking place.

Regarding the data encryption, although the encryption algorithm based on the STM and the LFSR has been proven to be secure when the 8 LSBs are used to form the keystream (Section 3.5), in this case, the 16 LSBs are used to form the keystream to encrypt the payload. To see how this issue can affect the security of the system, several sequences have been generated and subjected to the NIST randomness tests. The measured average passing rate has been 0.901, which is lower than the ideal one (0.99) and the one obtained by using only the 8

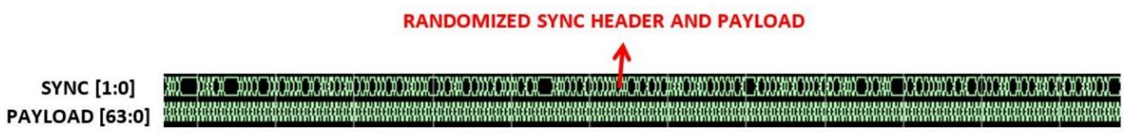


Fig. 5.16. Synchronization headers and payloads after performing the proposed encryption.

LSBs (0.989). Therefore, we can conclude that the security level in this case is lower than the ideal one. To solve this issue, in a future implementation, it could be possible to use a bank of 8 chaotic cells (instead of 4), each of them using only the 8 LSBs to form the keystream. This way, the security would be increased at a cost of increasing the implementation area.

Regarding the synchronization header encryption, since a chaotic cell that uses only the LSB is being used, we can conclude that those bits are properly encrypted.

Finally, in Fig. 5.16, it can be seen how, with the proposed encryption method, the synchronization headers are randomized so that it is not possible for an attacker to know if communication is taking place.

5.4. Conclusions

In this chapter, the chaotic algorithm proposed in Section 3.5 have been applied to encrypt 1 Gb and 10 Gb Optical Ethernet communications at the physical layer. Since in the physical layer the coding and decoding is performed (8b/10b in 1 Gb and 64b/66b in 10 Gb), the encryption algorithm has been adapted to preserve the coding.

In the case of the 8b/10b coding, since there are only 267 possible input symbols, a modulo-267 operation has been applied to a 73-bit pseudorandom bitstream to obtain an integer uniformly distributed between 0 and 266. To generate this 73-bit bitstream, a bank of ciphers as the one proposed in Section 3.5 has been used.

In the case of the 64b/66b coding, to encrypt the synchronization header, it has been mapped to a single bit, XORed with a pseudorandom bit, and de-mapped to obtain an encrypted header. To encrypt the 64-bit payload, it has been XORed with a 64-bit pseudorandom bitstream. In this case, to generate the pseudorandom bitstream, the same chaotic algorithm has been used but extracting the 16 LSBs instead of the 8 LSBs.

The proposed encryption systems have been implemented in a Xilinx Virtex 7 FPGA and tested with real Ethernet traffic. In both cases, the encryption and the decryption have worked properly, achieving a maximum line rate and adding a very low latency. Regarding the extra resources needed to implement the encryption functions, in the case of 1 Gb, the keystream generator have added 10943 LUTs, 3629 registers and 144 DSPs in the transmitter while, in the case of 10 Gb, the keystream generator has added a total of 4680 LUTs, 2271 registers and 80 DSPs. It must be noticed that the keystream generator used in the 10 Gb case uses almost 50% less resources than the one used in the 1 Gb case despite achieving 10 times more throughput. This is due to two factors: first, in the 10 Gb case the 16 LSB of each chaotic cell are used; second, in the case of a 64b/66b coding, it is not necessary to use a modulo operation so a huge amount of resources is saved (not only because of the modulo operation but also because it is not necessary to add 64 extra bits in the input to prevent the bias after performing this operation).

By analyzing the security of these systems we have concluded that the encryption system proposed for 1 Gb communications is secure and that the modulo 267 operation does not add any significant bias. In the case of 10 Gb, we have measured that, due to the fact that the 16 LSBs have been used as part of the output, the average NIST passing rate is 0.901 so the security could be compromised. To solve this problem in future implementations it could be possible to use more chaotic cells and using only the 8 LSBs in each case. This way, the system would be secure at a cost of approximately doubling the implementation area.

Finally, in both cases, we have checked that, by encrypting not only the data symbols but also the control symbols and the flag bits or synchronization headers, the data traffic is completely hidden so that it is not possible for an attacker to know if a communication is taking place. This property adds extra security to the system and it is a considerable improvement with respect to other Ethernet encryption standards such as IPsec or MACsec.

5.5. References

[BAL13] N. Balakrishnan, Vassilly Voinov, M. S. Nikulin, “Chi-Squared Goodness of Fit Tests with Applications,” *Waltham, MA, Academic Press, Elsevier*, 2013.

[BAR15] E. Barker, J. Kesley, “Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Appendix A),” *NIST Special Publication 800-90A*, 2015.

[BUT11] J. Buttler, T. Sasao, “Fast Hardware Computation of x Mod z,” *Proceedings of the Parallel and Distributed Processing Workshops and PhD Forum (IPDPSW)*, 2015.

[IEE18] IEEE Standards Association, “IEEE Standard for Ethernet,” *IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015)*, pp. 1-5600, 2018.

[KNU81] D. Knuth, “The Art of Computer Programming, Volume 2: Seminumerical Algorithms” *Addison-Wesley*, 1981.

[LAZ17] J. Lázaro, A. Astarloa, J. Araujo, N. Moreira, U. Bidarte, “MACsec Layer 2 Security in HSR Rings in Substation Automation Systems,” *Energies*, vol. 10, no. 2, pp. 162, 2017.

[MAR13] Marko Bobinac, “Layer 2 Network Encryption – Where Safety is not an Optical Illusion,” *SafeNet Day*, 2013.

[MIC16] MicroSemi, “In-Flight Encryption in Service Provider Networks,” *Document PMC-2150716,* no. *2*, 2016.

[RUK10] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” *NIST Special Publication 800-22 Rev.1.a,* 2010.

[SAU11] T. Sauter, M. Lobashov, “How to Access Factory Floor Information Using Internet Technologies and Gateways,” *IEEE Transactions on Industrial Informatics,* vol. 7, no.4, pp. 699-712, 2011.

[SMI04] D. R. Smith, “Digital Transmission Systems,” *Springer-Verlag,* 2014.

[PER18a] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, “Using a Chaotic Cipher to Encrypt Ethernet Traffic,” *Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS 2018),* 2013.

[PER18b] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, “Chaos-Based Stream Cipher for Gigabit Ethernet,” *Proceedings of 9th IEEE Latin American Symposium on Circuits and Systems (LASCAS 2018),* 2018.

[PER18c] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, “Chaotic Encryption Applied to 10 Gbps Ethernet Optical Links,” *Proceedings of 2018*

International Symposium on Nonlinear Theory and its Applications (NOLTA 2018), 2018.

[PER19a] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, "Chaotic Encryption Applied to Optical Ethernet in Industrial Control Systems," *IEEE Transactions on Instrumentation and Measurement,* 2019 (early access).

[PER19b] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, "Physical Layer Encryption for Industrial Ethernet in Gigabit Optical Links," *IEEE Transactions on Industrial Electronics,* vol. 66, no. 4, pp. 3287-3295, 2019.

[PER19c] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, "Self-Synchronized Encryption for Physical Layer in 10 Gbps Optical Links," *IEEE Transactions on Computers,* vol. 68, no. 6, pp. 899-911, 2019.

[PER19d] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, "Chaotic Encryption for 10-Gb Ethernet Optical Links," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 66, no. 2, pp. 859-868, 2019.

[TRO05] L. Troell, J. Burns, K. Chapman, D. Goddard, M. Soder-Lund, C. Ward, "Converged vs. Dedicated IPSec Encryption Testing in Gigabit Ethernet Networks," *Rochester Institute of Technology, Technical Report 1743,* Available online: http://scholarworks.rit.edu/article/1743 (accessed on 30 June 2019), 2005.

[XEN06] C. Xenakis, N. Laotaris, L. Merakos, I. Stavrakakis, "A Generic Characterization of the Overheads Imposed by IPsec and Associated Cryptographic Algorithms," *Computer Networks,* vol. 50, no.17, pp. 3225-3241, 2006.

6 Conclusions

In the four chapters that form the core of this Thesis, the most relevant conclusions concerning each chapter have been abridged. Finally, this last chapter summarizes the general conclusions, clarifying the main results and highlighting its novelties and contributions to the state of the art in the fields of chaos-based cryptography, true random number generation and Optical Ethernet encryption.

This chapter is divided in two sections. First, the degree of fulfillment of the objectives described in Chapter 1 is reviewed and general conclusions are described. Then, future research lines derived from the general conclusions are pointed out. These future lines are questions not considered in this Thesis that could enhance the reported results and a development of other questions that have already been addressed.

6.1. General conclusions

As explained in the introduction, most of the chaos-based cryptosystems that had been proposed so far could not achieve enough speed for Gigabit Ethernet Communications required a big amount of area to be implemented, lacked a strict security analysis or did not address the key generation process.

The main goal of this Thesis was to design a new secure physical layer encryption scheme for Gigabit Ethernet. To implement the full communication scheme, it was necessary to design new encryption algorithms as well as new TRNG's suitable for key generation. Finally, to perform the encryption at the physical layer, the algorithms had to be adapted to preserve the data coding. By reviewing the initial objectives, we have seen that, overall, they have been fulfilled:

- A study of the state of the art has been made, determining that, although the chaotic ciphers can be a good alternative to classic encryption, they present some problems, mainly due to the digitization process, that need to be addressed in order to use them in secure communication schemes.
- With the purpose of designing new encryption algorithms fast, secure, and implementable with a small amount of resources, we have focused on chaos based stream ciphers. This choice has been made because chaotic systems have some intrinsic properties such as ergodicity and random-like behavior that are closely related with the cryptographic properties of confusion and diffusion. To overcome the issues that arise when chaotic maps are digitized, several strategies have been proposed and tested in two simple chaotic maps: STM and LM. The first strategy has consisted of using a multi-encryption scheme based on a DWRR, the second one has consisted of changing the chaotic parameter dynamically and the third one has consisted of using an LFSR to perturb the chaotic orbits. In each case, several stream ciphers have been implemented and compared changing some parameters such as: the underlying chaotic map (STM or LM), precision (32 or 64 bits), number of bits of each state used as the output, etc. After testing several designs we have concluded that, with the last strategy, the best results can be obtained in terms of area, speed and security.
- A final implementation of a stream cipher based on a 64-bit STM that uses an LFSR to perturb the orbits have been made in a Virtex 7 FPGA. The stream cipher has achieved a throughput of 1392 Mbps using 192 slices (7.25 Mbps/slice) and 16 DSPs. This algorithm has also been implemented in an ASIC using a 0.18-µm technology and has achieved encryption speeds of at least 1 Gbps using approximately 20,000 2-NAND equivalent gates. In both cases, the implementation results have been compared with other proposed

chaos-based encryption algorithms implemented in similar platforms (FPGA or ASIC). With this comparison, we have determined that the proposed encryption algorithm obtain better results in terms of encryption speed per area (approximately a factor of 2 in the ASIC implementation), which implies a significant improvement with respect to the state of the art.

- In order to use a TRNG for key generation, two approaches have been studied: using the noise generated by a sensor and using the jitter that is typically present in DNOs. In the first case, the signal generated by a MEMS accelerometer at rest have been sampled at different frequencies and used to generate a binary bitstream by eliminating the DC level and applying a sign detection. In the second case, three different DNO-based TRNGs have been implemented in an Arty board (which includes an Artix 7 FPGA): the first one was based on a simple ring oscillator, the second one consisted on a Galois ring oscillator proposed by Golić and the third one was a novel structure similar as the second one, but using delays instead of inverters. To evaluate and compare the quality of all of the tested generators, some statistical tests have been applied. This way we have checked that all of them are good entropy sources and, therefore, can be used for true random number generation as long as the sampling frequency is low enough. For the same sampling frequency, the DNO-based TRNGs have been capable of generating sequences with higher entropy than the MEMS-based TRNGs and, among the DNOs, the proposed structure seems to be the best entropy source. Due to the bias present in these generators, neither of them have been capable of passing the NIST test. Therefore, a simple post-processing stage have been included consisting on XORing the sequences with sequences generated by a 4-order linear feedback shift register. This way, the MEMS-based TRNG have passed the NIST tests for sampling frequencies equal or lower than 2.5 kHz while, the DNO-based TRNG using the proposed structure have passed the NIST tests for sampling frequencies equal or lower than 2 MHz.
- Finally, the previously proposed encryption algorithm has been adapted to be used in 1 Gb and 10 Gb Optical Ethernet communications. In both cases, an encryption system that works at the physical layer preserving the coding (8b/10b or 64b/66b) has been implemented in a Virtex 7 FPGA. The

cryptosystems have been tested with real Ethernet traffic by using two SFP modules and have been capable of encrypting Ethernet packages at line rate introducing a very low latency. The proposed encryption systems encrypt both the data and the control symbols so, apart from the standard security that can be expected from an encryption algorithm, these cryptosystems can also hide the data traffic so that a possible attacker cannot know if a communication is taking place. As far as we know, there are not currently any encryption algorithms for Optical Gigabit Ethernet that achieve this property so this work is a significant contribution to the state of the art.

It must be noticed that additional aspects have been developed during this Thesis that have contributed to the successful achievements of the reported results. First, the statistical tests (pattern distribution of bytes, correlation coefficients and ASR) used to evaluate the quality of the proposed TRNGs have allowed as to easily compare visually the randomness (including both the bias and the statistical independency) of the generated sequences. This way, we have obtained more information about each generator than what we would have obtained if we had only used the NIST randomness tests. Second, for implementing the DNOs in FPGAs it has been necessary to learn how to force the synthesizer to maintain the hierarchy (for example, prevent the synthesizer from synthetizing an odd number of inverter as simply one inverter) as well as to change the design rules to allow the implementation of combinational loops. Furthermore, in this case, the locations of the LUTs and registers have been fixed. To achieve this, LUT primitives as well as special Xilinx instructions have been used to force the location of each LUT or register within the FPGA.

6.2. Future research lines

In this Thesis, we have not used a TRNG for key generation in the final cryptosystems implemented for Optical Gigabit Ethernet communications in Chapter 5. Therefore, regarding the future research lines, we plan to include it in a future implementation once we have determined which of the proposals is better considering some aspects such as generation speed, randomness of the generated sequences or robustness of the system. Furthermore, in the case of 10 Gb Ethernet, an alternative implementation where only 8 LSBs (instead of 16) are used to generate the keystream could be made to achieve a higher level of security. This way, an average NIST passing rate of approximately 0.99 (instead of the previous 0.901) could be obtain at a cost of approximately, dividing the throughput per area by a factor of 2.

In the case of the proposed stream cipher, although a cryptanalysis has been made and it is apparently secure, this cryptanalysis has focused only on the encryption algorithm, without taking into account its implementation. Therefore, it could be vulnerable against side channel attacks, consisting of extracting information of the system by analyzing some parameters such as its power consumption, execution time or electromagnetic radiation. In a similar manner, it could also be vulnerable against fault attacks, consisting of introducing erroneous calculations leading to the exposure of the key. Therefore, the security of the implemented algorithms against these attacks should be evaluated and, in case that they are vulnerable, some countermeasures should be taken such as modifying the implementation of the algorithm or even modifying the algorithm. On the other hand, although the encryption algorithm seems to be secure, new attacks are constantly being proposed so, in the future, some vulnerabilities might be found. It is crucial to keep analyzing this algorithm trying to find some possible vulnerabilities and fixed them if it is possible.

Another research line would be to study the possibility of implementing the proposed stream cipher in IoT devices. In this case, it might be convenient to modify the algorithm to reduce its power consumption by reducing the encryption rate. If it is not possible, a future research line would be to design new encryption algorithms (not necessarily chaotic) that are secure and suitable for IoT applications.

Regarding the TRNGs, it would be interesting to continue the study of DNO-based TRNGs. This study could include some aspects such as their behavior depending on the temperature, the FPGA where they have been implemented or their location within the FPGA. Furthermore, new structures that achieve better randomness or are more robust could be proposed. A good parameter to evaluate the quality of each structure would be their minimum entropy. To find good structures, a possible option would be to find a way of modeling these systems so that they can be simulated accurately. This way, many possible topologies (changing the number of LUTs, the feedbacks or the functions used) could be simulated until a topology that obtains good results is found. On the other hand, it would also be convenient to construct and stochastic model of these kind of systems, capable of estimating their entropy.

Finally, a topic that is tightly related to this Thesis is the design of Physically Unclonable Functions (PUFs). While these functions can have several applications such as device authentication, in this work they could be used as an alternative to TRNG for secure key generation and storage.

A

NIST Statistical Test Suite

The NIST statistical test suite consists of a software provided by the National Institute of Standards and Technology which applies a set of statistical tests to a given set of sequences to determine if they are random [RUK10].

The list of applied tests with a brief description are presented below:

1. Frequency (Monobits Test)

This tests determine whether the number of ones and zeros in a sequence are approximately the same as would be expected in a truly random sequences.

2. Test for Frequency within a Block

This test measures the proportion of zeroes and ones within M-bit blocks. The purpose of this test is to determine whether the frequency of ones in each block is approximately M/2.

3. Runs Tests

This test analyzes the runs of zeros and ones in the entire sequence, where a run is an uninterrupted sequence of identical bits. The purpose of this tests is to determine if the number of runs of ones and zeroes of various lengths is as expected for a random sequence.

4. Test for the Longest Run of Ones in a Block

This test measures the longest run of ones within M-bit blocks to determine if they are consistent with the length of the longest run of ones that would be expected in a random sequence.

5. Random Binary Matrix Rank Test

This test calculates the rank of disjoint sub-matrices of the sequence to check for linear dependence among fixed length substrings.

6. Discrete Fourier Transform (Spectral) Test

This test calculates the peak heights in the discrete Fast Fourier Transform. This test is used to detect periodic features in the sequence.

7. Non-Overlapping (Aperiodic) Template Machine Test

This test measures the number of occurrences of some pre-defined target substrings. If there are too many occurrences of some patterns, the sequence is considered to be non-random. For this test, an m-bit window is used to search for a specific *m*-bit pattern. If the pattern is not found, the window slides one bit position. When the pattern is found, the window is reset to the bit after the found pattern and the search resumes.

8. Overlapping (Periodic) Template Matching Test

This test also measures the number of occurrences of some pre-defined target substrings. The difference between this test and the previous one is that when the pattern is found, the window slides only one bit before resuming the search.

9. Maurer's Universal Statistical Test

This test determines if the sequence can be significantly compressed without loss of information. A compressible sequence is considered not random.

10. Linear Complexity Test

This test measures the length of a generating feedback register. Random sequences are characterized by a longer generating feedback register.

11. Serial Test

This test measures the frequency of all possible overlapping *m*-bit patterns across the entire sequence. If the sequence is random, the number of occurrences of each one of the 2^m *m*-bit patterns should be approximately the same. For m=1 this test is equivalent to the Frequency Test.

12. Approximate Entropy Test

This test also focuses on all possible overlapping *m*-bit patterns across the entire sequence. The purpose of the test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (*m* and *m*+1) against the expected result for a random sequence.

13. Cumulative Sum Test

This test transforms the ‘0’ digits into -1 and perform the cumulative sums of partial sequences and determines if it is too large or too small compared to the expected behavior of that cumulative sum for random sequences.

14. Random Excursion Test

This test focuses on the number of cycles having exactly *K* visits in a cumulative sum random walk. A cycle of a random walk consists of a sequence of steps of unit length taken at random that begin at and return to the origin. This test determines if the number of visits to a particular state within a cycle deviates from what one would expect from a random sequence.

15. Random Excursion Variant Test

This test focuses on the total number of times that a particular state is visited (occurs) in a cumulative sum random walk and determines if it deviates from what one would expect from a random sequence.

16. Cumulative Sum Test Reverse

Is the same as test 13 but working in reverse mode. If each value of the sequence (with the ‘0’ digits transformed to -1) is X_k, in test 13, the cumulative sums are calculated as $S_k = S_{k-1} + X_k$ (with $S_1 = X_1$) while in this test the cumulative sums and are calculated as $S_k = S_{k-1} + X_{n-k+1}$ (with $S_1 = X_n$).

17. Lempel-Ziv Compression Test[2]

This test measures the number of cumulatively distinct patterns (words) in a sequence to determine how far the sequence can be compressed. If it can be significantly compressed, the sequence is considered to be non-random.

[2] This Test was removed from the Statistical Test Suite in revision 2008 and has not been incorporated since then.

[RUK10] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” *NIST Special Publication 800-22 Rev.1.a,* 2010.

B

TSMC 0.18 μm CMOS Logic or MS/RF, General Purpose 1.8V/3.3V

TSMC 0.18- μm technology is a single poly, six metal layer process with low-k dielectrics.

The main property of the process is the option of using copper interconnects in the top two metal layers instead of aluminum ones. This way, it is possible to reduce interconnect resistance and make higher speeds possible.

Another feature of this process is the Shallow Trench Isolation (STI), which improves surface planarity, compared to older isolation techniques, making it possible to achieve high-reliability metallization. STI also reduces capacitive coupling between adjacent transistors, increasing circuit density and reducing power consumption.

Some major constraints of the design presented in Section 3.6 were a maximum clock period of 14 ns, load ranging from 0.01 to 1.0 pf at outputs, and drive up to 0.4 kΩ at inputs. This set of constraints was enough to implement satisfactorily the proposed cipher.

The main process characteristics for the TSMC 0.18 μm technology are shown in the following table:

Table B.1. Technology characteristics

Feature	Physical property
Metal layers	3 to 6
Contacted metal pitch	M1 0.46 μm, M2-5 0.56 μm, M6 0.90 μm
Core voltage	1.8 V
I/O voltage option	3.3 V
Temperature range	-40 ºC to 125 ºC
Interconnect material	AlCu
Interconnect dielectric	FSG (low-k)
Isolation	Shallow Trench Isolation (STI)
6T SRAM Cell	4.65 μm^2

For the digital design proposed in Section 3.6, the TCB018GBWP7T library has been used. This library contains 568 different cells, including all the combinational, storage and other special cells necessary to complete the digital design. As an example, the main parameters of a 2-NAND gate are shown in the table below

Table B.2. 2-NAND gate information

Cell Name	Gate Count	Width (μm)	Leakage (nW)			Propagation Delay (Worst Case)
			Min.	Ave.	Max	
ND2D0BWP7T	1	2.24	0.04451	0.1243	0.21569	0.0857+8.799*Cload
ND2D1BWP7T	1	2.24	0.08034	0.2191	0.37188	0.0806+4.3015*Cload
ND2D1P5BWP7T	2	3.92	0.12145	0.3394	0.58757	0.0799+2.8893*Cload
ND2D2BWP7T	2	3.92	0.15887	0.4369	0.74375	0.0767+2.1557*Cload
ND2D2P5BWP7T	3	5.04	0.20208	0.5591	0.95944	0.0795+1.7289*Cload
ND2D3BWP7T	3	5.04	0.23907	0.6559	1.11562	0.0801+1.4343*Cload
ND2D4BWP7T	4	6.72	0.31831	0.8743	1.4875	0.0804+1.0761*Cload
ND2D5BWP7T	5	8.4	0.39775	1.093	1.85937	0.0791+0.8615*Cload
ND2D6BWP7T	6	9.52	0.47698	1.311	2.23125	0.0809+0.7173*Cload
ND2D8BWP7T	8	12.32	0.63563	1.748	2.975	0.0802+0.5379*Cload

C

Xilinx 7 Series FPGAs

Xilinx 7 Series FPGA comprise four FPGA families ranging from low cost, small form factor, cost-sensitive, high volume applications to ultra high-end connectivity bandwidth, logic capacity and signal processing capability for the most demanding high-performance applications [XIL18a]. The four families are:

Spartan 7: Optimize for low cost, lower power and high I/O performance.

Artix 7: Optimized for low power applications requiring serial transceivers and high DSP and logic throughput.

Kintex 7: Optimized for best price-performance with a 2X improvement compared to previous generations.

Virtex 7: Optimized for highest system performance and capacity with a 2X improvement in system performance.

Zynq 7: These System on Chips integrate the software programmability of an ARM-based processor with the hardware programmability of an FPGA (Artix, Kintex or Virtex) [XIL18b].

During this Thesis, three different boards have been used, each of them including a different FPGA:

Zybo board: This board includes a Zynq 7000 series FPGA, XC7Z010, formed by an ARM processor and an Artix 7 based programmable logic.

Arty board: This board includes an FPGA from the Artix 7 family, XC7A35T.

Virtex 7 VC707 Evaluation Kit: This board includes an FPGA of the Virtex 7 family, XC7VX485T.

Each FPGA is organized as an array of Configurable Logic Blocks (CLBs). In the used FPGAs, each CLB contains two slices and each slice contains 4 6-input LUTs and 8 flip-flops. Some of these slices, can use their LUTs as distributed RAM or Shift Register Lookup Tables (SRLs). In addition to this, the used FPGAs contain Block RAM blocks and Digital Signal Processing (DSP) slices, each of them containing a pre-adder, a 25x18 multiplier, an adder and an accumulator.

To summarize the main features of the used FPGAs are presented below:

Table C.9.1. Mean features of Arty, Zybo and VC707 boards

Board name	**Arty**	**Zybo**	**VC707**
FPGA family	Artix 7	Zynq 7000	Virtex 7
Device	XC7A35T	XC7Z010	XC7VX485T
Logic Cells	33,280	28,000	485,760
Slices	5,200	4,400	75,900
LUTs	20,800	17,600	303,600
Flip-Flops	41,600	35,200	607,200
Max Distributed RAM (Kb)	400	375	8,175
DSP slices	90	80	2,800
Block RAM Blocks (18 Kb)	100	120	2,060
Block RAM Blocks (36 Kb)	50	60	1,030
Maximum Block Ram (Kb)	1,800	2,160	37,080

[XIL18a] Xilinx, “7 Series FPGAs Data Sheet: Overview,” 2018. Available at: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf, accessed June 2019.

[XIL18b] Xilinx, “Zynq-7000 SoC Data Sheet: Overview,” 2018. Available at: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf, accessed June 2018.

D

List of own publications

Publications in Journals

[PER19a] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, "Chaotic Encryption Applied to Optical Ethernet in Industrial Control Systems," *IEEE Transactions on Instrumentation and Measurement,* 2019 (early access).

[GAR19a] M. Garcia-Bosque, G. Díez-Señorans, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, S. Celma, "A 1 Gbps Chaos-Based Stream Cipher Implemented in 0.18□m CMOS Technology," *MDPI Electronics*, vol. 8, no. 6, pp. 1-10, 2019.

[PER19c] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, "Self-Synchronized Encryption for Physical Layer in 10 Gbps Optical Links," *IEEE Transactions on Computers,* vol. 68, no. 6, pp. 899-911, 2019.

[PER19b] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, "Physical Layer Encryption for Industrial Ethernet in Gigabit Optical Links," *IEEE Transactions on Industrial Electronics,* vol. 66, no. 4, pp. 3287-3295, 2019.

[PER19d] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, "Chaotic Encryption for 10-Gb Ethernet Optical Links," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 66, no. 2, pp. 859-868, 2019.

[GAR18d] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, S. Celma, "A New Simple Technique for Improving the Random Properties of Chaos-Based Cryptosystems," *AIP Advances*, vol. 8, no. 3, pp. 035004-1:035004-10, 2018.

[GAR18e] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, S. Celma, "Chaos-Based Bitwise Dynamical Pseudorandom Number Generator On FPGA," *IEEE Transactions on Instrumentation and Measurements*, vol. 68, no. 1, pp. 291-293, 2018.

[GAR17b] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, S. Celma, “Application of a MEMS-Based TRNG in a Chaotic Stream Cipher,” *MDPI Sensors*, vol. 17, no. 3, pp. 1-15, 2017.

Publications in International Conferences

[PER19e] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, “Self-Synchronized Encryption Using an FPE Block Cipher for Gigabit Ethernet,” *Proceedings of the 15th Conference on PhD Research in Microelectronics and Electronics (PRIME 2019),* 2019.

[GAR19b] M. Garcia-Bosque, G. Díez-Señorans, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, S. Celma, “A New Lightweight CSPRNG Implemented in a 0.18 □m CMOS Technology,” *Proceedings of the 15th Conference on PhD Research in Microelectronics and Electronics (PRIME 2019),* 2019.

[ADD19] T. Addabbo, A. Fort, R. Moretti, M. Mugnaini, V. Vignoli, M. Garcia-Bosque, “Lightweight True Random Bit Generators in PLDs: Figures of Merit and Performance Comparison,” *Proceedings of the 2019 International Symposium on Circuits and Systems (ISCAS 2019),* 2019.

[PER18c] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, “Chaotic Encryption Applied to 10Gbps Ethernet Optical Links,” *Proceedings of the 2018 International Symposium on Nonlinear Theory and its Applications (NOLTA 2018),* 2018.

[GAR18a] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, S. Celma, “A Multi-Encryption Scheme and its Application to Improve the Random Properties of a Chaos-Based Stream Cipher,” *Proceedings of the 2018 International Symposium on Nonlinear Theory and its Applications (NOLTA 2018),* 2018.

[PER18a] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, “Using a Chaotic Cipher to Encrypt Ethernet Traffic,” *Proceedings of the 2018 International Symposium on Circuits and Systems (ISCAS 2018),* 2018.

[ADD18] T. Addabbo, A. Fort, M. Mugnaini, V. Vignoli, M. Garcia-Bosque, “Digital Nonlinear Oscillators in PLDs: Pitfalls and Open Perspectives for a Novel Class of True Random Number Generators,” *Proceedings of the 2018 International Symposium on Circuits and Systems (ISCAS 2018),* 2018.

[GAR18c] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, S. Celma, “A New Technique for Improving the Security of Chaos-Based Cryptosystems,” *Proceedings of the 2018 International Symposium on Circuits and Systems (ISCAS 2018),* 2018.

[GAR18b] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, S. Celma, “A New Randomness-Enhancement Method for Chaos-Based Cryptosystems,” *Proceedings of the 9th IEEE Latin American Symposium on Circuits and Systems (LASCAS 2018),* 2018.

[PER18b] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma, “Chaos-Based Stream Cipher for Gigabit Ethernet,” *Proceedings of the 9th IEEE Latin American Symposium on Circuits and Systems (LASCAS 2018),* 2018.

[GAR17a] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, S. Celma, “A New Multiple Ciphering Scheme for Improving Randomness,” *Proceedings of the 23th European Conference on Circuit Theory and Design (ECCTD 2017)*, 2017.

[ROY17] G. Royo, M. Garcia-Bosque, C. Sánchez-Azqueta, C. Aldea, S. Celma, C. Gimeno, “Transimpedance Amplifier with Programmable Gain and Bandwidth for Capacitive MEMS Accelerometers,” *Proceedings of the 2017 IEEE International Instrumentation and Measurement Technology Conference*, 2017.

[AGU17] J. Aguirre, E. Guerrero, C. Sánchez-Azqueta, A. D. Martínez, M. Garcia-Bosque, C. Gimeno, S. Celma, “Continuous-Time Equalizer for CMOS Integrated Photodiodes,” *Proceedings of the 2017 IEEE International Instrumentation and Measurement Technology Conference*, 2017.

[GAR17c] M. Garcia-Bosque, A. Pérez, C. Sánchez-Azqueta, S. Celma “Fast and Secure Chaotic Stream Cipher with a MEMS-Based Seed Generator,” *Proceedings of the 2017 IEEE International Instrumentation and Measurement Technology Conference*, 2017.

[GAR17d] M. Garcia-Bosque, A. Pérez, C. Sánchez-Azqueta, G. Royo, S. Celma “MEMS-Based Seed Generator Applied to a Chaotic Stream Cipher,” *Proceedings of SPIE Microtechnologies*, 2017.

[GAR16c] M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma “Sensor-Based Seeds for a Chaotic Stream Cipher,” *Proceedings of EUROSENSORS 2016*, 2016.

[GAR16a] M. Garcia-Bosque, C. Sánchez-Azqueta, G, Royo, S. Celma “Lightweight Ciphers Based on Chaotic Map-LFSR Architectures,” *Proceedings of the 12th Conference on PhD Research in Microelectronics and Electronics (PRIME 2016)*, 2016.

[GAR16b] M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma “Secure Communication System Based on a Logistic Map and a Linear Feedback Shift Register,” *Proceedings of the 2016 International Symposium on Circuits and Systems (ISCAS 2016)*, 2016.