UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: Implementation and Analysis of authentication protocols in the transmission of information by radiofrequency

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero en Tecnologías de la Información

Autor:

Juan David Callataxi Suárez

Tutor:

Cristian René Iza Paredes, Ph.D.

Urcuquí, abril del 2021

Urcuquí, 10 de junio de 2021

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN
ACTA DE DEFENSA No. UITEY-ITE-2021-00015-AD

A los 10 días del mes de junio de 2021, a las 16:00 horas, de manera virtual mediante videoconferencia, y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.
Miembro No Tutor	Dr. ARMAS ARCINIEGA, JULIO JOAQUIN , Ph.D.
Tutor	Dr. IZA PAREDES, CRISTHIAN RENE , Ph.D.

El(la) señor(ita) estudiante **CALLATAXI SUAREZ, JUAN DAVID**, con cédula de identidad No. **1722146485**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **TECNOLOGÍAS DE LA INFORMACIÓN**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-43-No.496-2014**, realiza a través de videoconferencia, la sustentación de su trabajo de titulación denominado: **Implementation and Analysis of authentication protocols in the transmission of information by radiofrequency.**, previa a la obtención del título de **INGENIERO/A EN TECNOLOGÍAS DE LA INFORMACIÓN**.

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor	Dr. IZA PAREDES, CRISTHIAN RENE , Ph.D.

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación a través de videoconferencia, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Presidente Tribunal De Defensa	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.	9,0
Miembro Tribunal De Defensa	Dr. ARMAS ARCINIEGA, JULIO JOAQUIN , Ph.D.	9,0
Tutor	Dr. IZA PAREDES, CRISTHIAN RENE , Ph.D.	10,0

Lo que da un promedio de: **9.3 (Nueve punto Tres)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

Certifico que *en cumplimiento del Decreto Ejecutivo 1017 de 16 de marzo de 2020, la defensa de trabajo de titulación (o examen de grado modalidad teórico práctica) se realizó vía virtual, por lo que las firmas de los miembros del Tribunal de Defensa de Grado, constan en forma digital.*

JUAN DAVID CALLATAXI SUAREZ
Firmado digitalmente por JUAN DAVID CALLATAXI SUAREZ
Fecha: 2021.06.18 13:30:52 -05'00'

CALLATAXI SUAREZ, JUAN DAVID
Estudiante

Firmado electrónicamente por:
OSCAR GUILLERMO CHANG TORTOLERO

Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.
Presidente Tribunal de Defensa

CRISTHIAN RENE IZA PAREDES
Digitally signed by CRISTHIAN RENE IZA PAREDES
Date: 2021.06.14 18:04:11 -05'00'

Dr. IZA PAREDES, CRISTHIAN RENE , Ph.D.
Tutor

Firmado electrónicamente por:
JULIO JOAQUIN ARMAS ARCINIEGA

Dr. ARMAS ARCINIEGA, JULIO JOAQUIN , Ph.D.
Miembro No Tutor

TATIANA BEATRIZ TORRES MONTALVAN
Firmado digitalmente por TATIANA BEATRIZ TORRES MONTALVAN
Fecha: 2021.06.10 18:33:31 -05'00'

TORRES MONTALVÁN, TATIANA BEATRIZ
Secretario Ad-hoc

Autoría

Yo, **Juan David Callataxi Suárez**, con cédula de identidad **1722146485**, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Abril del 2021.

JUAN DAVID CALLATA XI SUAREZ

Firmado digitalmente por JUAN DAVID CALLATAXI SUAREZ
Fecha: 2021.07.03 19:56:35 -05'00'

Juan David Callataxi Suárez
CI: 1722146485

Autorización de publicación

Yo, **Juan David Callataxi Suárez** , con cédula de identidad **1722146485**, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Urcuquí, Abril del 2020.

JUAN DAVID CALLATAXI SUAREZ
Firmado digitalmente por JUAN DAVID CALLATAXI SUAREZ
Fecha: 2021.07.03 19:57:18 -05'00'

Juan David Callataxi Suárez
CI: 1722146485

Dedication

To my family who has always supported me and encouraged me to improve myself and always keep moving forward.

Juan David

Acknowledgments

A deep thanks to all the academic staff of the Universidad de Investigación de Tecnología Experimental Yachay for having provided me with an international quality education.

Thank you very much to my advisor Cristhian Iza, for all the teachings and the help provided in the process of developing this thesis.

Thanks to my professors, for instilling in me this great passion for computing and programming.

I thank my friends Giovanny, Inti, Henrry S., Henry C., Ivan, Omar and Alejandro who met them in nivelation course, for all the teachings they left me during my university life. I will miss you so much..

Juan David Callataxi Suarez.

Abstract

In recent years, radio frequency identification (RFID) has become one of the most widely used technologies for the storage and transmission of data in various applications such as inventory control, user access systems, personal information storage, among others. The fundamental purpose of RFID technology is to transmit the identity of an object using radio waves. However, the devices used to send information like radiofrequency readers and RFID tags are vulnerable to various types of attacks. There are a number of measures to safeguard the security of RFID device operations and communications. One of the most important is the authentication protocols. The objective of this project is to evaluate two authentication protocols.These protocols must meet certain standards such as being robust against external threats and light for implementation in passive RFID tags. The first part of the project is the implementation of the protocols in integrated circuits of particular use through the use of VHDL programming language, and ModelSim software for the simulation of the circuits. Both protocols use a pseudorandom number generator algorithm, which provides the information with greater protection to the detection, mainly of the RFID tags. Once the implementation of the protocols has been carried out, in the second part of the project, the data obtained is analyzed mainly data related to the LUT, slices, Flip-flops necessary for the implementation of the protocols and power consumption through the Altera Quartus II and ISE Xilinx Design Suite tool. This allows analyzing and comparing which of the implemented protocols is better adapted to the needs and mainly to the limitation of resources in the RFID tags.

Keywords: RFID, authentication, protocol.

Resumen

En los últimos años, la identificación por radiofrecuencia (RFID) se ha convertido en una de las tecnologías más utilizadas para el almacenamiento y transmisión de datos en diversas aplicaciones como control de inventarios, sistemas de acceso de usuarios, almacenamiento de información personal, entre otras. El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto mediante ondas de radio. Sin embargo, los dispositivos utilizados para enviar información como lectores de radiofrecuencia y etiquetas RFID son vulnerables a varios tipos de ataques. Hay una serie de medidas para salvaguardar la seguridad de las operaciones y comunicaciones de los dispositivos RFID. Uno de los más importantes son los protocolos de autenticación. El objetivo de este proyecto es evaluar dos protocolos de autenticación, los cuales deben cumplir con ciertos estándares como ser robustos frente a amenazas externas y ligeros para su implementación en etiquetas RFID pasivas. La primera parte del proyecto es la implementación de los protocolos en circuitos integrados de uso particular mediante el uso del lenguaje de programación VHDL, y el software Mod-elSim para la simulación de los circuitos. Ambos protocolos utilizan un algoritmo generador de números pseudoaleatorios, que proporciona a la información una mayor protección para la detección, principalmente de las etiquetas RFID. Una vez realizada la implementación de los protocolos, en la segunda parte del proyecto, los datos obtenidos se analizan principalmente datos relacionados con las LUT, rodajas, Flip-flops necesarios para la implementación de los protocolos y consumo de energía a través del Altera Quartus. II e ISE Xilinx Design Suitetool. Esto permite analizar y comparar cuál de los protocolos implementados se adapta mejor a las necesidades y principalmente a la limitación de recursos en las etiquetas RFID.

Palabras clave: RFID autenticación, protocolo.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatic identification is a technology that consist in obtaining information from an object in an efficient and fast way [1]. There are some identification technologies such as Voice Recognition, Fingerprint, Optic Character Recognition, Optical Strip, Magnetic Strip and Bar code. For many years, bar code is one of the most used automatic identification technology in supply chain management and inventory control systems, specifically in the process of scanning products and store management. [2] [3] [4] However, in recent years another technology appears into mainstream applications which identifies objects using radio frequency signals know as RFID (Radio Frequency Identification) [1]. Even though, it is not a new technology, RFID can be applied in new ways such as mentioned above and other areas like animal tracing, medical drug control, transportation, hospitals, banks and schools[3] [4]. RFID offers much advantages over traditional identification technology like bar code such as it does not need to be in line of sight with the label (cardboard or paper), it means that that the human assistance is not required and the data can read automatically at a distance of several meters.[2] [5]

Radio Frequency Technology uses electromagnetic fields to transfer and retrieve information by using devices called RFID tags and readers. [3] RFID tags store information similar to the barcode which contains unique identifier to a one object and can incorporate additional information such as product type, manufacturer, and even environmental factors such as temperature and humidity[5] [6]. Even though, this technology has many advantages within the RFID systems there are several security issues such as confidentiality, reliability, anonymity, integrity [6]. Therefore, the implementation of authentication protocols is necessary to solve the security threats.

1.1 Problem Statement

One of the many advantages of RFID technology is the greater data storage capacity on the RFID tag. However, due to the relative ease with which third parties could access, read or modify the data, users of this technology see an increase in the vulnerability of their information systems. To solve these problems, authentication protocols are used that allow a secure identification of the agents involved in the RFID communication (tag and reader), before starting the data exchange.

Chapter 2

Objectives

2.1 General Objective

Implement and analysis two secure and lightweight authentication protocols that fulfill the requirements of RFID technology.

2.2 Specific Objectives

- Review suitable authentication protocols to RFID technology.
- Design the digital circuits made to measure for the particular use of the authentication protocols.
- Develop simulations of a RFID system to verify the correct operation of the authentication protocols.
- Compare the performance of the authentication protocols implemented.

2.3 Contributions

- Programming of two RFID protocols in VHDL.
- Analysis of the results obtained from the programmed RFID protocols.

2.4 Manuscript Outline

This work is structured into the following chapters:

- **Chapter 1 - Introduction.** The introduction, the problem statement and the justifications of this work are presented.

- **Chapter 2 - Objectives.** The general objective and the specific objectives of this work are established.
- **Chapter 3 - Technical Background.** A technical background about RFID technology and applications are presented.
- **Chapter 4 - Related Work.** A review of recent work related to RFID authentication protocols is presented.
- **Chapter 5 - Methodology.** The procedure and tools that are necessary to implement the authentication protocols are shown. Also, the RFID authentication protocols are presented and explained.

  c
- **Chapter 6 - Results.** The results obtained from the simulation of the authentication protocols are described.
- **Chapter 7 - Conclusions** It finishes this thesis with the conclusions and directions for future work.

Chapter 3

Technical Background

3.1 RFID Technology

Radio Frequency Identification (RFID) is a wireless technology that uses radio waves to transfer and retrieve some identifying information from any object that includes a tag attached to it for automatic identification. [7] [8]

3.2 RFID Technology Architecture

An RFID system is basically composed of a RFID tag attached to an object and stores information that identifies it. In addition, an RFID reader is a device that captures specific information from RFID tag, and a back-end database that processes the information gathered by the reader[1] [9], as we can see in Figure 3.1.

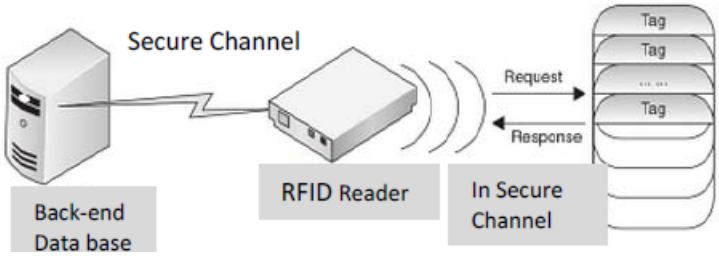

Figure 3.1: Radio Frequency Identification Model

.

3.2.1 RFID Tag

The fundamental RFID purpose for existing is to identify and recognize an item. The last is furnished with a chip known as RFID tag and contains an antenna coil and an integrated circuit that contains memory used to store data [10], as can be seen in Figure 3.2.

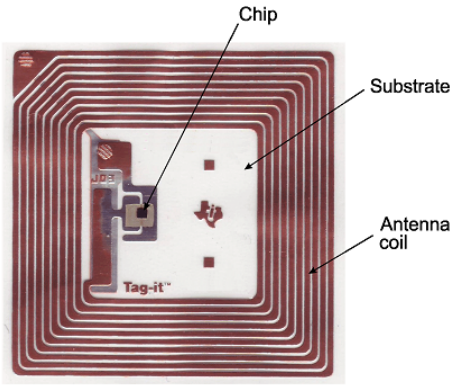

Figure 3.2: RFID tag

RFID tags can be classified according two criteria:

- **Power Supply Type**
  - **Active Tags:** Active RFID tags run with their own power source used to activate the microchip's circuitry and propagate their signal to the reader. An active tag has more storage capacity, and the transmission range is larger than passive tags. However, this type of tags is more expensive, and the lifetime depends on the battery[11] [12].
  - **Passive Tags:** Passive RFID tags achieve power from the signal of an external reader. It means a passive tag operates without battery[11]. The signal that comes from the readers induces a slight electrical current sufficient to work the integrated circuit of the tag so that it can generate and transmit a response. It has a short transmission range and storage capacity and it is considered the lowest cost than active tags and has a higher lifetime [12].
  - **Semi-Passive Tags:** This type of tags is a combination of characteristics of the two previous types. On one hand, it activates the chip circuit power by a battery (like active RFID tags) but on the other hand, it needs the energy to communicate with the reader [11] [12].
- **Memory Type** There are two kinds according to the type memory as read-only and read-write. Depending on the type of tag, it can be programmable, which means that the data can be modified or not. Read-only tags cannot be modified after is manufacture, therefore the data is static. Other disadvantages are that it can store a limited amount of data but a read-only tag is cheaper than read-write tags [2] [12].

  The classification above Figure 3.3 is part of the EPCglobal architectural framework and has become very popular in manufacturing and supply chain applications. [13]

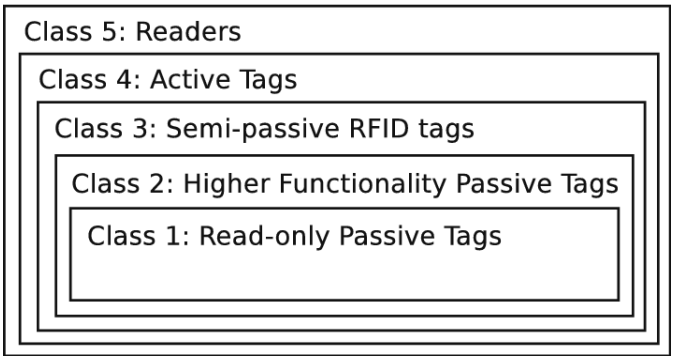

Figure 3.3: RFID tag classification

- **Class 1:** Passive tags with the only read/write memory function.
- **Class 2:** Passive tags with additional security functions.
- **Class 3:** Semi-passive tags. This type of tags has a battery that is used to power a portion of the circuit and provides a better read range.
- **Class 4:** These tags use a battery that power transmissions and operate the tag.
- **Class 5:** Readers area connected to back end database and have power to activate other tags

3.2.2 RFID Reader

In Figure 3.4 shows an example of RFID reader. A modern RFID reader is a scanner device that uses radio waves to communicate and read the tag information and send it to a back-end server [14]. Readers can be categorized by internal storage memory, processing capabilities, and operating frequency.

Figure 3.4: RFID reader

3.2.3 Back-end database

The last component of the RFID system identification is a data processing system. RFID passive tags have limits in storage and their information is principally an identification

number provider by the manufacturer due to this issue a back-end server is needed. The back-end server processes the information from the reader and related the tag's identification number with the information of the product stored in the database [12].

3.3 Operation of RFID system

In general, the RFID systems consist of three parts specified tag, reader, and back-end server (database) as shown in Fig 3.5. The function of an RFID system consists of a reader that emits an interrogation radio wave that requests all tag identification. Any tags within the reading range of the signal will send authentication messages to the reader. The reader forwards the tag's identification to the back-end server, which stores all information about tags and validates and confirms each tag , see Fig 3.5. The communication between the reader and back-end server can be considered in general secure, meanwhile, the channel between the reader and tags is considered insecure. The insecure channel can produce multiple threats like spoofing, the man in the middle, espionage, among others [4].

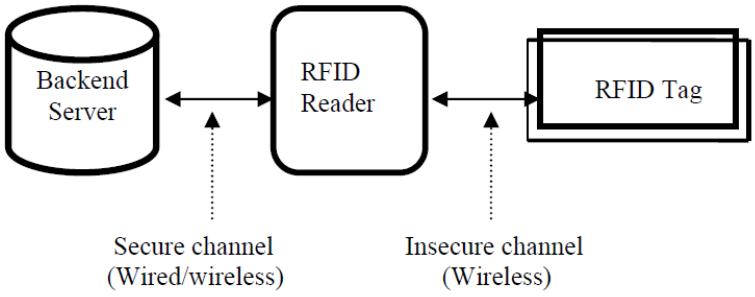

Figure 3.5: RFID system interaction

3.3.1 Communication methods

Communication between a passive or semi-passive tag and a reader consists of both power transfer and data transfer. The energy transfer is done by coupling through magnetic fields. RFID Tags operate either the magnetic field, the electric field, or both to receive energy from the reader [5].

- **Inductive Coupling**

  When a conductor is exposed to a magnetic field, it produces a current flow in it. This principle is utilized for communication in RFID systems. The reader's antenna generates an alternating magnetic field, which induces a current in the tag's antenna, which feeds the tag's circuits [5].

- **Backscatter Coupling**

  Since the magnetic field weakens with distance, the electric field is utilized to resonate the tag's antenna, in this way communication can be carried out at greater distances. This method of operation is typically used for systems that work in high

frequency (UHF or microwave), which allows the use of smaller, highly efficient antennas. The reader utilizes a system that allows it to separate the emitted signal from the signal [5].

3.3.2 Data Coding

1The coding and modulation of the signal accept the message to be transmitted and its representation in the form of a signal and optimally adapts it to the characteristics of the transmission channel. This process involves providing the message with a degree of protection against interference or collision and against intentional signal modifications, means in a reliable way. According to [1], RFID systems as show in Figure 3.6 can use NRZ (No Return to Zero) coding on the forward channel and Manchester coding in a backward channel.

1. NRZ (No Return to Zero): a binary '1' is represented by a "high" signal and a '0' for a "low" signal.

2. Manchester: a binary '1' is represented by a negative transition in the half the bit period and a '0' for a positive.

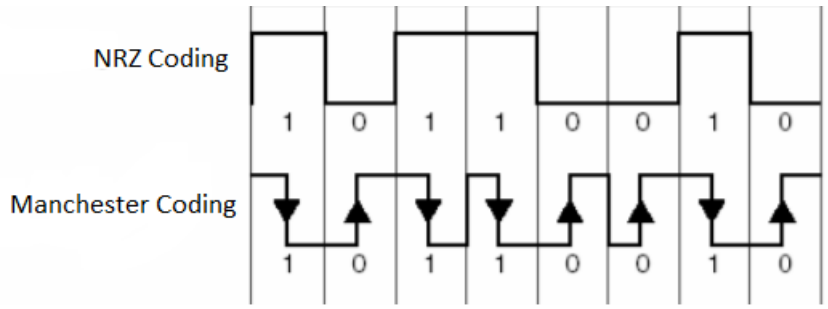

Figure 3.6: Codification schemes

3.3.3 Modulation signal

This process consists of modifying some of the variables of said signal in order to adapt it to the characteristics of both the communication channel and the agents involved in it. Typically, the variables modified are amplitude, frequency, or phase. In general, the RFID systems are preferred to use FSK and PSK [1]. In Figure, 3.7 can identify different types of modulations schemes and each scheme is defined below.

1. ASK Modulation (Amplitude Shift Keying): It is an amplitude modulation. Digital data are sent on an analog carrier. The binary '1' is represented by the presence of the carrier and the '0' by the absence of the carrier.

2. FSK Modulation (Frequency Shift Keying): It is frequency modulation. The two binary values are represented with two different frequencies, close to that of the carrier signal.

3. PSK Modulation (Phase Shift Keying): It is phase modulation. It consists to vary the phase of the carrier signal.

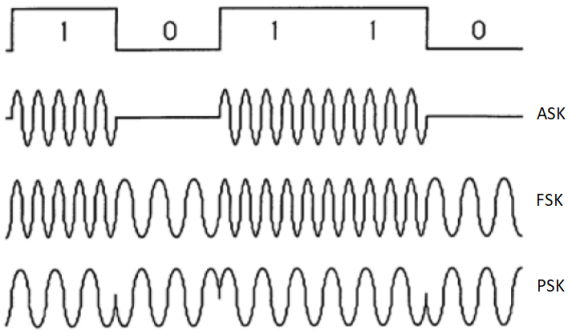

Figure 3.7: Modulation Representation

3.4 Frequency range

RFID Systems can operate in the three general bands of the electromagnetic spectrum [15] but this characteristic involves RFID tag reading range, the three bands mentioned below are:

1. Low Frequency: They are low speed, short distances and unit identification. Frequency range 125-134 KHz

2. High Frequency: Average speed and distances less than 2 meters. Frequency range 13.56 MHz.KHz

3. Ultra High Frequency: Massive high speed and long readings distances. Frequency range 860-960 MHZ.

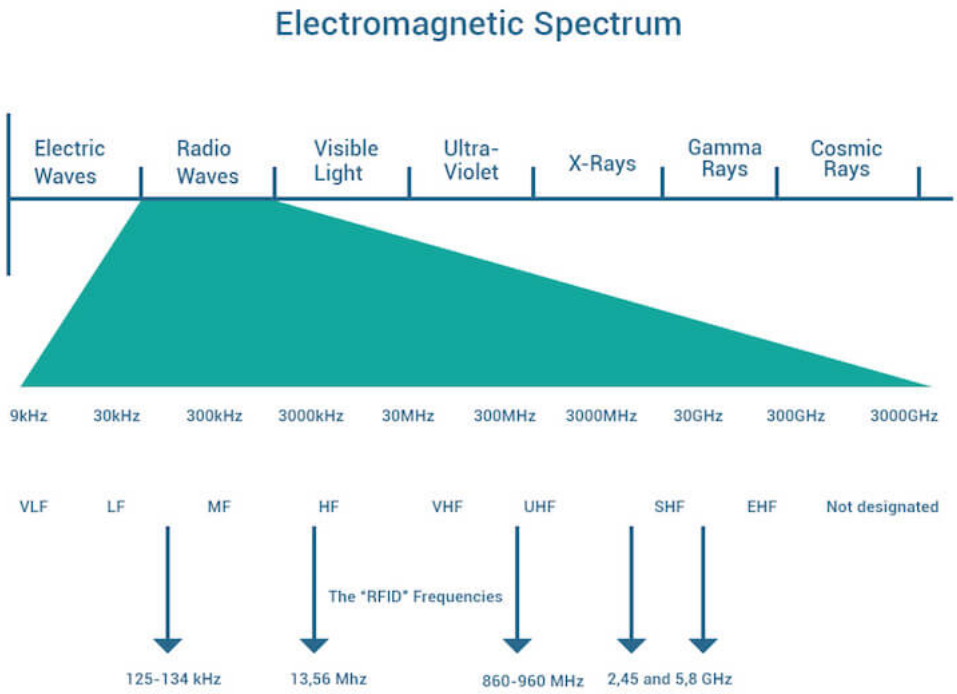

Figure 3.8: RFID Electromagnetic Spectrum

3.4.1 RFID authentication protocols classification

In [16] [17], authors divide the authentication protocols according to the complexity of operations of the protocol in four categories as follows:

1. Heavyweight or Full-fledged: These protocols support cryptography functions like hash function and public or private key encryption algorithms.

2. Middleweight or Simple: This type of protocol can support only hash function and pseudorandom number generators but no public key encryption algorithms.

3. Lightweight: This category of protocols can support functions like lightweight pseudo-random number generators (PNRG) and cyclic redundancy code (CRC).

4. Ultra-lightweight: This class of protocols involves tags that allow the incorporation of simple bit-wise logical functions like XOR, AND, OR, and mod 2^m.

3.4.2 EPC class-1 generation-2 standard

EPC Global is an organization in charge of registering all Electronic Product Codes (EPC), managing standards, as well as driving the global adoption and implementation of the EPC global Network in all sectors. EPC Global was created to establish and support the EPC network as a worldwide specification that will lead to an international standard (ISO) with the aim of achieving that the identification of any item within the

supply chain is immediate, automatic, and exact. Therefore, EPC Global has become the leading organization for the development of RFID specifications [18]. In Figure 3.9 shows the authentication scheme proposed by EPC global standard. The research of the EPC global organization has focused on the development of five essential elements for the EPC global network [19] [20].

- Electronic Product Code (EPC): It is a scheme for the universal identification of physical objects through RFID tags and other means. It consists of a manufacturer or administrator identification number, an object class identifier, and a serial number used to uniquely identify the identity of the object. [18]
- ID system: radio frequency readers and tags.
- Object Naming Service (ONS): It is a universal directory based on the Domain Name System (DNS). ONS does not contain the actual data about the EPC. It only contains the network address where the data resides.
- EPC Information Services (EPCIS): It is a standard for data exchange based on electronic product codes. It defines a schema for the data and interfaces for the capture and consultation of said data. [21]
- EPC middleware software (Savant): It is a software technology designed to manage and move information in a way that does not overload existing corporate and public networks, it acts as the middleware of the new EPC Global Network, managing the flow of information.

Figure 3.9 describes the authentication protocol by EPC class-1 generation-2 standard. The access password is a 32-bit value stored in the tag's reserved memory if this password is set, then data transfer will be established between tag and reader.

- Initially reader requests a random number from the tag.
- Tag generates a random number and sends to reader.
- The reader cover codes the password by performing a bitwise EX-OR between password and random number.
- The generated EX-OR output is transmits to the tag.
- The tag decodes the coded password by performing a bitwise EX-OR of the received cover-coded string with the original

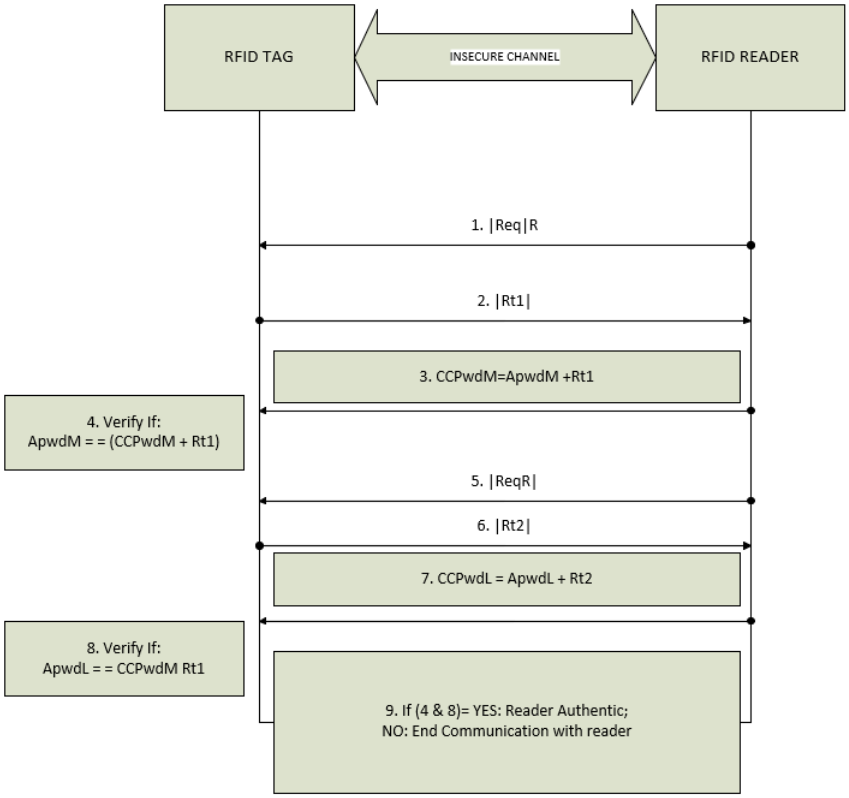

Figure 3.9: Authentication scheme proposed by EPC global standard

3.5 RFID Applications

Because of its versatility, RFID technology can be applied in a great number of sectors. Depending on the frequency, cost, and scope, the applications are different. Some of the areas in which RFID systems are used are as follows:

1. **Inventory and logistic.** It is currently the application to which RFID systems provide the greatest added value. The use of this technology allows any product to be located within the supply chain. Its application is becoming a fundamental technology for the development of global solutions for storage and distribution centers [22].

2. **Health care.** They are employed for the identification and location of assets, drug control, traceability and product temperature control [23]. It is equally applied for hospital management, establishing unique identification of the patient as well as their monitoring, both medical and physical, through the diverse areas of the center.

3. **Animals.** A microchip is used in a glass capsule that is inserted under the skin of the animal. It serves to identify it and store relevant information, such as who is its owner, vaccinations, age, etc. [22]

4. **Libraries and archives.** Bookstore management is an area where this technology is becoming increasingly popular. As this application was in several countries at the same time, it is not known exactly where it was first used.

RFID offers limitless possibilities for current and future use by providing an inexpensive, efficient, and reliable way to collect and store data. Figure 3.10 highlights just a few of the myriad uses of RFID technology.

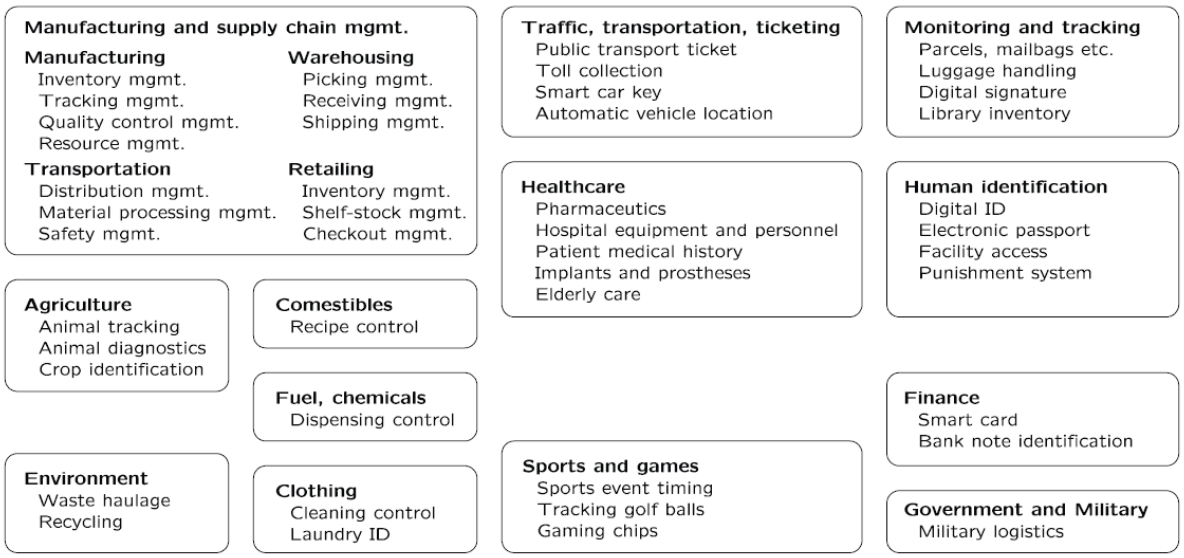

Figure 3.10: RFID application examples

3.6 RFID Advantages

Currently, the reduction of costs that are taking place in RFID devices, is positioning them as the future form of unitary and unambiguous identification of products. This makes it a reasonable substitute for the barcode, because of the various advantages it has over it:

1. Tags do not need to be in direct vision with the reader who identifies them.
2. Tags can identify each product individually, while barcodes uniquely identify the type of product.
3. Allows you to store much more information.
4. RFID tags can be read and write, while barcodes can only be read.
5. The information stored is encrypted using cryptographic and authentication systems.
6. They are more resistant and durable.

Barcodes and RFID devices automate the data collection process. However, the two technologies also differ significantly in many areas. This comparison can be seen in Table 3.1.

	RFID passive	**RFID active**	**Barcode**
Data security	Low to high	High	Minimum
Data modification	Modificable	Modificable	No Modificable
Storage	64 KB	8 MB	8 to 30 characters
Standards	Implementation phase	Owner and open standards	Stable and implanted
Time of life	Undefined	3-5 year battery life	Low for impairment
Reading distance	1 m	100 m	Few centimeters
Cost	Half	High	Low

Table 3.1: RFID-Barcode Comparative

3.7 RFID Disadvantages

However, RFID technology presents several problems and challenges that they will have to face in the coming years, such as security, implementation, cost, and others.

1. Privacy concerns are emerging as one of the biggest threats to RFID's rampant success. Today, RFID protocols are designed to deliver the most optimal performance between readers and tags, without addressing consumer privacy concerns.

2. Reader collision occurs when signals from two or more readers overlap. The concurrent query tag cannot respond and systems must be carefully configured to avoid this problem; Many systems use an anti-collision protocol (also called a simulation protocol. Anti-collision protocols allow tags to take turns transmitting to a reader).

3. When reading multiple tags at the same movement, it is possible that some tags will not be read and there is no sure method of determining this when the objects are not in sight. This problem will not occur with barcodes, for this when the barcode is scanned, it is instantly verified when read by a beep from the scanner and the data can then be entered manually if it does not scan.

3.7.1 FPGA

It is a programmable device that contains blocks of logic whose interconnection and functionality can be configured at the moment, using a specialized description language [24]. Their main characteristic and advantage is that they can be reprogrammed for a specific job or change their requirements after they have been manufactured. In Figure 3.11, we can see an FPGA example and it can be used for the implementation and

simulation of RFID protocols. Although, in this study a physical FPGA was not used, the authentication protocols were simulated in a virtual FPGA.

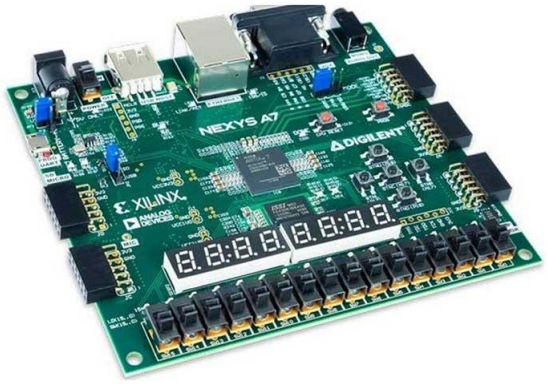

Figure 3.11: FPGA example

3.7.2 Finite States Machine

A finite state machine (FSM)1 consists of a set of states s_u and a set of transitions between pairs of states s_i,s_j . A transition is labeled condition/action: a condition that causes the transition to be taken and an action that is performed when the transition is taken. [25] In RFID authentication protocols, finite state machines are an essential part since they determine the flow of data that the RFID tag receives and the data it sends to the reader.

Chapter 4

Related Work

This chapter presents a review of the works related to RFID Systems. Radiofrequency identification systems have become popular in recent years and have been implemented in several areas. This type of system in the future will replace other systems like the barcode since they present more benefits. Furthermore, in the semiconductor industry, the drive to exploit ever-increasing miniaturization has led to a new trend towards installing large-scale electronic systems on a single chip. The proliferation of such "systems on a chip" has important implications for the structure of the electronics industry and the strategies of electronics companies. [26] Therefore, it has allowed the technological advancement of RFID technology and the implementation of better digital circuits in RFID tags. However, they present some security problems that affect the confidentiality and integrity of the information stored in the RFID tags. Various researchers have proposed various authentication protocols to solve these threats implementing different technique as pseudo-random number generators, encryption algorithms bit XOR, AND, OR operation, elliptic curve cryptography (ECC) encryption and others.

In [15], the author proposes a protocol, which each RFID tag has an EPC, password, and common architecture (truncated multiplier function) that is used to encrypt data from tag to reader and vice versa. They conclude the proposed protocol solves security threats in the transmission of tag's information. In addition, they compare their protocol with others based on parameters like power and logic elements. Likewise, Ohms [1] proposes a security layer for RFID systems focus on RFID active tags due to this category of tags have more risk of an intruder manipulating than passive tags since the reading range is larger in comparison with passive tags. The author mentions security problems can solve implementing a security layer. This security layer composes of four parts: air interface, data frame, authentication protocol and encryption algorithm. The author concludes the security problems can be neutralized, however, the implementation of this scheme demands an increase in power and cost of RFID active tags. In [27], the author proposes an ultralightweight RFID authentication protocol that provides strong authentication and integrity protection in a transmission. According to the presented results, the protocol requires only simple bit operations on the tag and can resist all possible attacks. In [28], the author describes a simple technique for inexpensive and untraceable identification of RFID tags. According to published results, the protocol requires minimal interaction between a tag and a reader. In addition, the protocol requires

a reduced computational load on the tag

(a single hash with a key). It also imposes a low computational load on the back-end server. In [29], the author proposes an authentication protocol that provides the identified privacy and security features. According to published results, the protocol resists leakage of tag information, tag location tracking, replay attacks, denial of service attacks, backward traceability, forward traceability (under a guess), and server impersonation (also under a guess). In [30], the authors demonstrate the limitations of an authentication protocol called RSEAP. In fact, they show that the vulnerabilities you suffer from. In addition, the authors present an authentication protocol called RSEAP2. This revamped version is more efficient (computing and communication costs) than the original RSEAP, while providing a more elevated level of security.

Because of the rapid development of wireless technologies and cloud computing, e-health based on the Internet of Things (IoT) represents a new field of research. In this context, an RFID authentication scheme based on the elliptic curve cryptography (ECC) encryption mechanism and the elliptic curve digital signature with message retrieval (ECDSMR) is presented in [31]. Analysis of the results shows that the proposed protocol provides many security features with a significantly low computational and storage cost. Likewise, in [32], the authors discuss the security of a hash-based protocol and a Rabin public key-based protocol. The authors show significant security problems in these schemes. In addition, they propose an improved version of Rabin's public key-based protocol to provide secure authentication between the tag and the reader. In [8] proposed two additional stages to any authentication protocol which improve the security information without real values (secret key and identifier). The back-end server generates a new secret key and an identifier that will be stored in tags. The author analyzes various threats with the scheme proposed.

In [33] the authors discuss security threats faced by lightweight RFID protocols, mainly privacy and integrity attacks that compromise card security. Also in the article they propose a lightweight authentication protocol with a single cryptographic mechanism, which is a synchronized random number generator and is shared with the back-end server.

Chapter 5

Methodology

This chapter describes the tools used for the implementation of the authentication protocols. Also, the authentication protocols are detailed along with the description of its implementation and the experiments carried out to evaluate the designs.

5.1 Tools

The programs that are used for the implementation of the authentication protocols are the following: ModelSim of Menthor Graphics, Xilinx ISE Design Suite and Altera Quartus II. Below, we present a brief description of each one.

- **ModelSim** It is a tool dedicated to the simulation and verification of digital circuits for VHDL, SystemVerilog and mixed languages design [34].
- **Xilinx ISE Design Suite** It is a software tool from Xilinx for synthesis and analysis of HDL designs, which primarily targets development of embedded firmware for Xilinx FPGA and CPLD integrated circuit (IC) product families [35].
- **Altera Quartus II** It is a software tool produced by Altera for the analysis and synthesis of designs made in HDL. It allows the developer to compile their designs, perform temporal analysis, examine RTL diagrams and configure the target device with the programmer [36].

5.2 Programming Languages

The programming language used in the development of the authentication protocols is VHDL. VHDL is an acronym for Very high speed integrated circuit (VHSIC) Hardware Description Language which is a programming language that describes a logic circuit by function, data flow behavior, and/or structure. This hardware description is used to configure a programmable logic device (PLD), such as a field-programmable gate array (FPGA), with a custom logic design.

5.3 Procedure

The development of the project consists of two main stages for the analysis of authentication protocols based on generated pseudorandom numbers. In the first place, the design of the protocols in the hardware description language is carried out, then the simulations of the protocols are developed with the ModelSim tool until the proper operation of the protocols is obtained. In the second phase, the protocols are evaluated with the Altera Quartus tool. The limitations of passive RFID cards are analyzed, mainly the area and energy consumption, to decide which protocol best suits the needs and limitations of this technology. Figure 5.1 presents the flow diagram used for the implementation of the authentication protocols.

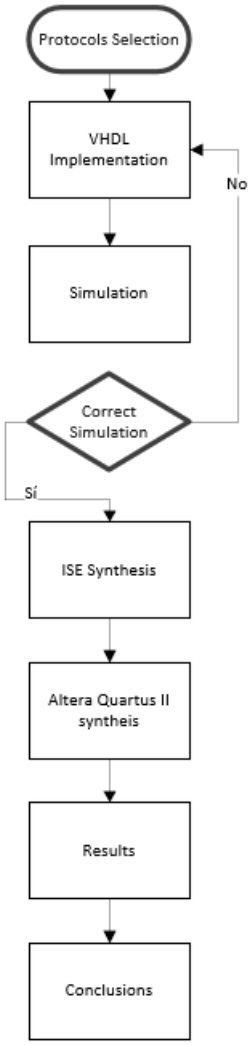

Figure 5.1: Flow diagram of the implementation of the protocols.

5.4 First RFID Authentication Protocol

The first authentication protocol (Flyweight RFID Authentication Protocol) [33] implemented aims to fulfill the following requirements: adapt to EPCGen2 standards, be robust against external threats and be light enough to allow its implementation in the limited resources that tags present Passive RFID.

5.4.1 Protocol Description

It is based on the basis that each tag (T) shares with the server (S) a synchronized random number generator (RNG), that is, the same algorithm, the same key, and the same seeds. T and S authenticate each other by exchanging between 3 and 5 numbers. The security of the protocol is based on the fact that the numbers created by the random number generator cannot be predetermined by the adversary and that T and S are synchronized at all times. The protocol supports mutual authentication, resistance to replay attacks, as well as forward and backward security.

Each T tag stores the following data in non-volatile memory:

- Two pseudo-random numbers: RN_1 and RN_2.
- Tag Identifier: ID_{tag}.
- Current state: g_{tag}.
- A flag (1bit) that is activated when communication begins: *cnt*

On the other hand, the server stores in its database for each tag:

- Six pseudo-random numbers: RN_1 cur, RN_{1next}, RN_2, RN_3, RN_4, RN_5.
- Tag identifier: ID_{tag}.
- Current state of the tag: g_{tag}.
- A flag (1bit) is activated when the communication begins: cnt'.

The database list is doubly indexed, using the current value of the first pseudo-random number RN_{1cur} and its next value RN_{1next}, so that the corresponding tag can be located using both numbers. To initialize the tag variables, two successive random numbers are generated which will be RN_1 and RN_2 and the cnt flag is set to zero. The server will do the same, setting cnt' to zero and generating six successive random numbers that will be assigned to the corresponding variables. Although in this work the part of the protocol corresponding to the tag will be developed (since it is the one with resource limitations) the protocol for both parts, tag and server is shown below.

Protocol

1. **The reader (R) sends the Query request signal to tag (T).**
    - Then, the tag assigns the value of the variable cnt to the alarm signal, which initially is 0. Later, cnt is updated to 1. The tag sends RN_1 to the reader.

2. **The reader receives RN_1 from the tag and compares it with the value stored in the database.**

   - If $RN_1 = RN_{1cur}$ for a tag in the database, then the reader assigns the value of the variable cnt' to the alarm signal, subsequently setting cnt' to 1. The reader sends RN2 to tag.
   - If $RN_1 = RN_{1next}$ for a tag in the database, then the alarm' signal is set to 0, the data values are updated and RN_2 is sent to tag.
   - If not, the process is aborted.

3. **The tag receives RN_2 from the reader and compares it with the value stored.**

   - If RN_2 is correct then it generates five successive numbers, they are assigned to variables and cnt is set to 0.
     - If alarm = 0 then RN_3 is sent.
     - If not, RN_4 is sent.
   - If the received RN_2 value does not match the value stored for this number in tag, the process is aborted.

4. **The reader receives RN_i from tag and compares it with the value stored in the database. Where 'i' indicates the number sent in each case.**

   - If $RN_i = RN_3$ and alarm'= 0 then update the data values and accept the tag as an authorized tag.
   - If $RN_i = RN_4$ then send RN3 and update the data values.
   - If not, the process is aborted.

5. **The tag receives RN_3 from the reader and compares it with the value stored.**

   - If valid, RN_5 is sent.
   - If not, the process is aborted.

6. **The reader receives RN_5 from the tag and compares it with the value stored.**

   - If it is valid then accept the tag as an authorized tag.
   - If not, the process is aborted.

5.4.2 Protocol Implementation

The architecture to implement this protocol is shown in Figure 5.2 and it consists of the following elements.

1. Registers store the variables RN_1, RN_2 and RN_3. The rest of the RN_4 and RN_5 values do not need to be stored since as they are generated they will be sent to the reader.

2. A random number generator.

3. A timer establishes the time the tag waits for a reply signal before aborting communication.

4. A state machine that controls the flow of the protocol.

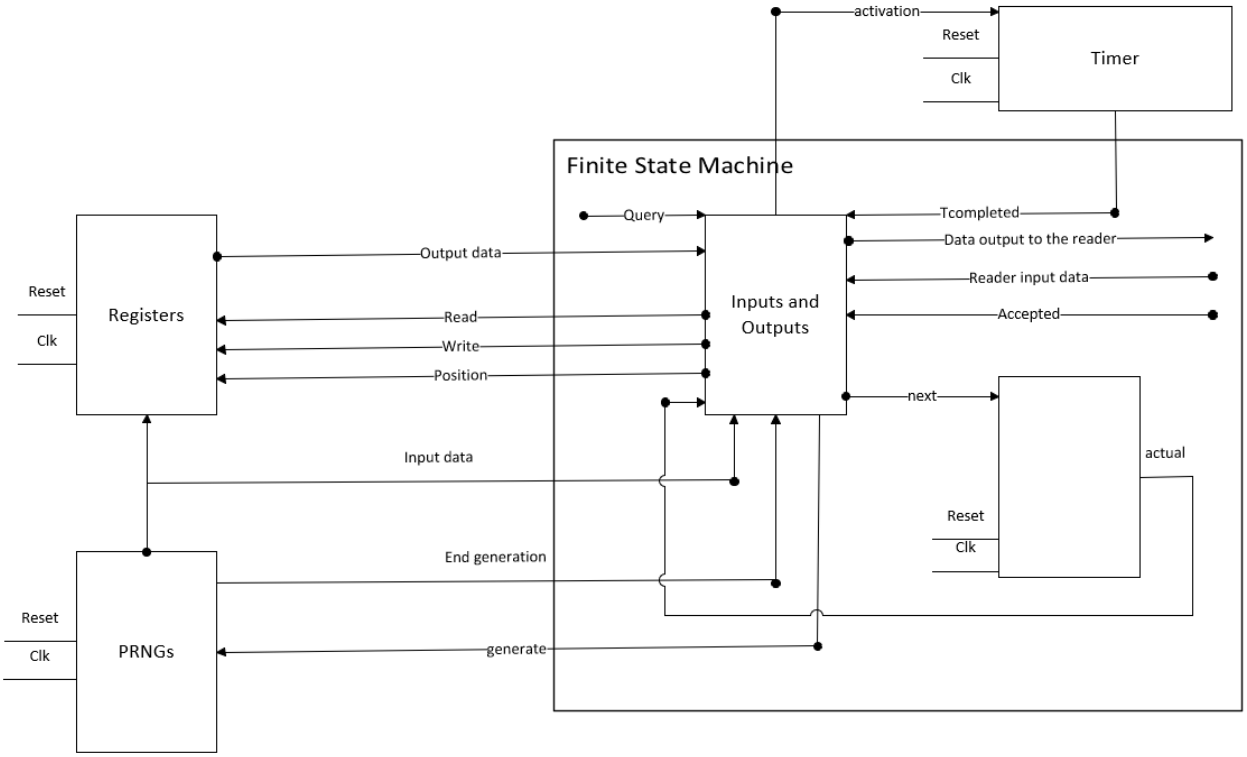

Figure 5.2: First authentication protocol architecture

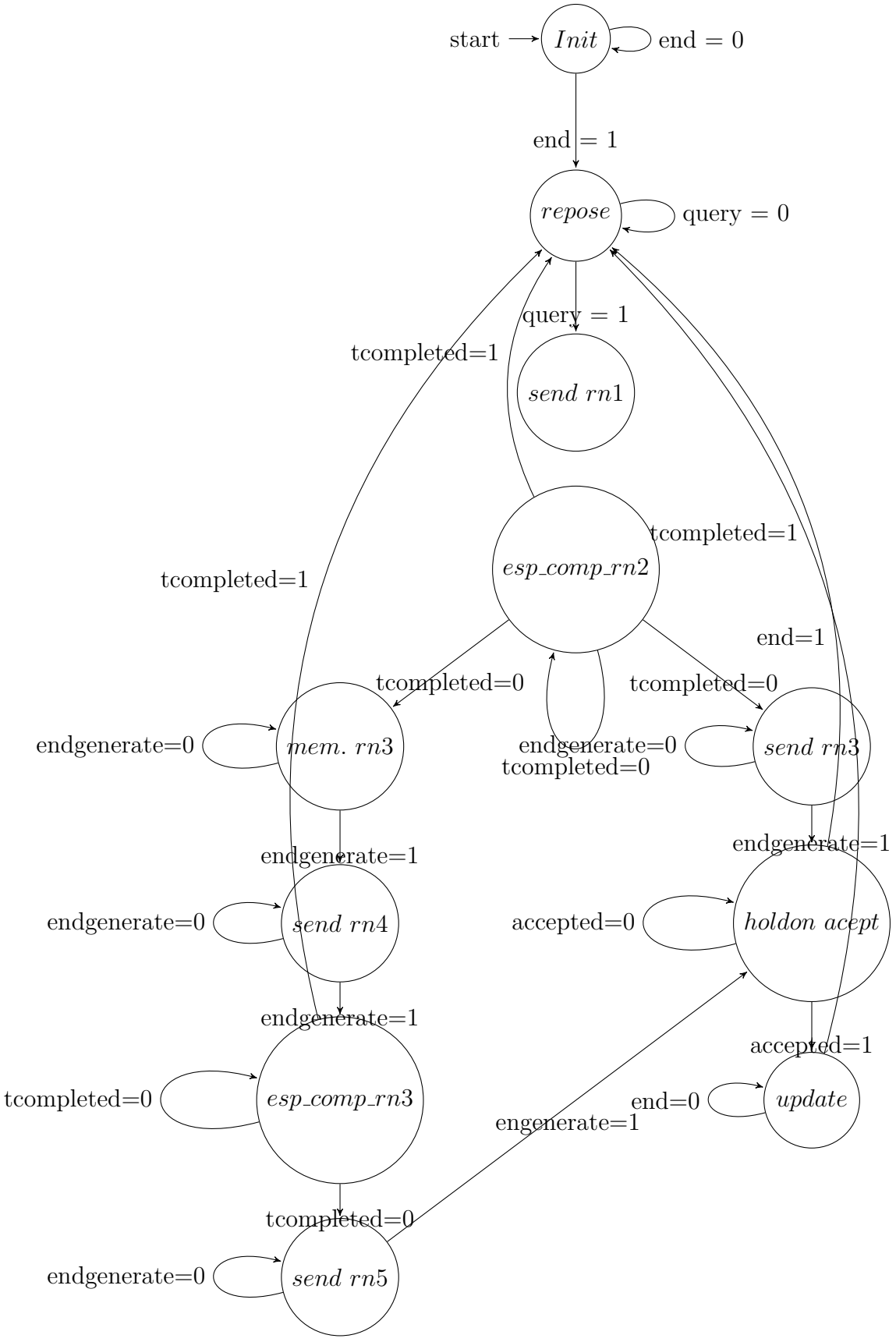

Figure 5.3: Description of the FSM

Figure 5.3 shows the state diagram of a finite state machine (FSM). It has been necessary to implement the operation of the authentication protocol. It consists of the following states. It starts from an initialization state in which values are assigned to RN_1 and RN_2, then going to the idle state, waiting for a communication request from the reader. Once this happens, the tag sends RN_1 to the reader, waiting to receive its reply, RN_2. If it is received before the th is reached, it will be compared with the value that the tag has stored. In the event that said comparison is affirmative, it must be assessed whether the alarm is activated or not, which will indicate whether or not any incident has occurred during the communication.

The time completed variables are indicators that the tag has not received a response from the reader and depending on the value of this variable, it will advance to the next state or return to previous states. Other variables to take into account is the end generate, which will indicate if the pseudo-random number generator can generate the numbers RN3, RN4 or RN5.

5.5 Second RFID Authentication protocol

The second authentication protocol, Chen and Deng protocol, [37] implemented aims to fulfill the following requirements: adapt to EPCGen2 standards, be robust against external threats and be light enough to allow its implementation in the limited resources that tags present (Passive RFID).

5.5.1 Protocol Description

1. The reader sends a message to tag, at that moment the authentication protocol begins. The data sent will be M_{req} and RND_{reader}

2. Once the message is received, the tag will generate two random numbers (RND_t, RND'_t)

3. Through XOR operations and the pseudo-random number generator, the following operations will be performed to obtain A, B, C.

   - A= PRNG (N_t XOR RND_t XOR RND XOR ID16)
   - B=PRNG ($EPC16_t$ XOR RND'_t)
   - C= RND'_t XOR K16

4. For the calculation of A, we have N_T that will be twice as many bits as the other variables, the XOR of the lower part of N_T will be performed first, to later concatenate it with the upper part of N_T and introduce it in the PRNG.

   - A=PRNG(N_t(high) N_t(low) XOR RNDt XOR RND XOR ID16)

5. Once these variables have been calculated, the tag will send the reader the following data RND_t A B C

6. Once the tag data is received, the reader performs two operations:

- Identification of the tag: In case there were several tags to which you want to access.
- Tag authentication: The reader will obtain its own A value and will compare it with the value sent by the tag, in the event that the comparison of different values the protocol is aborted. If the two values are equal the reader will calculate the next value D.
  - D= PRNG(N_t XOR $RND_{lreader}$ XOR RND_t XOR EPC16 XOR RND_t)
- Updating of the $N16$ and $K16$ data stored by the reader.

7. Reader sends D to tag.
8. Once the tag receives D from the reader, it calculates its own D value and will compare it with the value sent from the reader
   - D = PRNG (N_t XOR RND_{lector} XOR RND_t XOR EPC16 XOR RND_t). In this case, as for the calculation of A, N_t is twice as much as the other variables, therefore, the low part will be performed first and then concatenated with the high part.
   - D= PRNG(N_t(high) N_t(low) XOR RND_{lector} XOR RND_t XOR EPC16 XOR RND_t)
9. If $D_{reader} = D_{tag}$, the authentication process is completed.
10. Finally, the tag updates the following variables.
    - N16 = PRNG (N16 XOR RND')
    - K16 = PRNG (K16 XOR RND')

Communication between tag and reader will be done through an N bit channel for which the different data will be represented as follows in Table 5.1.

Variable	Value
N_t	32 bit tag password
N16	This value is obtained by xor function to the two blocks into which Nt is divided
K_t	Password kill of the 32 bit tag
K16	This value is obtained by xor function to the two blocks into which Kt is divided
EPC	96 bit tag EPC identification
EPC16	Apply XOR function to the 6 blocks of the EPC gives EPC16
ID16	16 bit reader identification number
PRNG()	16 bit pseudo-random number generator
RND	16 bit random number
RND´	16 bit secondary random number
XOR	XOR function
M_{req}	Message requested by the reader

Table 5.1: Protocol Variable

5.5.2 Protocol Implementation

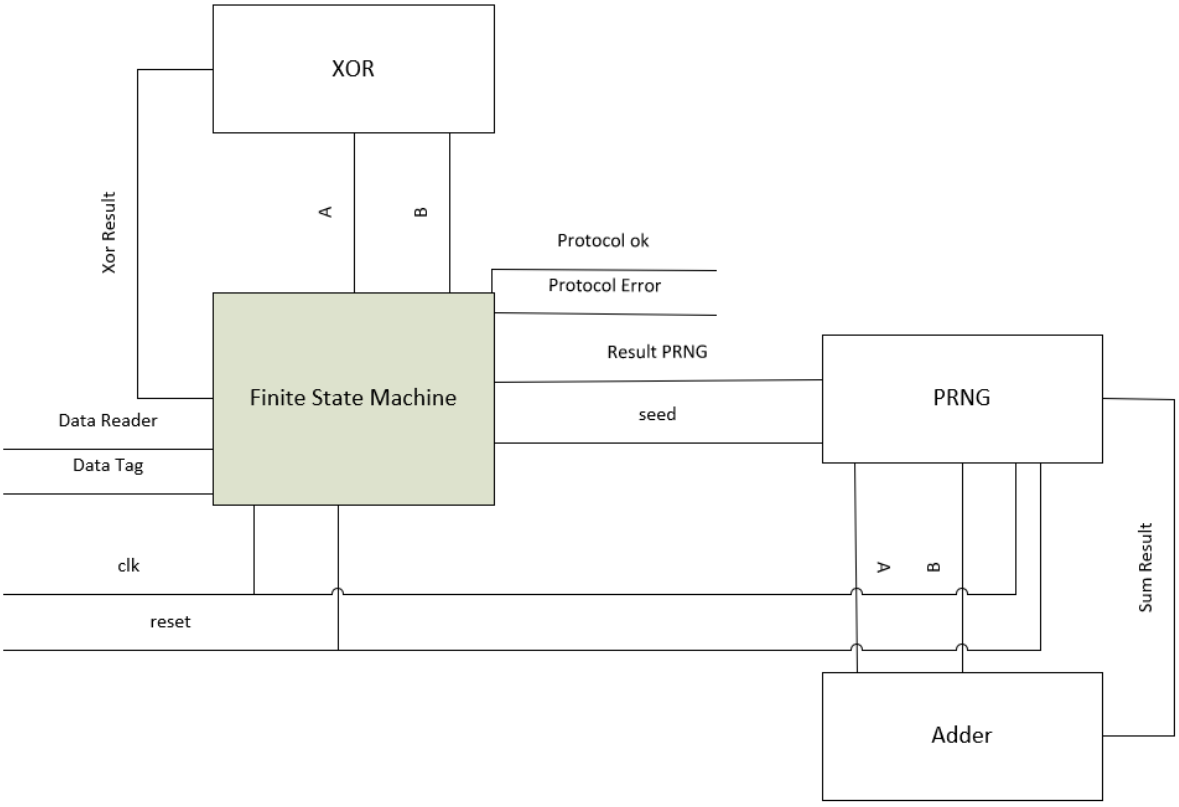

Figure 5.4: Second Protocol Architecture

The Figure 5.4 shows the architecture of the second authentication protocol and it consists of the following elements:

- Finite state machine. This block controls the flow of the input and outputs that the protocol use in the authentication process.
- Pseudo Random number generator.
- XOR block. This logic element performs a xor operations with pseudo random number generator to development A, B and C variables.
- Adder. This logic element carries out the function of concatenating the previous variables with the values of the pseudo-random generator to complete the variable D and verify the authentication of each tag

5.6 Random Number Generators

A random number generator is a mathematical algorithm that produces an indefinite sequence of apparently random numbers, that is, they satisfy the randomness tests. Most of the random number generators are, in reality, pseudo-random, since they require a seed (X0), from which to generate the desired sequence of numbers X1, X2, X3, ... If you start from the same seed will always obtain the same sequence. In this project, different generators [38] have been used in the implementation of the protocol to analyze the most efficient solution. These generators are described below.

5.6.1 Generator A

Option A

This algorithm is an ultralightweight PRNG, based on the use of a triangular function. It starts from two initial values called seeds, which will be the input variables (x_0 and x_1), see Figure 5.5. The constant x_0 will be the initial value of z with which a series of 64 iterations will start consisting of the total sum of z, the constant x_1, a shift of z to the right and a shift of z to the left. Finally, the output is filtered to use the number of significant bits required.

```
Opction A
x[0] = x[0] + ((x[0]*x[0])|5);
x[1] = x[1] + ((x[1]*x[1])|13);

//Non-linear function

z[0] = x[0];
for (i=0; i<64; i++)
        (z[0] = (z[0]>>1) + (z[0]<<1) + z[0] + x[1];)

//filter output
//z[0] is generator output
//I keep the least significant bits, which are the ones that vary
//the most significantly

z[0] = z[0] & 0x0000ffff;
```

Figure 5.5: Description of the prng (A algorithm) .

Option A1

This algorithm is a variation of the previous one. Starting from the same ideas, its implementation is based on the use of a half-bit adder that performs the same calculations as the generator of option A. The objective of this modification is to reduce the area used by the design.

5.6.2 Generator B

Option B

As in the case of algorithm A, it is also based on a triangular function. It starts from two initial values, called seeds (x_0 and x_1). It consists of two functions (A and B), each of which is performed for 24 iterations to obtain cryptographic secure pseudo-random numbers.

```
Option B

x[0] = x[0] + ((x[0]*x[0])|5);
x[1] = x[1] + ((x[1]*x[1])|13);

//Non-linear function

r1 = 24;

/* A - Function */
z[0] = x[0]^x[1];
for (i=0;i<r1+){z[0]=(z[0]<<1) + ((z[0] + (0x56AB0A))>>1);}


/* B - Function */
y[0] = x[1]^z[0];
for (i=0;i<r1++){y[0]=(y[0]>>1) + (y[0]<<1) + y[0] + (0x72A4FB);}

z[0] = z[0]^y[0];
//Filter output
//z[0] is generator output
//I keep the least significant bits, which are the ones that vary
//the most significantly


z[0] = z[0] & 0x0000ffff;
```

Figure 5.6: Description of the prng B algorithm.

In Figure 5.6, function A consists of assigning to z_0 the result of performing a bit-level XOR between x_0 and x_1. Then, the sum of a shift to the left of z plus a shift to the right of the sum of z with a number in hexadecimal would be carried out, repeating this the number of times indicated. Once the 24 iterations of function A are finished, it is passed to function B. The first thing to do will be to assign to y_0 the result of performing an XOR function at the bit level between x_1 and the value obtained from function A. carry out the sum of: a shift to the left of y, plus a shift to the right of y, plus y, plus a hexadecimal number, performing the relevant iterations. Finally, an XOR function will be carried out at the bit level between the value obtained from function A and the value obtained from function B. The number of significant bits required is selected from this result.

Option B1

This algorithm is a variation of the previous one. Starting from the same ideas, its implementation is based on the use of a half-bit adder that performs the same calculations as the generator of option B. The objective of this modification is to reduce the area used by the design.

Option B2

This algorithm is a variation of the previous B1. Its implementation is based on the use of a quarter-bit adder that performs the same calculations as the generator of option B1. Similarly, the objective of this modification is to reduce the area used by design.

Chapter 6

Results

In this chapter, the results of the analysis of the implemented protocols are shown. The results are obtained from the synthesis of our designs carried out in ISE Xilinx Design Suite and Altera Quartus II.

6.1 Performance Analysis

The analysis of the obtained results is performed in terms of number of LookUp Tables, Slices and Flip-Flops. This section describes the metrics used to test our implementations.

- **A LUT (LookUp Table)**. This truth table defines how our combinatorial logic behaves. The way FPGAs implement combinatorial logic is with LUTs, and when the FPGA is configured, it just populates the table's output values, which are called "LUT-Mask", and is physically made up of SRAM bits.
- **Slice**. Number of slices on an FPGA is a metric of the hardware resources that the FPGA has. A slice is made up of LUTs and flip flops.
- **Flip-Flop**. Flip-flops are storage registers used to store logical states in an FPGA
- **Logic elements**. These are the small electronic subsystems that perform the logic decisions of NOT, AND, OR, and so on, which are incorporated inside any piece of digital electronic equipment.
- **Propagation delay**. It is the amount of time it takes for the head of the signal to travel from the sender to the receiver.
- **Clock to output delay**. The maximum time required to obtain a valid output at an output pin after a clock transition at an input clock pin, directly or through a register.
- **PDP (Power delay product)**. It is a figure of merit correlated with the energy efficiency of a logic gate or logic family.

The synthesis of the authentication protocols is carried out using the ISE Xilinx Design Suite and Altera Quartus II tools. The data that will be observed once the design synthesis has been carried out are related to LookUp Table, slices and Flip-flops which were synthesized with the first tool (ISE Xilinx Design Suite). Likewise, the data logic elements, propagation delay, clock to output delay, critical path register to register delay, Static power dissipation are synthesized with the Altera Quartus II tool. The results of the synthesis of the first authentication protocol will be shown, using the two types of PRNGs (A and B) with their respective modifications (A1, B1 and B2).

6.1.1 First RFID Authentication Protocol

As can be seen in Table 6.1 and Figures 6.1 and 6.2, the number of LUTs of the first protocol, with the pseudo-random number generator A varies slightly with respect to the type A_1 generator since type of generator has more complexity than its predecessor. However, there is a great variation in the amount of LUTs with respect to the type B generator and its different modifications (B_1 and B_2). This is due to the fact that the type B generator is much more complex when implemented in VHDL.

	Pseudo-Random Number Generator				
Parameters	A	A1	B	B1	B2
LUT	434	507	671	821	827
Slices	232	266	356	436	447
Flip-flops	189	304	252	330	344

Table 6.1: Results of LUT, slices and flip-flops of the first protocol 32 bits

	Pseudo-Random Number Generator				
Parameters	A	A1	B	B1	B2
LUT	486	521	647	679	706
Slices	248	264	389	406	431
Flip-flops	207	256	289	357	378

Table 6.2: Results of LUT, slices and flip-flops of the first protocol 64 bits

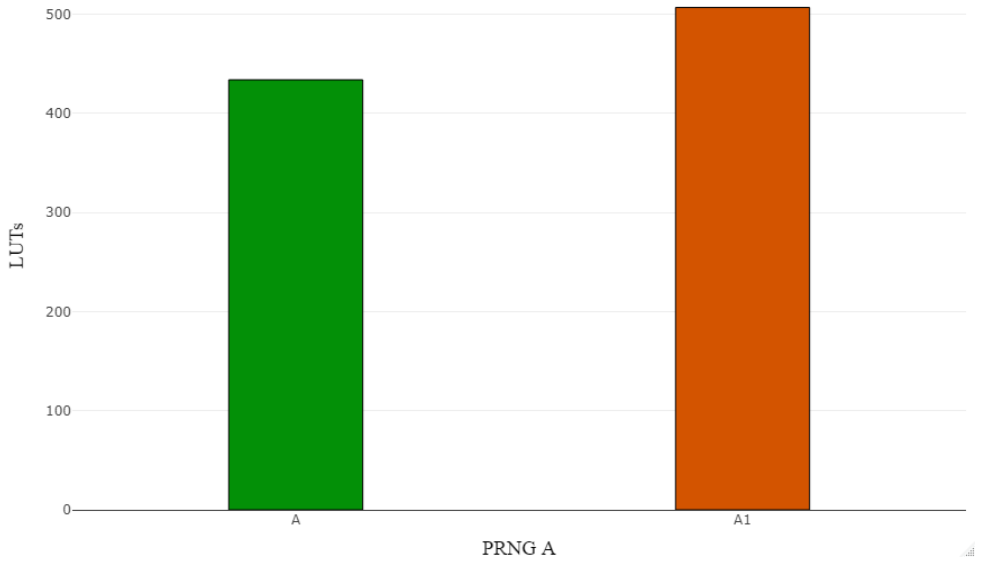

Figure 6.1: LUT comparison of the first protocol with PRNG A-A1

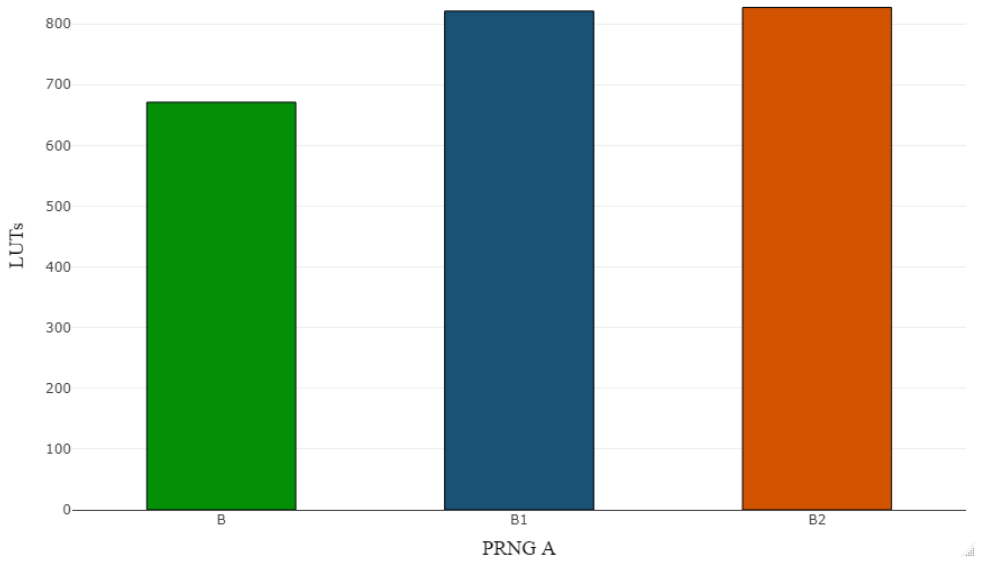

Figure 6.2: LUT comparison of the first protocol with PRNG B-B1-B2

Figures 6.3 and 6.4 show how the number of slices and the number of flip flops increases directly proportionally. This was to be expected given that a slice is made up of LUTs and flip flops. So, these three parameters give us a reference that the area increases as the protocol and the pseudo-random generators increase in complexity. We complement this information with the subsequent analysis of the number of logic elements.

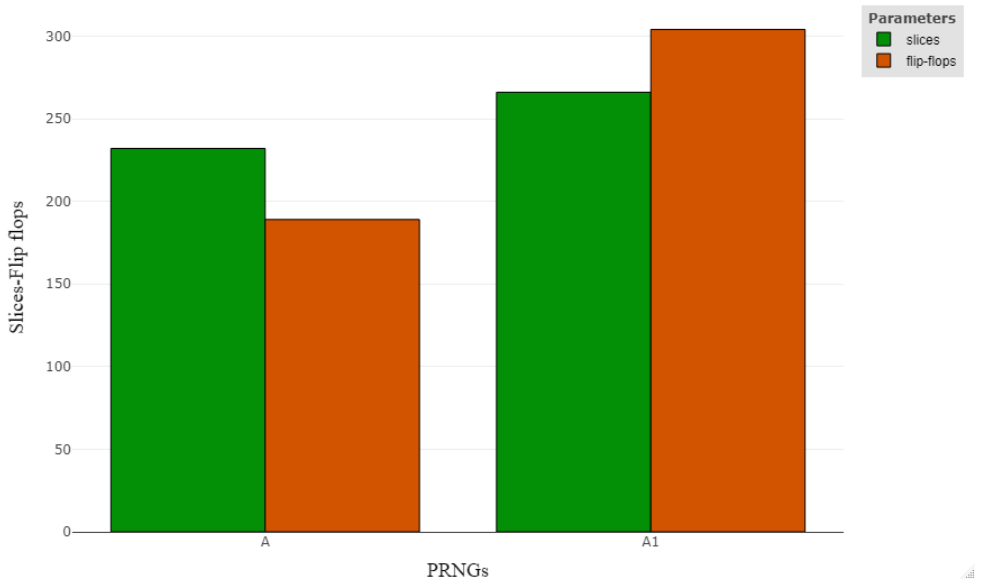

Figure 6.3: Slices-Flip flops comparison of the first protocol with PRNG A-A1

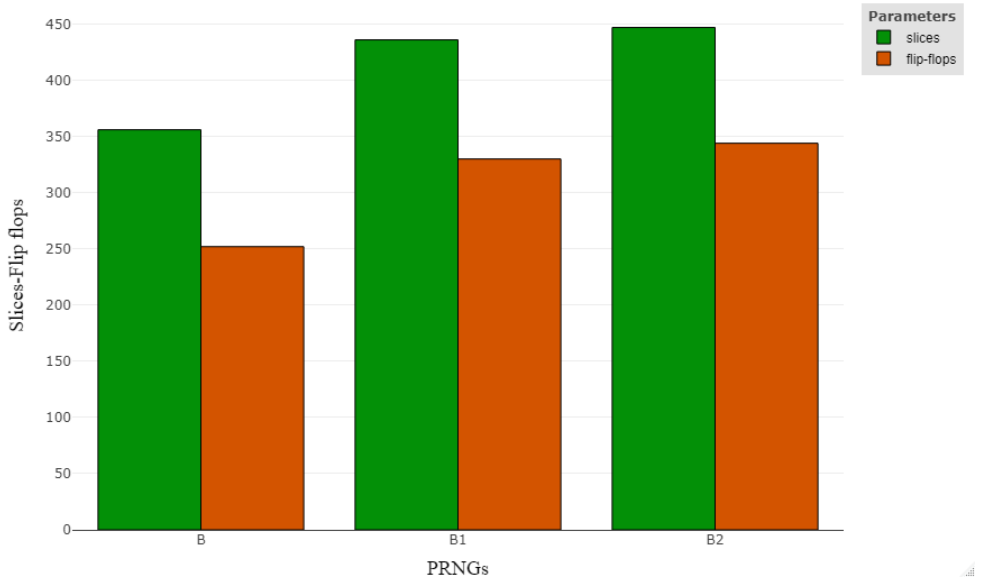

Figure 6.4: Slices-Flip flops comparison of the first protocol with PRNG B-B1-B2

As we can see in Table 6.3 and Figure 6.5, the number of logic elements increases using the type A generator and its modification A_1, likewise, this parameter increases when using the type B generator and its variants $B_1 - B_2$. This produces an increase in the area that the circuit occupies within the RFID tag, which complements the data obtained in Table 6.1. Finally, the response time of the transmission of information between the RFID tags and the reader increases slightly in the three parameters analyzed which are propagation delay, clock to output delay and critical path register to register delay.

	Pseudo-Random Number Generator				
Parameters	A	A1	B	B1	B2
Logic elements	534	546	575	603	635
Propagation delay (ns)	24.926	25.264	25.341	25.297	26.214
Clock to output delay (ns)	22.197	22.368	22.401	22.447	22.513
Critical path register to register delay (ns)	1.149	1.153	1.159	1.163	1.168

Table 6.3: Results of logic elements, propagation delay, clock output delay and critical path eof first protocol 32 bits

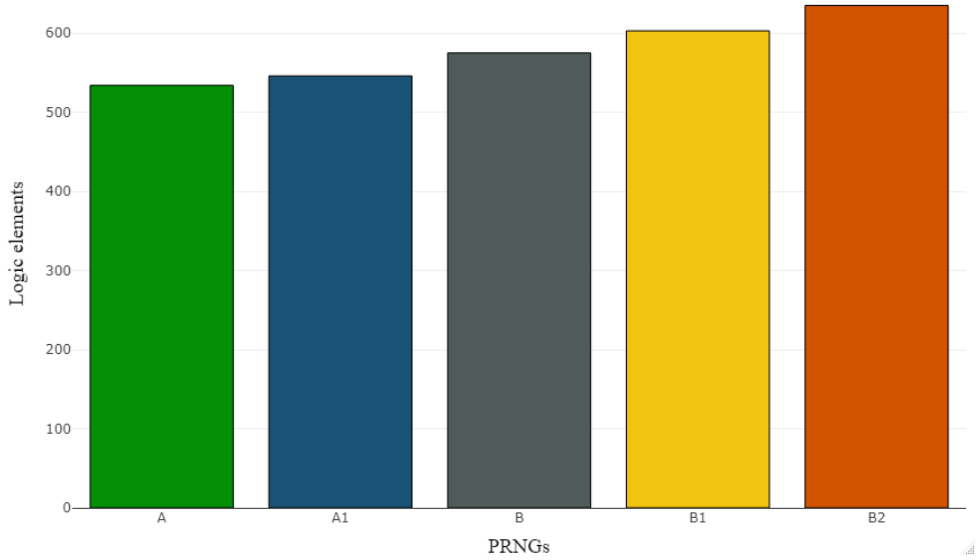

Figure 6.5: Logic elements of first protocol 32 bits

	Pseudo-Random Number Generator				
Parameters	A	A1	B	B1	B2
Logic elements	549	584	617	649	668
Propagation delay (ns)	25.347	25.873	26.238	26.492	26.572
Clock to output delay (ns)	22.384	22.406	22.467	22.564	22.681
Critical path register to register delay (ns)	1.249	1.347	1.376	1.418	1.453

Table 6.4: Comparison of power,delay and PDP estimation of first protocol 64 bits

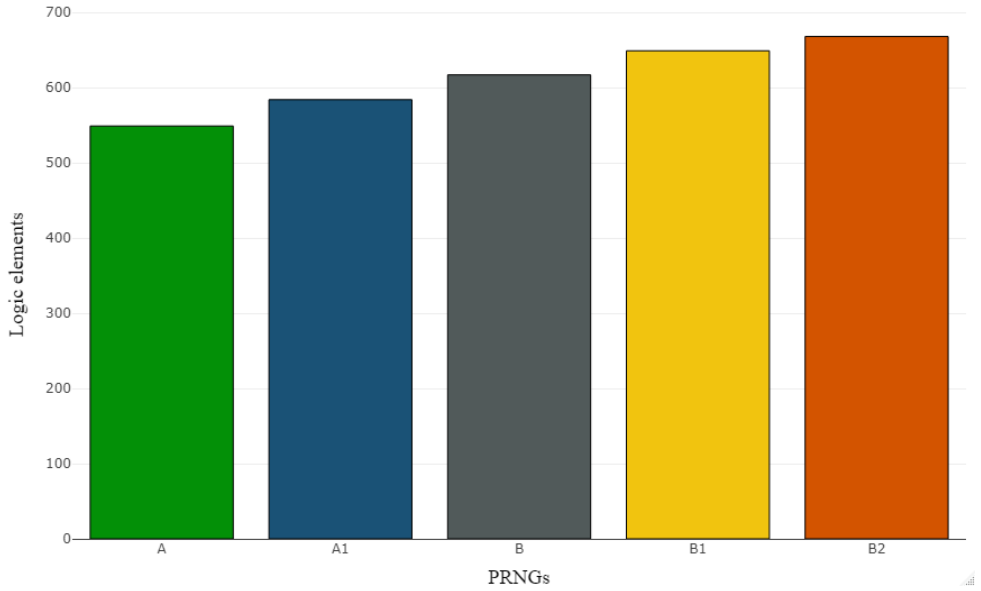

Figure 6.6: Logic elements of the first protocol 64 bits

Finally, Figure 6.6 shows how the number of logic elements is increased as in the first protocol with 32 bits, however, there is a significant difference with respect to the data obtained in Figure 6.5. The increase in the logic elements is due to the increase in the number of bits that are used in the exchange of information between the tag and the reader. In addition to the increase in the complexity of the authentication protocol algorithm and the type of pseudo-random generator that is used in the protocol, the response time in the transmission of information also increases.

6.1.2 Second RFID Authentication Protocol

Table 6.5, Table 6.6 and Figures 6.7 and 6.8 illustrate the number of LUTs of the second protocol, however, compared to the first protocol they are lower. This is because the pseudo-random generators type A and B are better coupled to the architecture of the second protocol.

	Pseudo-Random Number Generator				
Parameters	A	A1	B	B1	B2
LUT	417	503	682	805	837
Slices	217	260	371	442	459
Flip-flops	201	315	278	345	362

Table 6.5: Results of LUT, slices and flip-flops of the second protocol 32 bits

	Pseudo-Random Number Generator				
Parameters	A	A1	B	B1	B2
LUT	512	549	712	845	858
Slices	262	294	388	457	493
Flip-flops	246	337	359	371	394

Table 6.6: Results of LUT, slices and flip-flops of the second protocol 64 bits

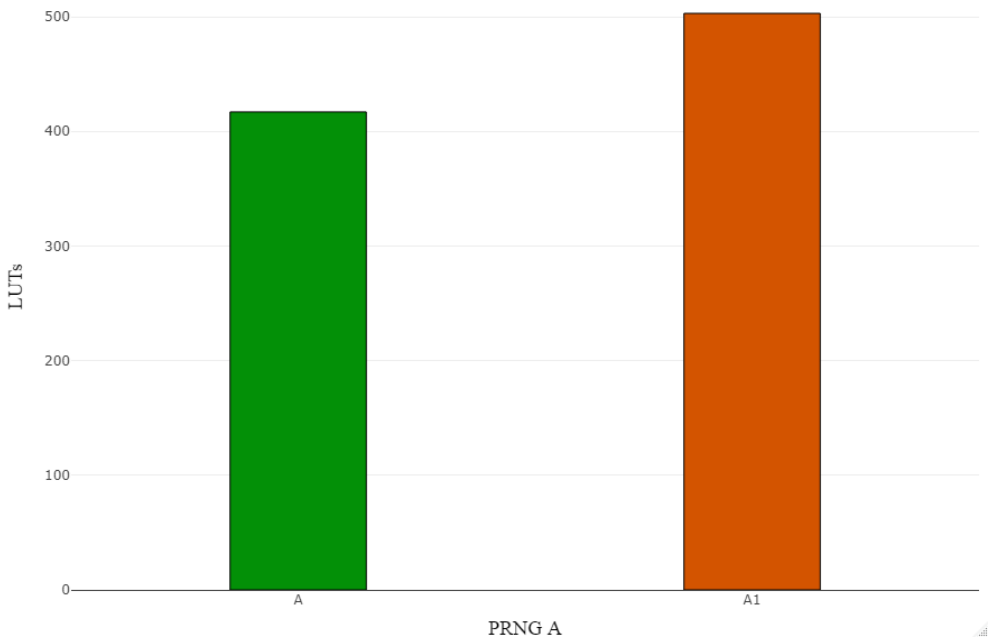

Figure 6.7: LUT comparison of the second protocol with PRNG A-A1

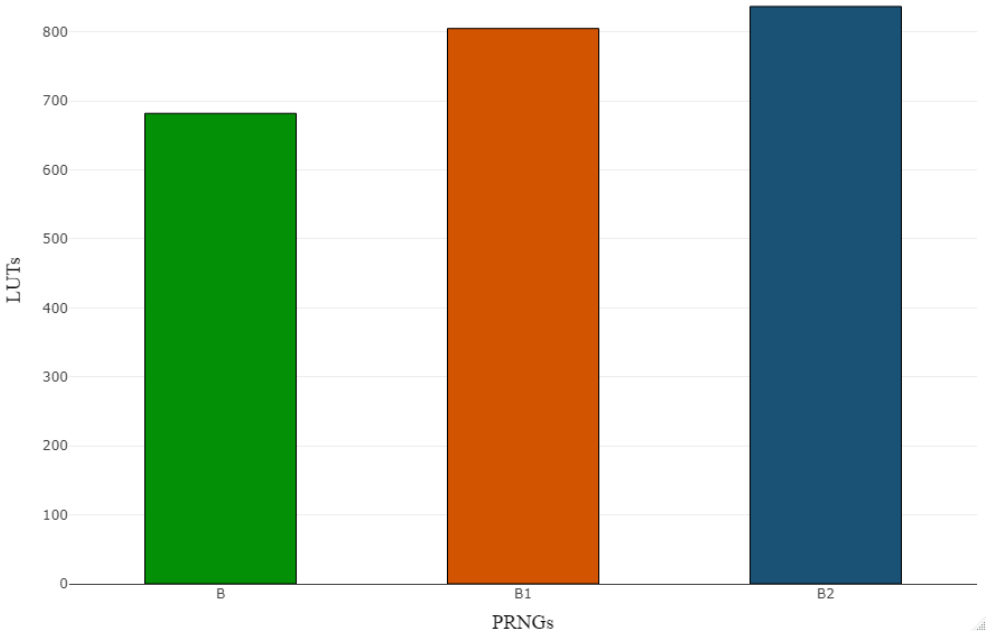

Figure 6.8: LUT comparison of the second protocol with PRNG B-B1-B2

Figures 6.9 and 6.10 depict how the number of slices increases directly proportional to the number of flip-flops. There is not a great variation with respect to the first protocol as can be seen in Figures 6.3 and 6.4. Therefore, by increasing the complexity of the protocol algorithm and the pseudo-random number generator, the area used in RFID tags increases.

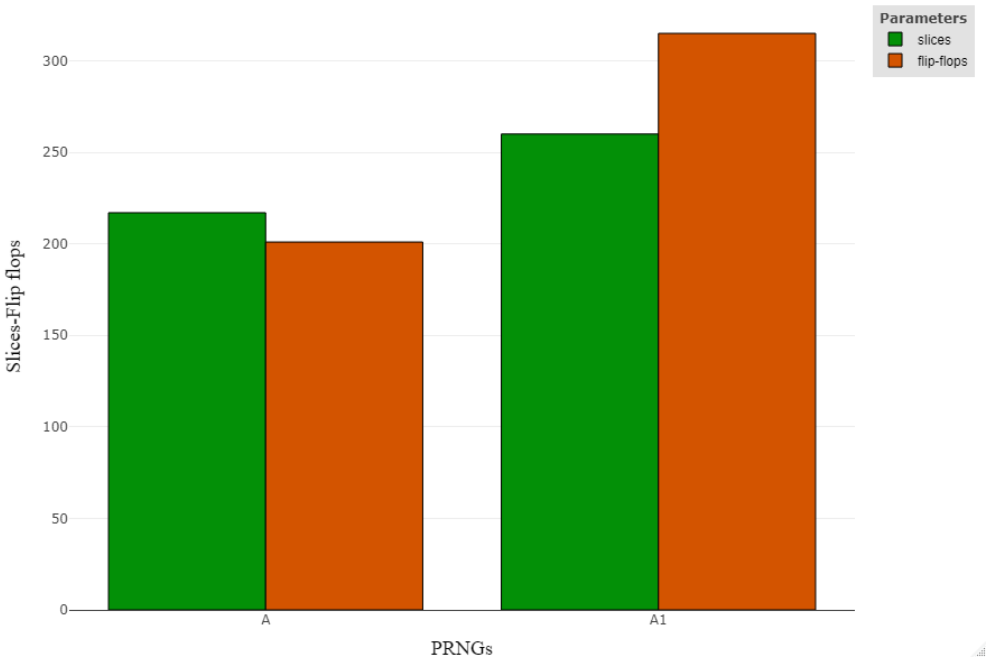

Figure 6.9: Slices and flip-flops comparison of the second protocol with PRNG A-A1

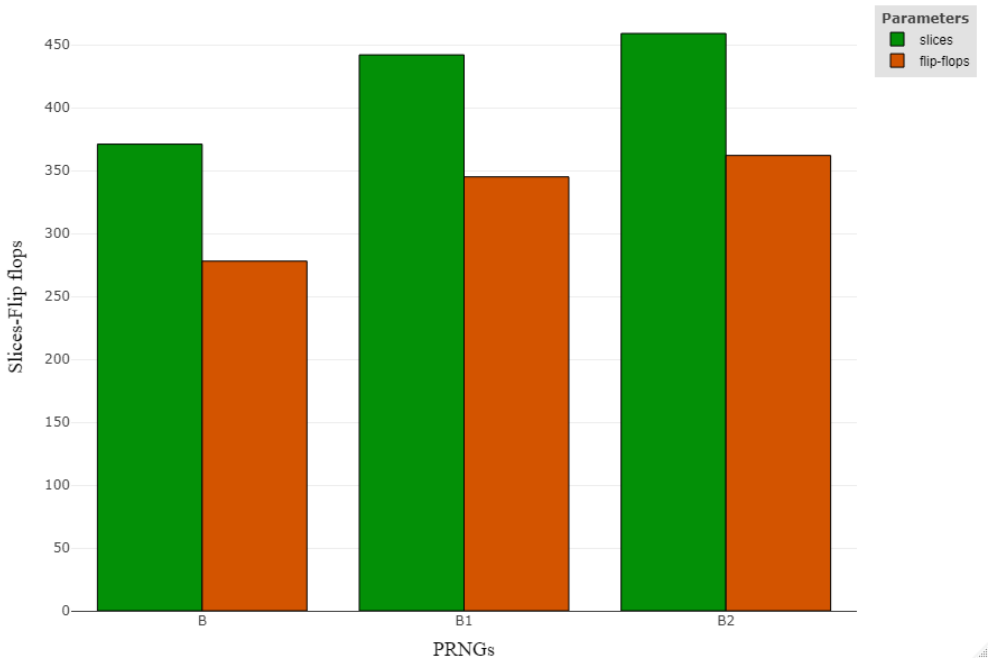

Figure 6.10: Slices and flip-flops comparison of the second protocol with PRNG B-B1-B2

Next, Table 6.7 and Figure 6.11 show how the number of logic elements increases its data, just like the first protocol. This information establishes that the second protocol will occupy much more space than the first protocol because it has greater complexity in its algorithm. Also, the type B generator and its modifications are more complex when performing its operations than generator type A. So, this will affect the increase in the value of the other parameters analyzed, and the information processing time in the RFID tags increases. Likewise, we can see in Figure 6.5 how the number of logic elements is significantly lower than the logic elements in Figure 6.11 when implementing the type A and A_1 pseudo-random number generator, and the type B and B_1 generator. In conclusion, the type A and B generators with their respective modifications adapt better to the second protocol and also require a greater amount of area when being implemented in the RFID tags.

	Pseudo-Random Number Generator				
Parameters	A	A1	B	B1	B2
Logic elements	537	559	581	614	658
Propagation delay (ns)	23.732	24.812	25.924	26.462	26.951
Clock to output delay (ns)	21.894	22.324	22.792	23.873	24.163
Critical path register to register delay (ns)	1.056	1.123	1.187	1.208	1.241

Table 6.7: Results of logic elements, propagation delay, clock output delay and critical path of second protocol 32 bits

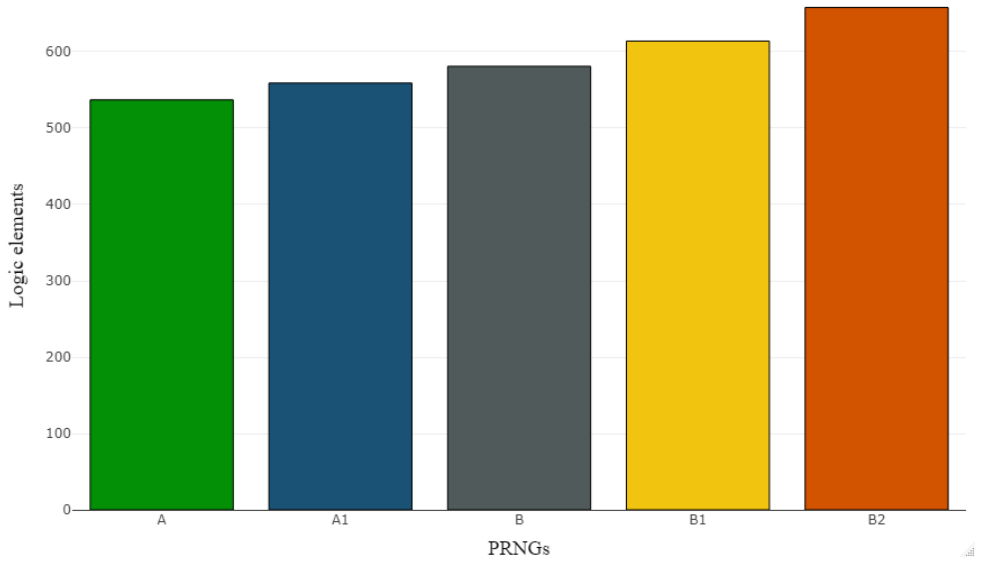

Figure 6.11: Logic elements of the second protocol 32 bits

	Pseudo-Random Number Generator				
Parameters	A	A1	B	B1	B2
Logic elements	554	593	624	674	693
Propagation delay (ns)	25.553	25.647	25.769	25.916	26.089
Clock to output delay (ns)	22.461	22.539	22.642	22.749	22.947
Critical path register to register delay (ns)	1.342	1.465	1.487	1.503	1.519

Table 6.8: Results of power,delay and PDP estimation of second protocol 64 bits

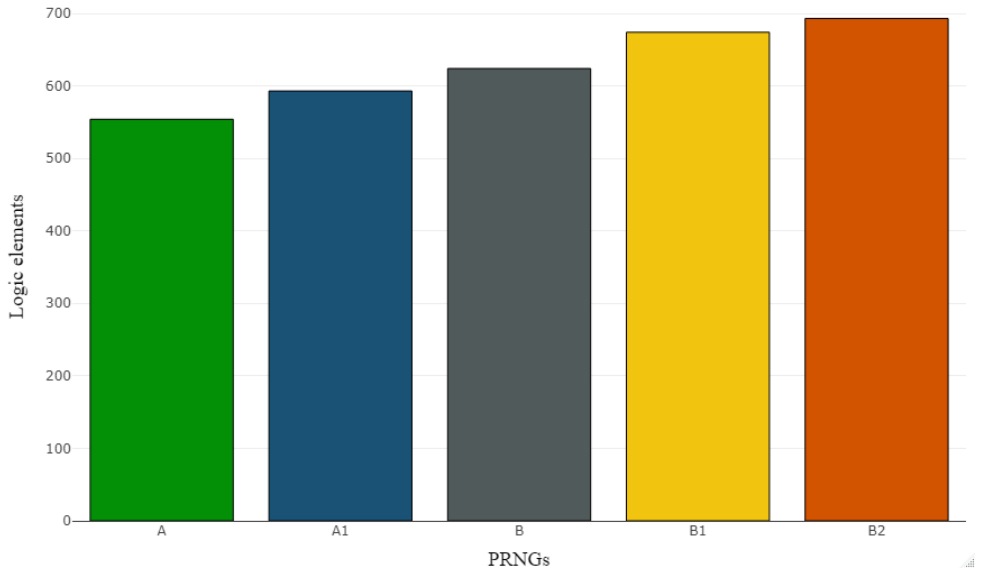

Figure 6.12: Logic elements of the second protocol 64 bits

In Figures 6.6, 6.12 and Table 6.4 and 6.8, the first and second protocols with 64 bits increase the number of logic elements, obtaining similar results as the 32-bit protocols previously analyzed. This is due to the fact that both protocols grow in complexity if increases the number of bits and also that the complexity of the pseudo-random number generators will produce an increase in complexity when implemented in VHDL. In addition, it can be observed in the first protocol with 32 and 64 bits, the information processing time is lower with respect to the second protocol. However, the second protocol has more logic elements. Thus, we can conclude that the second protocol occupies the largest area in RFID tags.

6.1.3 Power consumption of implemented authentication protocols

In this section, the efficiency of authentication protocols using Altera Quartus II tool to generate accurate predictions about power consumption is also addressed. In the Tables 6.9 and 6.10 show the results of the analysis of the authentication protocols with 32 bits using the Altera Quartus II tool. These results show a slight increase in both protocols with both static power dissipation and power–delay product (PDP), due to the complexity of the authentication protocol algorithm implemented in VHDL and the type of pseudo-random generator (A, A_1, B, B_1, B_2). which is used by the protocol.

	Pseudo-Random Number Generator				
Parameters	A	A1	B	B1	B2
Static power disipation (nW)	31.06	31.35	31.58	31.72	31.93
PDP (pJ)	755	769	785	797	807

Table 6.9: Results of static power disipation and PDP of the first protocol 32 bits

	Pseudo-Random Number Generator				
Parameters	A	A1	B	B1	B2
Static power disipation (nW)	30.32	30.45	30.63	30.71	30.97
PDP (pJ)	731	759	774	793	801

Table 6.10: Results of static power disipation and PDP of the second protocol 32 bits

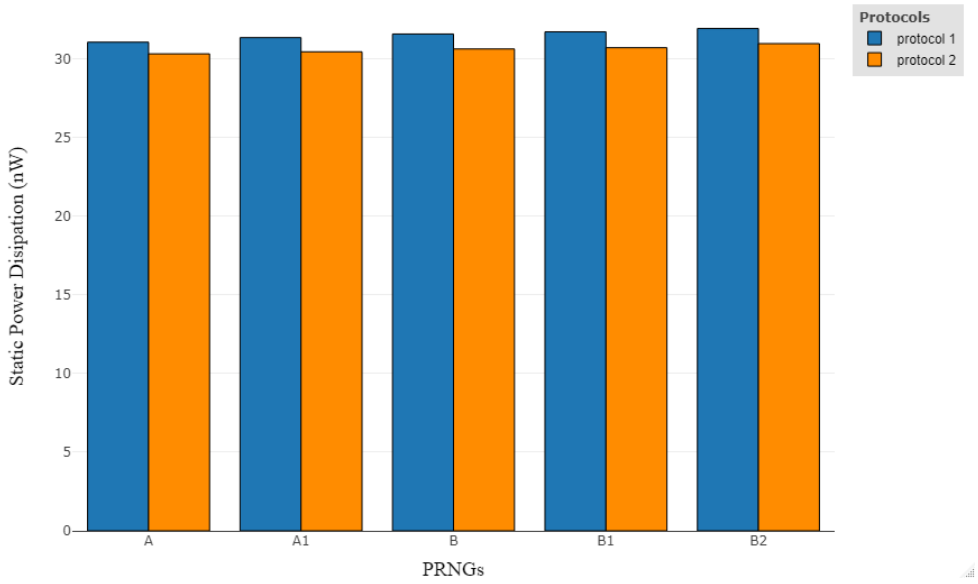

Figure 6.13: Comparison of static power disipation 32 bits

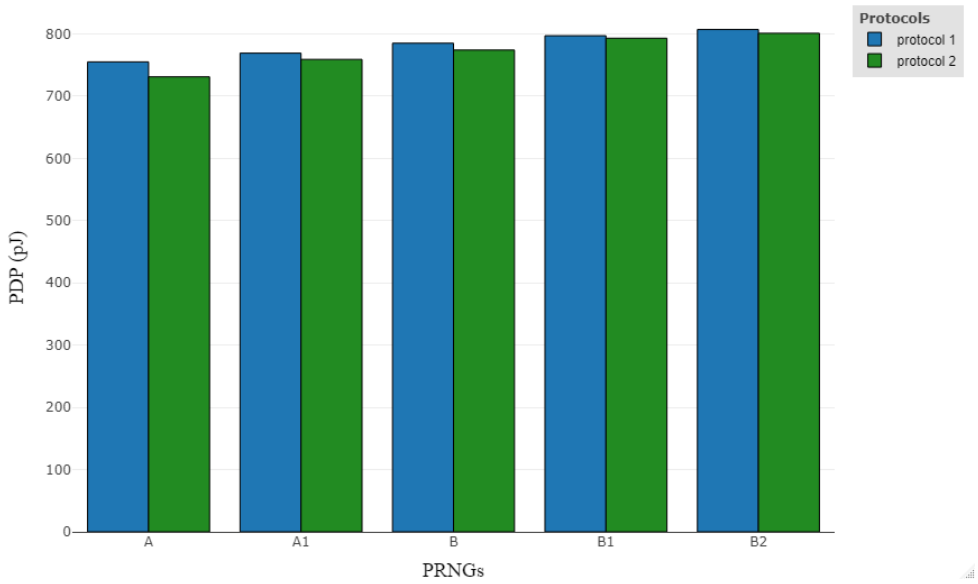

Figure 6.14: Comparison of PDP 32 bits

In Tables 6.9 and 6.10, we can see that the static power dissipation and the PDP are increased in both the first and second protocols. This is due to the complexity of the protocols. However, in Figure 6.13 we make a comparison between both protocols taking static power dissipation as the analysis parameter. We can see that the first protocol (blue) uses more power than the second protocol (orange). In Figure 6.14, a similar analysis was performed when comparing the PDP parameter of both protocols. The first protocol (blue) consumes more energy when using the generator type A and A1 than the second protocol (green). However, by implementing the type B, B_1 and B_2 generator, the difference in power consumption between both protocols is reduced. According to the Tables 6.3 and 6.7, we can conclude that the first protocol consumes more energy and power, which will cause a decrease in processing times and access to information and therefore is faster than the second protocol.

In the Tables 6.11 and 6.12 we can observe similar results to those obtained from the analysis with the Altera Quartus II tool of the authentication protocols with 64 bits. The parameters analyzed are slightly increased according to the level of complexity of the protocol algorithm, the generator that is being implemented . Note that in this analysis, the number of bits used was increased from 32 to 64.

	Pseudo-Random Number Generator				
Parameters	A	A1	B	B1	B2
Static power disipation	31.18	31.49	31.64	31.87	32.06
PDP	771	792	807	816	831

Table 6.11: Results of static power disipation and PDP of the first protocol 64 bits

	Pseudo-Random Number Generator				
Parameters	A	A1	B	B1	B2
Static power disipation	30.78	31.14	31.45	31.64	31.78
PDP	763	781	797	811	824

Table 6.12: Results of static power disipation and PDP of the second protocol 64 bits

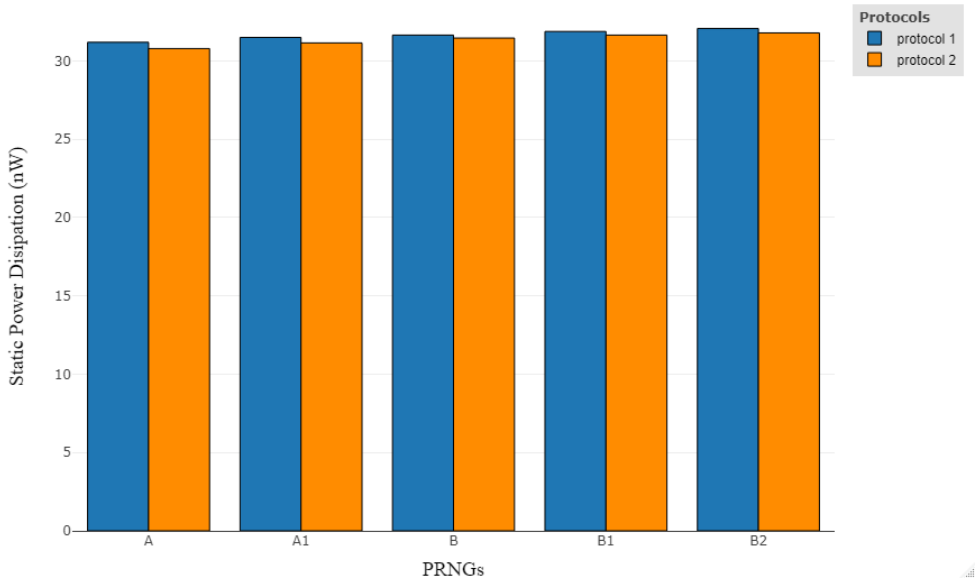

Figure 6.15: Comparison of static power disipation 64 bits

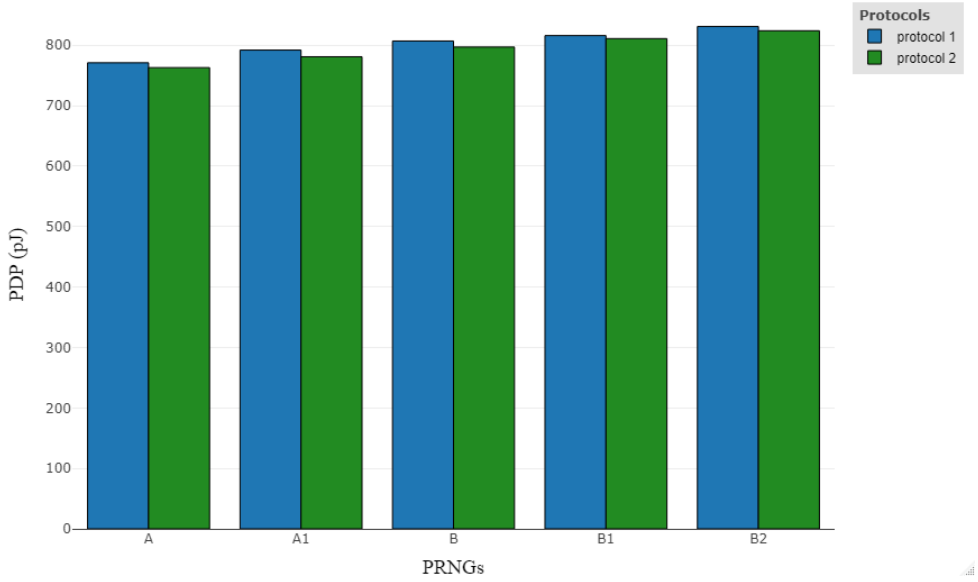

Figure 6.16: Comparison of PDP 64 bits

In Figure 6.15, we can see the comparison between the first protocol (blue) and the second protocol (orange) taking static power dissipation as the analysis parameter. The first protocol uses more power than the second protocol regardless of the type of pseudo-random generator that we are implementing in the protocol (blue). The second parameter that was previously analyzed is the PDP that, as we can see in Figure 6.16, the first protocol(blue) consumes more energy than the second (green). However, the difference between both protocols is considerably small compared to the data obtained in Figure 6.13 and Figure 6.15. The difference in static power dissipation between both protocols is significant. Based on this information, we can conclude that the first protocol is faster than the second. In fact, the first protocol utilizes more power and consumes more energy in the transmission of information between the tag and the reader either with 32 or 64 bits.

Chapter 7

Conclusions

This chapter presents our conclusions about implementation, results, analysis, and future research ideas related to authentication protocols for RFID.

7.1 Conclusions

1. The authentication protocols suitable for RFID technology are flyweight rfid authentication protocol [33] and the protocol proposed by Cheng and Deng [37]. These protocols are robust against external threats and light enough to be implemented in RFID tags.

2. The implementation and analysis of two authentication protocols using hardware description language has been implemented using pseudo-random number generators. These protocols were selected for their lightness and security features.

3. The synthesis of the two selected protocols and the different pseudo-random number generators has been carried out. We have also carried out an analysis of the results obtained from these syntheses, based on aspects such as the number of LUTS and flip-flops. The data obtained allow us to conclude that the second implemented protocol will always require a greater number of logic elements, LUTs, slices and flip-flops, due to the greater complexity of its algorithm. With regard resources, the results show how the number of bits used increases the area and power consumption of the digital circuit.

4. Both protocols allow mutual authentication of the reader and the tag. Also, the second protocol performs more complex operations to obtain the data exchanged between the reader and the tag. On the contrary, the first protocol executes simpler operations, therefore the exchange of information between the tag and the reader will be faster.

5. Taking into account all the above, the designs implemented using protocol 1 with generator A of 32 bits and protocol 2 with generator A of 64 bits have been selected as the most suitable options. Also, the first protocol is faster in the transmission of information, but the second protocol is more secure due to the complexity of

its structure. Both authentication protocols comply with the area restrictions presented by RFID tags and cover the minimum security requirements necessary for this type of system, thus fulfilling the objectives set out in this project.

7.2 Future Work

The following points are proposed as future work:

1. Search and implementation of new lightweight protocols that can be applied to low-cost RFID tags.
2. Extend the model of the evaluation of the implemented protocols to obtain reliable tests of their level of security.
3. Implement simpler pseudo-random generators that meet the requirements of the authentication protocols.
4. Hardware implementation of the RFID authentication protocols proposed in this thesis project.

Bibliography

[1] C. Ohms, "DESIGN AND IMPLEMENTATION OF A SECURITY LAYER FOR RFID SYSTEMS," pp. 69–83, no date.

[2] D. Parkash, T. Kundu, and P. Kaur, "The RFID Technology and ITS Applications: A Review," *International Journal of Electronics, Communication & Instrumentation Engineering Research and Development (IJECIERD)*, vol. 2, no. 3, pp. 109–120, 2012. [Online]. Available: http://www.tjprc.org/view{_}archives.php?year=2012{&}id=16{&}jtype=2{&}page=2

[3] D. Daya Priyanka, T. Jayaprabha, D. Daya Florance, A. Jayanthi, and E. Ajitha, "A survey on applications of RFID technology," *Indian Journal of Science and Technology*, vol. 9, no. 2, pp. 1–5, 2016.

[4] J. S. Chou, "An efficient mutual authentication RFID scheme based on elliptic curve cryptography," *Journal of Supercomputing*, vol. 70, no. 1, pp. 75–94, 2014.

[5] R. Want, "An introduction to RFID technology," *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 25–33, 2006.

[6] Y. Yousuf and V. Potdar, "A survey of RFID authentication protocols," *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, no. January 2008, pp. 1346–1350, 2008.

[7] P. Dass and H. Om, "A Secure Authentication Scheme for RFID Systems," *Physics Procedia*, vol. 78, no. December 2015, pp. 100–106, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.procs.2016.02.017

[8] A. Alqarni, M. Alabdulhafith, and S. Sampalli, "A proposed RFID authentication protocol based on two stages of authentication," *Procedia Computer Science*, vol. 37, pp. 503–510, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.procs.2014.08.075

[9] R. Baashirah and A. Abuzneid, "Survey on prominent RFID authentication protocols for passive tags," *Sensors (Switzerland)*, vol. 18, no. 10, 2018.

[10] B. Fennani, H. Hamam, and A. O. Dahmane, "RFID overview," *Proceedings of the International Conference on Microelectronics, ICM*, no. May 2019, 2011.

[11] R. Baashirah and A. Abuzneid, "Survey on prominent RFID authentication protocols for passive tags," *Sensors (Switzerland)*, vol. 18, no. 10, 2018.

[12] P. Peris and J. C. Hern, "Ph . D . THESIS Lightweight Cryptography in Radio Frequency Identification (RFID) Systems," *Pdfs.Semanticscholar.Org*, no. October, p. 270, 2008. [Online]. Available: https://pdfs.semanticscholar.org/136b/d92b7274c7e48ca8b0f3c03b304ed3af0a7c.pdf{%}0Ahttp://orff.uc3m.es/bitstream/handle/10016/5093/Tesis{_}Pedro{_}Peris{_}Lopez.pdf?sequence=1

[13] T. S. López, D. C. Ranasinghe, B. Patkai, and D. McFarlane, "Taxonomy, technology and applications of smart objects," *Information Systems Frontiers*, vol. 13, no. 2, pp. 281–300, 2011.

[14] D. Weldemedhin, "RFID based Anti-theft System for Metropolia UAS Elec- tronics laboratories," 2016.

[15] V. R. Vijaykumar and S. Elango, "Hardware implementation of tag-reader mutual authentication protocol for RFID systems," *Integration, the VLSI Journal*, vol. 47, no. 1, pp. 123–129, 2014.

[16] M. Kwak, J. Kim, and K. Kim, "Desynchronization and Cloning resistant lightweight RFID authentication protocol using integer arithmetic for low-cost tags," *Review Literature And Arts Of The Americas*, 2009.

[17] M. Khalid, U. Mujahid, and N. U. I. Muhammad, "Ultralightweight RFID Authentication Protocols for Low-Cost Passive RFID Tags," *Security and Communication Networks*, vol. 2019, 2019.

[18] EPCglobal, "Specification for RFID Air Interface EPC ™ Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz – 960 MHz," *Intellectual Property*, no. October, 2006.

[19] R. I. Cuc, "RFID - EPC Código electrónico de producto como herramienta de control de merma," *Inge Cuc*, vol. 9, no. 2, pp. 11–20, 2013.

[20] P. D. E. E. Documento, "Sobre EPCglobal."

[21] GS1, "EPC Information Services (EPCIS) Standard," pp. 1–138, 2016. [Online]. Available: https://www.gs1.org/sites/default/files/docs/epc/EPCIS-Standard-1.2-r-2016-09-29.pdf

[22] M. J. Martinez Belmonte, "Validación de protocolos de acceso al medio probabilísticos en sistemas RFID con dispositivos pasivos," 2008.

[23] J. I. Portillo García, A. B. Bermejo Nieto, and A. M. Bernardos Barbolla, *tecnologia de identificacion por radiofrecuencia (RFID)*, 2008. [Online]. Available: http://www.tagingenieros.com/sites/default/files/vt13_rfid_0.pdf

[24] G. Krishna and S. Roy, "Fundamentals of FPGA Architecture," *Fundamentals of FPGA Architecture*, no. November, pp. 12–30, 2017. [Online]. Available: www.tscipub.com

[25] P. C. Jorgensen, "Finite State Machines," *The Craft of Model-Based Testing*, pp. 55–79, 2017.

[26] G. Linden and D. Somaya, "System-on-a-Chip Integration in the Semiconductor Industry: Industry Structure and Firm Strategies," *SSRN Electronic Journal*, no. June 2003, 2005.

[27] H. Chien, "Sasi: A new ultralightweight rfid authentication protocol providing strong authentication and strong integrity," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 4, pp. 337–340, 2007.

[28] G. Tsudik, "Ya-trap: yet another trivial rfid authentication protocol," in *Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06)*, 2006, pp. 4 pp.–643.

[29] B. Song and C. J. Mitchell, "Rfid authentication protocol for low-cost tags," in *Proceedings of the First ACM Conference on Wireless Network Security*, ser. WiSec '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 140–147. [Online]. Available: https://doi.org/10.1145/1352533.1352556

[30] M. Safkhani, C. Camara, P. Peris-Lopez, and N. Bagheri, "Rseap2: An enhanced version of rseap, an rfid based authentication protocol for vehicular cloud computing," *Vehicular Communications*, vol. 28, p. 100311, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209620300826

[31] S. Izza, M. Benssalah, and K. Drouiche, "An enhanced scalable and secure rfid authentication protocol for wban within an iot environment," *Journal of Information Security and Applications*, vol. 58, p. 102705, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212620308516

[32] M. Hosseinzadeh, O. H. Ahmed, S. H. Ahmed, C. Trinh, N. Bagheri, S. Kumari, J. Lansky, and B. Huynh, "An enhanced authentication protocol for rfid systems," *IEEE Access*, vol. 8, pp. 126 977–126 987, 2020.

[33] M. Burmester and J. Munilla, "A flyweight RFID authentication protocol," *Workshop on RFID Security*, pp. 1–14, 2009. [Online]. Available: http://eprint.iacr.org/2009/212.pdf

[34] S. Version, "ModelSim ® Tutorial," 2008.

[35] S. Holgado and X.-i. I. Software, "Tutorial Xilinx -ISE Manejo básico Xilinx -ISE," 2009.

[36] P. S. Landaeta, "Introducción a las FPGAs Usando el Chip CYCLONE II Introduction to FPGAs Using CYCLONE II Chip," no. February, pp. 0–10, 2019.

[37] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Tapiador, and J. C. Van Der Lubbe, "Cryptanalysis of an EPC Class-1 Generation-2 standard compliant authentication protocol," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 6, pp. 1061–1069, 2011.

[38] P. F. I. N. D. E. Carrera, "ESCUELA POLITÉCNICA SUPERIOR INGENIERÍA TÉCNICA INDUSTRIAL : ELECTRÓNICA INDUSTRIAL GENERADORES DE NÚMEROS PSEUDO-ALEATORIOS (PRNG ´ S)," 2009.

Chapter 8

annexes

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Sist_autenticacion is
port (clk, reset: in std_logic;
query: in std_logic;
aceptado: in std_logic;
dato_e_lector: in std_logic_vector (31 downto 0);
dato_s_lector: out std_logic_vector (31 downto 0));
end Sist_autenticacion;
architecture Behavioral of Sist_autenticacion is
signal tcumplido_a, fin_generacion_a: std_logic;
signal activacion_a, generar_a: std_logic;
signal escribir_a: std_logic;
signal posicion_a: std_logic_vector (1 downto 0);
signal dato_s_mem: std_logic_vector (31 downto 0);
signal dato_e_mem: std_logic_vector (31 downto 0);
component Maquina
port( clk : in STD_LOGIC;
reset : in STD_LOGIC;
query : in STD_LOGIC;
dato_memoria: in std_logic_vector (31 downto 0);
dato_generado: in std_logic_vector (31 downto 0);
dato_e_lector: in std_logic_vector (31 downto 0);
tcumplido : in STD_LOGIC;
```

```
aceptado : in STD_LOGIC;
fin_generacion: in std_logic;
dato_s_lector: out std_logic_vector (31 downto 0);
escribir: out std_logic;
posicion: out std_logic_vector (1 downto 0);
generar : out STD_LOGIC;
activacion : out STD_LOGIC);
end component;
component Temporizador
port( clk : in STD_LOGIC;
reset : in STD_LOGIC;
activacion : in STD_LOGIC;
tcumplido : out STD_LOGIC);
end component;
component Memoria
port( clk : in STD_LOGIC;
reset : in STD_LOGIC;
dato_entrada : in STD_LOGIC_VECTOR (31 downto 0);
dato_salida : out STD_LOGIC_VECTOR (31 downto 0);
escribir : in STD_LOGIC;
posicion : in std_logic_vector(1 downto 0));
end component;
component prngA8
GENERIC(N:INTEGER:=64);
port( z0:OUT std_logic_vector((n/2)-1 DOWNTO 0);
INICIO: IN STD_LOGIC;
FIN: OUT STD_LOGIC;
CLK,RESET: IN STD_LOGIC);
end component;
begin
mi_Maquina: Maquina port map (clk, reset, query, dato_s_mem, dato_e_mem,
dato_e_lector, tcumplido_a, aceptado,
fin_generacion_a, dato_s_lector, escribir_a, posicion_a, generar_a, activacion_a);
mi_Temporizador: Temporizador port map (clk, reset, activacion_a, tcumplido_a);
mi_Memoria: Memoria port map (clk, reset, dato_e_mem, dato_s_mem, escribir_a,
posicion_a);
mi_Generador: prngA8 port map (dato_e_mem , generar_a, fin_generacion_a, clk, reset
```

```
);
end Behavioral;
------ MAQUINA DE ESTADOS ------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Maquina is
Port (clk : in STD_LOGIC;
reset : in STD_LOGIC;
query : in STD_LOGIC;
dato_memoria: in std_logic_vector (31 downto 0);
dato_generado: in std_logic_vector (31 downto 0);
dato_e_lector: in std_logic_vector (31 downto 0);
tcumplido : in STD_LOGIC;
aceptado : in STD_LOGIC;
fin_generacion: in std_logic;
dato_s_lector: out std_logic_vector (31 downto 0);
escribir: out std_logic;
posicion: out std_logic_vector (1 downto 0);
generar : out STD_LOGIC;
activacion : out STD_LOGIC);
end Maquina;
architecture Maquina_a of Maquina is
type estado is (inicializacion, reposo, envio_rn1, esp_comp_rn2, envio_rn3,
memorizar_rn3,
envio_rn4, espera_acep, esp_comp_rn3, envio_rn5, actualizacion);
signal actual, siguiente: estado;
signal alarma, cont, fin, gen_aux: std_logic;
signal cont_aux: std_logic_vector (1 downto 0);
begin
process(clk, reset)
begin
if reset='1' then
actual <= inicializacion;
elsif clk'event and clk='1' then
actual <= siguiente;
```

```
end if;
end process;
process(actual, query, tcumplido, dato_e_lector, fin, aceptado, cont_aux,
fin_generacion, dato_memoria,
dato_generado, gen_aux, alarma)
variable cont_alarma: std_logic;
begin
case actual is
when inicializacion =>escribir<='1';
activacion <= '0';
generar<=gen_aux;
posicion<=cont_aux;
dato_s_lector<=(others=>'0');
if fin= '1' then
siguiente <= reposo;
else
siguiente <= inicializacion;
end if;
when reposo => escribir <= '0';
activacion <= '0';
generar <='0';
posicion<="00";
dato_s_lector<=(others=>'0');
if query = '1' then
siguiente <= envio_rn1;
else
siguiente <= reposo;
end if;
when envio_rn1 => escribir <= '0';
activacion <= '0';
generar <='0';
posicion<="01";
dato_s_lector<=dato_memoria;
siguiente <= esp_comp_rn2;
when esp_comp_rn2 => escribir <= '0';
activacion <= '1';
generar <='0';
```

```
posicion<="10";
dato_s_lector<=(others=>'0');
if tcumplido = '0' then
if dato_memoria=dato_e_lector then
if alarma = '1' then
siguiente <= memorizar_rn3;
else
siguiente <= envio_rn3;
end if;
else
siguiente <= esp_comp_rn2;
end if;
else
siguiente <= reposo;
end if;
when envio_rn3 => escribir <= '0';
activacion <= '0';
generar <='1';
posicion<="00";
dato_s_lector<=dato_generado;
if fin_generacion='1' then
siguiente <= espera_acep;
else
siguiente <= envio_rn3;
end if;
when memorizar_rn3 => escribir <= '1';
activacion <= '0';
generar <='1';
posicion<="11";
dato_s_lector<=dato_generado;
if fin_generacion='1' then
siguiente <= envio_rn4;
else
siguiente<= memorizar_rn3;
end if;
when envio_rn4 => escribir <= '0';
activacion <= '0';
```

```
generar <='1';
posicion<="11";
dato_s_lector<= dato_generado;
if fin_generacion='1' then
siguiente <= esp_comp_rn3;
else
siguiente <= envio_rn4;
end if;
when espera_acep => escribir <= '0';
activacion <= '1';
generar <='0';
posicion<="00";
dato_s_lector<=(others=>'0');
if tcumplido = '1' then
siguiente <= reposo;
elsif tcumplido = '0' then
if aceptado = '1' then
siguiente <= actualizacion;
else
siguiente <= espera_acep;
end if;
end if;
when esp_comp_rn3 => escribir <= '0';
activacion <= '1';
generar <='0';
posicion<="11";
dato_s_lector<=(others=>'0');
if tcumplido = '0' then
if dato_memoria=dato_e_lector then
siguiente <= envio_rn5;
else
siguiente <= esp_comp_rn3;
end if;
else
siguiente <= reposo;
end if;
when envio_rn5 => escribir <= '0';
```

```
activacion <= '0';
generar <='1';
posicion<="00";
dato_s_lector<=dato_generado;
if fin_generacion='1' then
siguiente <= espera_acep;
else
siguiente <= envio_rn5;
end if;
when actualizacion => escribir<='1';
activacion <= '0';
generar<=gen_aux;
posicion<=cont_aux;
dato_s_lector<=(others=>'0');
if fin = '1' then
siguiente <= reposo;
else
siguiente <= actualizacion;
end if;
end case;
end process;
process(clk, reset)
begin
if reset='1' then
cont_aux<="01";
elsif clk'event and clk='1' then
if actual=actualizacion or actual=inicializacion then
if fin_generacion='1' then
cont_aux<=cont_aux+'1';
end if;
else
cont_aux<="01";
end if;
end if;
end process;
fin<='1' when cont_aux="11" else '0';
gen_aux<='1' when cont_aux>"00" and cont_aux<"11" else '0';
```

```
process(clk, reset)
begin
if reset='1' then
cont<='0';
alarma<='0';
elsif clk'event and clk='1' then
if actual=reposo then
alarma<=cont;
elsif actual=envio_rn1 then
cont<='1';
end if;
end if;
end process;

end Maquina_a;
------ TEMPORIZADOR ------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Temporizador is
Port ( clk : in STD_LOGIC;
reset : in STD_LOGIC;
activacion : in STD_LOGIC;
tcumplido : out std_logic);
end Temporizador;
architecture Temporizador_a of Temporizador is
signal contador: std_logic_vector (4 downto 0);
begin
process(clk, reset)
variable contador_aux: std_logic_vector (4 downto 0):="00000";
begin
if reset = '1' then
contador_aux := "00000";
elsif clk'event and clk='1' then
if activacion = '1' then
contador_aux := contador_aux + '1';
```

```
else
contador_aux := "00000";
end if;
end if;
contador<=contador_aux;
end process;
tcumplido <= '1' when contador="00101" else '0';
end Temporizador_a;
------ MEMORIA ------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Memoria is
Port ( clk : in STD_LOGIC;
reset : in STD_LOGIC;
dato_entrada : in STD_LOGIC_VECTOR (31 downto 0);
dato_salida: out STD_LOGIC_VECTOR (31 downto 0);
escribir : in STD_LOGIC;
posicion : in std_logic_vector(1 downto 0));
end Memoria;
architecture Memoria_a of Memoria is
signal rn1, rn2, rn3: std_logic_vector(31 downto 0);
begin
process(clk, reset)
variable aux: std_logic_vector(31 downto 0);
begin
if reset='1' then
rn3<=(others=>'0');
elsif clk'event and clk='1' then
if escribir='1' then
case posicion is
when "01" => rn1<=dato_entrada;
when "10" => rn2<=dato_entrada;
when "11" => rn3<=dato_entrada;
when others => aux:=dato_entrada;
end case;
```

```
end if;
end if;
end process;
dato_salida<=rn1 when posicion="001" else
rn2 when posicion="010" else
rn3 when posicion="011" else
(others=>'0');
end Memoria_a;


%%%S%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
-------------- Protocolo 2 ----------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5%

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

--ENTIDAD
ENTITY Protocolo_Azumi_32 IS
GENERIC(N:INTEGER:=32);
PORT(
Dato_Reader_entrada: In Std_logic_vector (2*N-1 Downto 0);
Dato_Reader_salida: out std_logic_vector(4*N-1 downto 0);
Flag_Reader: In std_logic;
Flag_tag: out std_logic;
ok, no_ok: out std_logic;
CLK,RESET: IN STD_LOGIC
);
END Protocolo_Azumi_32;


architecture protocolo_Azumi_32_a of protocolo_Azumi_32 is
 Signal a,b,dato_xor: Std_logic_vector(N-1 downto 0);
 signal termino1,termino2,salida: std_logic_vector(N-1 downto 0);
 signal dato_random: std_logic_vector(N-1 downto 0);
 signal dato_prgn: Std_logic_vector(N/2-1 downto 0);
```

```
 signal semilla_prgn: Std_logic_vector(N-1 downto 0);
 signal Flag_prgn,flag_random,inicio_prgn,inicio_random: Std_logic;
 signal carryin,carryout: std_logic;

Component Azumi_32
 GENERIC(N:INTEGER:=32);
PORT(
Dato_Reader_entrada: In Std_logic_vector (2*N-1 Downto 0);
Dato_Reader_salida: out std_logic_vector(4*N-1 downto 0);
Dato_xor: In Std_logic_vector(N-1 Downto 0);
Dato_prgn: In Std_logic_vector(N/2-1 Downto 0);
Dato_random: In Std_logic_vector(N-1 downto 0);
Operador_A: out Std_logic_vector((N-1) downto 0);
Operador_B: out Std_logic_vector (N-1 downto 0);
Semilla_Prgn: out Std_logic_vector (N-1 downto 0);
Flag_Prgn: in std_logic;
Flag_Random: in std_logic;
Flag_tag: out std_logic;
Inicio_prgn: out std_logic;
Inicio_random: out std_logic;
Flag_Reader: in std_logic;
ok, no_ok: out std_logic;
CLK,RESET: IN STD_LOGIC
);
 end component;

Component funcion_xor_32
 GENERIC(N:INTEGER:=32);
 port (a : in Std_logic_vector(N-1 DOWNTO 0);
        b : in Std_logic_vector(N-1 DOWNTO 0);
        dato_xor : out Std_logic_vector(N-1 DOWNTO 0)
 );
 end component;


Component prngc8_1
 GENERIC(N:INTEGER:=32);
```

```
 PORT(
 z0: OUT STD_LOGIC_VECTOR(((n/2)-1) DOWNTO 0);
 seed: IN STD_LOGIC_VECTOR (N-1 downto 0);
 INICIO: IN STD_LOGIC;
 FIN: OUT STD_LOGIC;
 CLK,RESET: IN STD_LOGIC;
 a1,a2: out std_logic_vector(N-1 downto 0);
 resultsuma:in std_logic_vector (N-1 downto 0)
 );
end component;


Component prngc8_4
 GENERIC(N:INTEGER:=32);
 PORT(
 z0: OUT STD_LOGIC_VECTOR(((n/2)-1) DOWNTO 0);
 seed: IN STD_LOGIC_VECTOR (N-1 downto 0);
 INICIO: IN STD_LOGIC;
 FIN: OUT STD_LOGIC;
 CLK,RESET: IN STD_LOGIC
 );
end component;

Component sumador_2
 GENERIC (N:INTEGER:=32);
 Port( termino1 :IN std_logic_Vector(N-1 DOWNTO 0);
 termino2 :IN std_logic_Vector(N-1 DOWNTO 0);
 salida: out std_logic_Vector(N-1 DOWNTO 0));
END component;

Component sumador_3
 GENERIC (N:INTEGER:=32);
 Port( termino1 : in std_logic_Vector((N/2)-1 DOWNTO 0);
 termino2 : in std_logic_Vector((N/2)-1 DOWNTO 0);
 carryin : in std_logic;
 salida : out std_logic_Vector((N/2)-1 DOWNTO 0);
 carryout : out std_logic);
```

```
END component;

begin
 mi_maquina:azumi_32 Port map (dato_reader_entrada,dato_reader_salida,dato_xor,dato_p
dato_random,a,b,semilla_prgn,flag_prgn,flag_random,flag_tag,inicio_prgn,inicio_random
reset);
 mi_funcion_xor: funcion_xor_32 Port map (a,b,dato_xor);
 --mi_random:random_azumi_32 Port map(dato_random,inicio_random,flag_random,clk, res
 mi_prgn: prngc8_1 Port map(dato_prgn,semilla_prgn,inicio_prgn,flag_prgn, clk,reset,t
 mi_sumador: sumador_2 Port Map (termino1,termino2,salida);
end protocolo_azumi_32_a;


library ieee;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.numeric_std.all;
--ENTIDAD
ENTITY Azumi_32 IS
GENERIC(N:INTEGER:=32);
PORT(
Dato_Reader_entrada: In Std_logic_vector (2*N-1 Downto 0);
Dato_Reader_salida: out std_logic_vector(4*N-1 downto 0);
Dato_xor: In Std_logic_vector(N-1 Downto 0);
Dato_prgn: In Std_logic_vector(N/2-1 Downto 0);
Dato_random: In Std_logic_vector(N-1 downto 0);
Operador_A: out Std_logic_vector((N-1) downto 0);
Operador_B: out Std_logic_vector (N-1 downto 0);
Semilla_Prgn: out Std_logic_vector (N-1 downto 0);
Flag_Prgn: in std_logic;
Flag_Random: in std_logic;
Flag_tag: out std_logic;
Inicio_prgn: out std_logic;
Inicio_random: out std_logic;
Flag_Reader: in std_logic;
ok, no_ok: out std_logic;
```

```
CLK,RESET: IN STD_LOGIC
);
END Azumi_32;

architecture Azumi_32_a of Azumi_32 is
type estado is (espera,inicio,Random_1,obtener_random_1,random_2,
funcion_A1,funcion_A2,funcion_A3,funcion_A4,funcion_A5,funcion_A6,
Prng_A,Prng_A2,Resultado_A,
Funcion_B1,Funcion_B2,Prgn_B,Prgn_B2,Resultado_B,
Funcion_C1,Resultado_C,
Enviar_Tag,Recibir_D,
Funcion_D1,Funcion_D2,Funcion_D3,Funcion_D4,Funcion_D5,Funcion_D6,Funcion_D7,Funcion_
Prng_D,Prng_D2,Resultado_D, Updating,error,comparar,
Updating_k16,Updating_N16,Envio_datos,Prng_k16,Prng_N16, Prng_N16_2,Prng_k16_2);

signal actual,siguiente: estado;
signal h1,h2: std_logic_vector(N-1 downto 0); -- Registros seales intermedias
signal h3:std_logic_vector(N-1 downto 0):= x"00000066";--RND
signal h4: std_logic_vector(N-1 downto 0):=x"00000021";--RND'
signal Nt: std_logic_Vector(2*N-1 downto 0):= x"0000000000000022";
signal EPC16: std_logic_vector(N-1 downto 0):= x"00000033";
signal k16: std_logic_vector(N-1 downto 0):= x"00000044";
signal resultado,Resultado_aux:std_logic_vector(4*N-1 downto 0);
signal N16: std_logic_vector(N-1 downto 0):= x"00000055";
signal flag_h1: std_logic_vector(1 downto 0);
signal aux_h2, aux_k16,aux_N16,aux_h3,aux_h4: std_logic_vector(N-1 downto 0);

begin

--PROCESO SECUENCIAL DE LA MAQUINA DE ESTADOS
process(clk, reset)
begin
 if reset = '1' then
 actual<=espera;
 elsif clk'event and clk='1' then
 actual<=siguiente;
end if;
```

```
end process;

-- Proceso secuencial H1
process(clk,reset)
 begin
 if reset='1' then
 H1<=(others=>'0');
 elsif clk'event and clk='1' then
 if flag_h1="01" then --Guardar el valor de ID16
 h1<=Dato_Reader_entrada(2*n-1 downto N);
 elsif flag_h1="11" then
 h1<=Dato_Xor;

 end if;
end if;
end process;

-- Proceso secuencial
process(clk,reset)
 begin
 if reset='1' then
 H2<=(others=>'0');
 Resultado<=(others=>'0');
 h3<=(others=>'0');
 h4<=(others=>'0');
 elsif clk'event and clk='1' then

 h2<=aux_h2;
 k16<=aux_k16;
 N16<=aux_N16;
 h3<=aux_h3;
 h4<=aux_h4;
 Resultado<=Resultado_aux;

 end if;
end process;
```

```
-- Proceso Combinacional
Process (Actual,flag_reader,flag_random,dato_xor,flag_Prgn,h1,h2,h3,Nt,EPC16,H4,k16,R
Begin
 Case Actual Is
 When Espera=>
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=Dato_Reader_entrada(n-1 downto 0);
 aux_k16<=x"00000044";
 aux_N16<=x"00000055";
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 if flag_reader = '1' then
 siguiente<=inicio;
 else
 siguiente<=actual;
 end if;


When Inicio=>
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
```

```
 flag_tag<='0';
 flag_h1<="01";
 aux_h2<=Dato_Reader_entrada(n-1 downto 0);
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Random_1;


when Random_1=> --Se pide calcular RND
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='1';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=Dato_Reader_entrada(n-1 downto 0);
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Dato_Reader_Salida<=Resultado;
 Resultado_aux<=Resultado;
 If flag_random='1' then
 siguiente<=Obtener_Random_1;
 else
 siguiente<=actual;
 end if;

when obtener_Random_1=>
 Operador_A<=(others=>'0');
```

```
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=Dato_Reader_entrada(n-1 downto 0);
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=dato_random;
 aux_h4<=h4;
 Dato_Reader_Salida<=Resultado;
 Resultado_aux<=Resultado;
 siguiente<=Random_2;

when Random_2=> --Se pide calcular RND'
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='1';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=Dato_Reader_entrada(n-1 downto 0);
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=Dato_Random;
 Dato_Reader_Salida<=Resultado;
 Resultado_aux<=Resultado;
 if Flag_random='1' then
        siguiente<=Funcion_A1;
 else
```

```
        siguiente<=actual;
 end if;

when Funcion_A1 => --XOR (rnd_reader,id16)
 Operador_A<=H1;
 Operador_B<=H2;
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="11";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_A2;

When Funcion_A2 => -- Guardar resultado XOR
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
```

```
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_A3;

When Funcion_A3 => -- XOR (RND,H1)
 Operador_A<=H1;
 Operador_B<=H3;
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="11";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_A4;

When Funcion_A4 => -- Guardar resultado XOR
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
```

```
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_A5;

When Funcion_A5 => -- XOR (Nt,H1)
 Operador_A<=H1;
 Operador_B<=Nt(N-1 downto 0);
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="11";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_A6;

When Funcion_A6 => -- Guardar resultado XOR
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
```

```
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=PRng_A;

When Prng_A => -- Calcular el numero PRNG (low) para A
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<= H1;
 Inicio_PRGN<='1';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 --Resultado_aux<=Resultado(127 downto 96) & Dato_Prgn & Resultado(79 downto 0);
 Resultado_aux(95 downto 80)<=Dato_Prgn;
 Dato_Reader_Salida<=Resultado;
 if flag_Prgn='1' then
        siguiente<=Prng_A2;
 else
        siguiente<=Actual;
 end if;

When Prng_A2 => -- Calcular el numero PRNG (High) para A
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<= Nt(2*N-1 downto N);
 Inicio_PRGN<='1';
 Inicio_Random<='0';
 ok<='0';
```

```
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 if flag_Prgn='1' then
        siguiente<=Resultado_A;
 else
        siguiente<=Actual;
 end if;

when Resultado_A => --Obtener el resultado para A
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 -- Resultado_aux<=Resultado(127 downto 80) & Dato_Prgn & Resultado(63 downto 0);
 Resultado_aux(79 downto 64)<=Dato_Prgn;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_B1;

When Funcion_B1 => -- XOR Para B (RPC16t, RND')
```

```
 Operador_A<=EPC16;
 Operador_B<=H4;
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="11";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_B2;

When Funcion_B2 => -- Guardar el resultado
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=PRgn_B;
```

```
when PRgn_B => -- Obtener el PRGn de B (low)
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<= H1;
 Inicio_PRGN<='1';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 -- Resultado_aux<=Resultado(127 downto 64) & Dato_Prgn & Resultado(47 downto 0);
 Resultado_aux(63 downto 48)<=Dato_Prgn;
 Dato_Reader_Salida<=Resultado;
 if flag_prgn='1' then
        siguiente<=Prgn_B2;
 else
        siguiente<=actual;
 end if;

when PRgn_B2 => -- Obtener el PRGn de B (High)
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=Conv_std_logic_Vector(0,N);
 Inicio_PRGN<='1';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
```

```
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 if flag_prgn='1' then
        siguiente<=Resultado_B;
 else
        siguiente<=actual;
 end if;

when Resultado_B => -- Resultado
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 -- Resultado_aux<=Resultado(127 downto 48) & Dato_Prgn & Resultado(31 downto 0);
 Resultado_aux(47 downto 32)<=Dato_Prgn;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_C1;

When Funcion_C1=> --Obtener C xor Rnd',k16
 Operador_A<=H4;
 Operador_B<=k16;
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
```

```
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="11";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux(127 downto 96)<=H3;
 Resultado_aux(31 downto 0)<=Dato_xor;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Resultado_C;

When Resultado_C=> --Obtener Resultado_C
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Enviar_Tag;

When Enviar_Tag=> --Se envia el valor de Resultado al reader
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
```

```
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='1';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 if flag_reader='1' then
        siguiente<=Recibir_D;
 else
        siguiente<=actual;
 end if;

When Recibir_D=> -- Guardar el valor de D del lector y calcular el tag=>D
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux(31 downto 0)<= Dato_reader_entrada(N-1 downto 0);
 Resultado_aux(127 downto 32)<=conv_std_logic_vector(0,3*N);
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_D1;
```

```
when Funcion_D1=> -- Calcular el valor D desde la tag
 Operador_A<=H4;
 Operador_B<=EPC16;
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="11";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_D2;

when Funcion_D2 => --Obtener valor de XOR
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
```

```
 siguiente<=Funcion_D3;

When Funcion_D3 => -- XOR
 Operador_A<=H1;
 Operador_B<=H3;
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="11";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_D4;

When Funcion_D4 => -- Guardar XOR
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
```

```
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_D5;

When Funcion_D5 => -- XOR
 Operador_A<=H1;
 Operador_B<=H2;
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="11";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_D6;

When Funcion_D6 => -- Guardar
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
```

```
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_D7;

When Funcion_D7 => -- XOR
 Operador_A<=Nt(N
-1 downto 0);
 Operador_B<=H1;
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="11";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Funcion_D8;

When Funcion_D8 => -- Guardar el valor semilla
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
```

```
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=PRng_D;

When PRng_D => -- Calcular el prgn para D (low)
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<= h1;
 Inicio_PRGN<='1';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux(47 downto 32)<=Dato_Prgn;
 Dato_Reader_Salida<=Resultado;
 if flag_prgn='1' then
        siguiente<=Prng_D2;
 else
        siguiente<=Actual;
 end if;

When PRng_D2 => -- Calcular el prgn para D (High)
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=Nt(2*n-1 downto 32);
 Inicio_PRGN<='1';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
```

```
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux(63 downto 48)<=Dato_prgn;

 Dato_Reader_Salida<=Resultado;
 if flag_prgn='1' then
        siguiente<=Resultado_D;
 else
        siguiente<=Actual;
 end if;

When Resultado_D => -- Calcular D
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Comparar;

When Comparar => -- Comparar D tag con D Reader
 Operador_A<=(others=>'0');
```

```
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 if resultado(63 downto 32)=Resultado(31 downto 0) then
        siguiente<=Updating;
 else
        siguiente<=error;
 end if;

When Updating => -- Procolo Ok=> actualizar N16,K16
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='1';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
```

```
 Dato_Reader_Salida<=Resultado;
 siguiente<=Updating_N16;

When Error => -- Abortar protocolo
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='0';
 no_ok<='1';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Error;

When Updating_N16 => -- Xor N16,RND'
 Operador_A<=h4;
 Operador_B<=N16;
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='1';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=dATO_XOR;
 aux_h3<=h3;
 aux_h4<=h4;
```

```
 Dato_Reader_Salida<=Resultado;
 Resultado_aux<=Resultado;
 siguiente<=Prng_N16;

When Prng_N16 => -- PRng N16,RND' parte low
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<= N16;
 Inicio_PRGN<='1';
 Inicio_Random<='0';
 ok<='1';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16(n/2-1 downto 0)<=Dato_prgn;
 aux_h3<=h3;
 aux_h4<=h4;
 Dato_Reader_Salida<=Resultado;
 Resultado_aux<=Resultado;
 if flag_prgn='1' then
        siguiente<=Prng_N16_2;
 else
        siguiente<=Prng_N16;
 end if;

When Prng_N16_2 => -- PRng N16,RND' parte high
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<= N16;
 Inicio_PRGN<='1';
 Inicio_Random<='0';
 ok<='1';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
```

```
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16(n
-1 downto 16)<=Dato_prgn;
 aux_h3<=h3;
 aux_h4<=h4;
 Dato_Reader_Salida<=Resultado;
 Resultado_aux<=Resultado;
 if flag_prgn='1' then
        siguiente<=Updating_k16;
 else
        siguiente<=Prng_N16_2;
 end if;

When Updating_K16 => -- Xor k16,RND'
 Operador_A<=K16;
 Operador_B<=H4;
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='1';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=Dato_Xor;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Dato_Reader_Salida<=Resultado;
 Resultado_aux<=Resultado;
 siguiente<=PRng_K16;

When Prng_k16 => -- PRng k16,RND'
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<= k16;
```

```
 Inicio_PRGN<='1';
 Inicio_Random<='0';
 ok<='1';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16(n/2-1 downto 0)<=dato_prgn;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Dato_Reader_Salida<=Resultado;
 Resultado_aux<=Resultado;
 if flag_prgn='1' then
 siguiente<=Prng_K16_2;
 else
 siguiente<=Prng_k16;
 end if;

When Prng_k16_2 => -- PRng k16,RND' parte high
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<= N16;
 Inicio_PRGN<='1';
 Inicio_Random<='0';
 ok<='1';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16(n-1 downto 16)<=Dato_prgn;
 aux_h3<=h3;
 aux_h4<=h4;
 Dato_Reader_Salida<=Resultado;
 Resultado_aux<=Resultado;
 if flag_prgn='1' then
```

```
        siguiente<=Envio_datos;
 else
        siguiente<=Prng_k16_2;
 end if;

When Envio_datos => -- Estado de envio de datos
 Operador_A<=(others=>'0');
 Operador_B<=(others=>'0');
 Semilla_PRGN<=(others=>'0');
 Inicio_PRGN<='0';
 Inicio_Random<='0';
 ok<='1';
 no_ok<='0';
 flag_tag<='0';
 flag_h1<="00";
 aux_h2<=h2;
 aux_k16<=k16;
 aux_N16<=N16;
 aux_h3<=h3;
 aux_h4<=h4;
 Resultado_aux<=Resultado;
 Dato_Reader_Salida<=Resultado;
 siguiente<=Espera;

 end case;
 end process;

End Azumi_32_a;




library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
```

```
entity funcion_xor_32 is
 GENERIC(N:INTEGER:=32);
 port (a : in Std_logic_vector(N
-1 DOWNTO 0);
 b : in Std_logic_vector(N
-1 DOWNTO 0);

 dato_xor : out Std_logic_vector(N
-1 DOWNTO 0)
 );
end funcion_xor_32;

-- implementacin de comportamiendo del xor
architecture funcion_xor_32_a of funcion_xor_32 is
begin
dato_xor<=a + b;
end funcion_xor_32_a;


-- ALGORITMO A 8 BITS
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
 --ENTIDAD
---GENERADOR PRNG-1, VARIOS SUMADORES
ENTITY prngC8_1 IS
GENERIC(N:INTEGER:=32);
PORT(
z0: OUT STD_LOGIC_VECTOR(((n/2)-1) DOWNTO 0);
seed: IN STD_LOGIC_VECTOR (N-1 downto 0);
INICIO: IN STD_LOGIC;
 FIN: OUT STD_LOGIC;
 CLK,RESET: IN STD_LOGIC;
 a1,a2: out std_logic_vector(N-1 downto 0);
 resultsuma:in std_logic_vector (N-1 downto 0)
);
```

```
END prngC8_1;

architecture opcionA_1 of prngC8_1 is
--DECLARACION DE SEALES
 SIGNAL z,x1,a_z,a_x1: std_logic_Vector((N-1) DOWNTO 0);--z en el reset obtendr el v
 SIGNAL contador_i: INTEGER RANGE 0 TO 64;
 SIGNAL start:STD_LOGIC;

 TYPE ESTADO IS (Reposo,IterandoZ,iterandoy,final);
 SIGNAL ACTUAL,SIGUIENTE:ESTADO;

 BEGIN

 -- PROCESO PARA LA MAQUINA DE ESTADOS
 PROCESS(CLK,RESET)
 BEGIN
 IF reset='1' THEN
 actual<=reposo;
 elsif clk 'EVENT AND clk='1' THEN
 actual<=siguiente;
 END IF;
 END PROCESS;

--MAQUINA DE ESTADOS----------------------------------------------------------------

PROCESS (ACTUAL,Inicio,contador_i,z,x1,seed)variable aux: STD_LOGIC_VECTOR (N-1 downt
 BEGIN
 a_z<=z;-- el registro mantiene estado...
 a_x1<=x1;
 start<='0';
 CASE ACTUAL IS

 WHEN Reposo=>
 if inicio ='1' then
 a_z<=seed;-- valores semillas entrantes, si siempre son los mismos comentar...
 a_x1<=x"00000021";--valores semillas precalculados
 siguiente<=iterandoz;
```

```
 else
 siguiente<=reposo;
 end if;

 WHEN IterandoZ=>
 start<='1';
 aux:=z+x"00000022";---constante 22
 a_z<=(z(n-2 downto 0)& z(n-1))+(aux(0)& aux(n-1 downto 1));
 IF contador_i=31 THEN
 start<='1';
 siguiente<=iterandoy;
 ELSE
 siguiente<=IterandoZ;
 END IF;

 WHEN iterandoy=>
 aux:=(z(n-2 downto 0)& z(n-1))+ (z(0)& z(n-1 downto 1))+z+ x1;
 a_z<=aux XOR z;
 siguiente<=final;

 WHEN final=>
 siguiente<=reposo;
 END CASE;
 END PROCESS;

  --CONTADOR
PROCESS(reset,clk)
 BEGIN
 IF reset='1' THEN
 contador_i <= 0;

 elsif clk'EVENT AND CLK='1' THEN

 if start='1' then

 contador_i<=contador_i+1;
 else
```

```
 contador_i<=0;

 end if;
 end if;
end process;

fin<='1' When(actual=final) ELSE '0';
z0<=z((n/2)-1 downto 0);

PROCESS(reset,clk)
 BEGIN
 IF reset='1' THEN
 z <= x"00000023";--valores semillas precalculados
 x1 <= x"00000032"; --valores semillas precalculados
 ELSIF clk'EVENT AND CLK='1' THEN
 z<=a_z;
 x1<=a_x1;
 END IF;
END PROCESS;
END opcionA_1;

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
 --ENTIDAD
ENTITY sumador_2 IS
 GENERIC (N:INTEGER:=32);
Port( termino1 :IN std_logic_Vector(N-1 DOWNTO 0);
 termino2 :IN std_logic_Vector(N-1 DOWNTO 0);
 salida: out std_logic_Vector(N-1 DOWNTO 0));
END sumador_2;

ARCHITECTURE f_2 OF sumador_2 is
 begin
process(termino1,termino2)
 begin
```

```
 salida<=termino1 + termino2;

end process;
 end f_2;
```