Lightweight PRNG for Low-Cost Passive RFID Security Improvement

By
Joan Melià-Seguí

A Dissertation Submitted in Partial Fulfillment
of the Requirements for the PhD Degree

Advisors:
Dr. Joaquin Garcia-Alfaro
Dr. Jordi Herrera-Joancomartí

April 2011

KISON Research Group
Internet Interdisciplinary Institute
Universitat Oberta de Catalunya

By Joan Melià Seguí

Cover: Berta Mir Daza

2011

Als meus pares, i a la meva germana,
pel seu suport i amor des del primer dia.

Abstract

This dissertation deals with security concerns regarding low-cost radio frequency identification (RFID) communications. RFID systems are composed by tags (also known as electronic labels) storing an identification sequence which can be wirelessly retrieved by an interrogator, and transmitted to the network through middlewares and information systems. Low-cost RFID integrates different technologies, regarding the resource constrained characteristic (thus, reduced cost) of the RFID tags.

The main example of low-cost RFID is the Electronic Product Code Class 1 Generation 2 (EPC Gen2) technology, which is designed to balance cost and functionality. The development of EPC Gen2 tags faces, in fact, several challenging constraints such as cost, compatibility regulations, power consumption, and performance requirements. As a consequence, the computational capabilities of EPC Gen2 tags are very simple. In this sense, the EPC Gen2 specification only considers two basic on board security features: pseudo-random number generators (PRNGs) and password-protected operations. The pseudo-randomness offered by on-board PRNGs is, indeed, used to protect the password-protected operations. PRNGs are also used as an anti-collision mechanism for inventorying processes, and to acknowledge other Gen2 specific operations (e.g., memory writing, decommission of tags, and self-destruction). PRNGs are, therefore, the crucial components that guarantee Gen2 security.

Cryptographic suitable PRNG designs must satisfy unpredictability characteristics. For example, an external adversary who eavesdrops the communication cannot compute the PRNG internal state, even

if many outputs of the generator have been observed. The adversary cannot either compute the next sequence, even if many other previous sequences have been observed. If the adversary can observe, or even manipulate, the input samples that are fed by a PRNG, but its internal state is not known, the adversary must not be able to compute the next sequence. Finally, if the adversary has somehow learned the internal state of the PRNG, but the input samples that are fed in cannot be observed, then the adversary should not figure out the internal state of the PRNG after the re-keying operation. Most of these characteristics are, in fact, required by the EPC Gen2 specification. Hence, the use of weak PRNG designs that allow the predictability of the outgoing sequences introduces important security flaws in EPC Gen2 communications.

This dissertation includes the following points: In the first part, we present the main parameters of the EPC Gen2 technology, for both the communications interface and hardware characteristics. A complete state of the art on PRNGs and stream ciphers is introduced, with special emphasis to RFID and resource-constrained specific designs. The background obtained in this part gives us the framework to focus into the security analysis of RFID based on PRNGs.

The second part of the document deepens on the analysis of PRNGs for RFID. We demonstrate the likelihood to predict a novel PRNG proposal based on a linearity vulnerability, and we also demonstrate statistical deviations on PRNGs from commercial RFID tags. The work presented in this part implies a lack of security in the communications of RFID systems.

In the third part of this dissertation we propose a novel PRNG scheme for RFID, improving the state of the art for resource-constrained low-cost devices. Our proposal solves the linearity problem found in the analysis section, and is designed following the restrictions applying the low-cost RFID technology. A logical description and a hardware implementation are provided to test its suitability to the low-cost RFID technology.

Finally, the fourth and last part of this work presents an evaluation of our proposed PRNG based on four parameters: statistical behavior, security, hardware complexity and power consumption. The main interest is to demonstrate the hardware feasibility of our proposal to

the EPC Gen2 technology, while providing a secure enough communication link.

The contribution of this dissertation is the improvement of the state of the art on security in RFID EPC Gen2 technology. With the knowledge obtained from the analysis of commercial EPC Gen2 tags, and the evaluation of scientific proposals, we have been able to propose a new PRNG design compatible with the hardware and statistical EPC Gen2 requirements, and with improved security properties. We can conclude that our work leads to new design paradigms and recommendations for the security in low-cost RFID, and particularly for the EPC Gen2 technology.

Keywords: Low-Cost RFID, EPC Gen2, PRNG, Lightweight Security, Attack Implementation, Empirical Analysis, Multiple Polynomial, Logical Gates, Power Consumption.

Resum

El treball desenvolupat en aquesta tesi tracta la seguretat en sistemes d'identificació per radiofreqüència (RFID) de baix cost. Els sistemes RFID estan composats per tags (coneguts també com d'etiquetes electròniques) que emmagatzemen un codi d'identificació que es pot obtenir via radiofreqüència per un equip lector, i ser transmesa a la xarxa a través dels sistemes d'informació. L'RFID de baix cost integra diferents tecnologies amb un denominador comú, la limitació de recursos computacionals dels tags (és a dir el seu baix cost).

La tecnologia RFID de baix cost més comuna és l'Electronic Product Code Class 1 Generation 2 (EPC Gen2). El desenvolupament dels tags d'aquesta tecnologia ha de fer front a diverses limitacions, com són el cost de fabricació, compatibilitat amb les regulacions, consum de potència i funcionalitat. Com a conseqüència, les capacitats computacionals dels tags EPC Gen2 són limitades. En aquest sentit, les especificacions de la tecnologia EPC Gen2 només considera dos funcions de seguretat en els tags: generadors de nombres pseudoaleatoris (PRNGs) i operacions protegides amb contrasenya. La pseudoaleatorietat proporcionada als tags s'empra, de fet, per xifrar les operacions protegides amb contrasenya. Els PRNGs també són la base del mecanisme d'anticol·lisió en el procés d'inventariat dels tags, i del mecanisme de comprovació de recepció de certes operacions de l'estàndard EPC Gen2 (per exemple l'escriptura en memòria o la desactivació de tags). Per tant, els PRNGs són els elements bàsics en la seguretat de la tecnologia EPC Gen2.

Les seqüències generades dels PRNG destinats a operacions criptogràfiques no poden ser predictibles. Per exemple, si un adversari

escolta la comunicació no ha de ser capaç de calcular l'estat intern del PRNG, independentment del nombre de seqüències observades. L'adversari tampoc ha de ser capaç de predir la propera seqüència del generador, tot i que s'hagin escoltat les seqüències prèvies. Si l'adversari pot observar, o fins i tot manipular, les seqüències d'entrada al PRNG, però desconeix el seu estat intern, l'adversari no ha de poder predir la següent seqüència. Finalment, si l'adversari ha aconseguit conèixer l'estat intern del PRNG però no té accés a les seqüències d'origen, tampoc podrà calcular les noves seqüències del generador. EPC Gen2 inclou aquestes característiques a les seves especificacions. De fet, la utilització de dissenys de PRNG no adequats per criptografia que permetessin la predicció de les seqüències pseudoaleatòries generaria debilitats en les comunicacions de la tecnologia EPC Gen2.

El treball que recull aquesta tesi es resumeix a continuació: En la primera part, es presenten les principals característiques de la interfície de comunicacions i característiques de hardware de la tecnologia EPC Gen2. També s'inclou un estat de l'art de PRNG i sistemes de xifrat de flux, fent èmfasi en l'àrea dels RFID i dissenys amb recursos limitats. La informació d'aquesta primera part de la tesi proporciona el marc de coneixement per analitzar la seguretat en sistemes RFID basats en PRNGs.

La segona part d'aquest document es centra en l'anàlisi de PRNGs per RFID. Es demostra la possibilitat de predir les seqüències d'una proposta recent de PRNG per una vulnerabilitat deguda a la linealitat del disseny. També s'analitzen tags comercials, detectant desviacions estadístiques en els seus generadors pseudoaleatoris. Els resultats d'aquesta part de la tesi demostren una falta de seguretat en les comunicacions de la tecnologia RFID analitzada.

En la tercera part d'aquest document es proposa un nou disseny de PRNG per RFID, millorant l'estat de l'art per als dispositius de baix cost amb recursos limitats. Aquesta proposta resol els problemes de linealitat detectats prèviament, i en el seu disseny s'han considerat les restriccions inherents a la tecnologia RFID de baix cost. La proposta inclou una descripció lògica i una implementació en hardware per comprovar la seva aplicabilitat a la tecnologia RFID de baix cost.

Finalment, la darrera part d'aquesta tesi presenta una avaluació de la proposta de disseny de PRNG, basada en els següents punts: propietats estadístiques, seguretat, complexitat de hardware i consum de potència. L'objectiu és demostrar l'adequació de la proposta de PRNG a les limitacions de la tecnologia EPC Gen2, proporcionant un nivell de seguretat suficient a les comunicacions.

La contribució d'aquesta tesi és la millora de l'estat de l'art en la seguretat per la tecnologia RFID EPC Gen2. Amb el coneixement obtingut de l'anàlisi de tags comercials, i l'avaluació de propostes de la literatura científica, es proposa un nou disseny de PRNG compatible amb els requisits de la tecnologia EPC Gen2, millorant-ne les propietats de seguretat. Es pot concloure que el treball inclòs en aquesta tesi condueix a un nou paradigma en el disseny i recomanacions per a la seguretat en sistemes RFID de baix cost, i en particular, per la tecnologia EPC Gen2.

Paraules clau: RFID de Baix Cost, EPC Gen2, PRNG, Seguretat Lleugera, Implementació d'Atac, Anàlisi Empírica, Múltiples Polinomis, Portes Lògiques, Consum de Potència.

Acknowledgments

In the following lines I would like to thank those people which, somehow or other, have encouraged and helped me during the realization of this thesis.

En primer lloc, vull donar les gràcies al Dr. Jordi Herrera Joancomartí i al Dr. Joaquin Garcia Alfaro, directors d'aquesta tesi. Gràcies a la seva dedicació, energia i suport per introduir-me en el món de la recerca, i per fer possible aquesta tesi. A ambdós els vull expressar la meva més sincera gratitud. Així mateix, també vull agrair a la resta de membres del grup KISON tota l'ajuda rebuda en aquest període, així com a tot l'equip de l'IN3-UOC amb el seu programa de beques doctorals. Aquesta tesi ha sigut parcialment finançada pel Ministeri Espanyol d'Innovació i Ciència mitjançant els fons FEDER sota els projectes TSI2007-65406-C03-03 E-AEGIS, i CONSOLIDER CSD2007-00004 ARES.

En el desenvolupament d'aquesta tesi també hi tenen molt a veure els meus companys de doctorat de l'IN3, a tots ells gràcies pel fantàstic ambient de feina que he viscut aquests anys. També vull deixar constància del paper determinant que el Dr. Rafael Pous Andrés ha tingut al llarg dels meus períodes universitari i professional.

Des d'un vessant més personal, vull transmetre la meva sincera i profunda estimació a totes aquelles persones que des de Menorca, Barcelona o altres indrets, han compartit estones, i sentiments amb jo, com la meva família de Sant Just, que va acollir-me als primers

anys universitaris com un més de la família, i a na Montse i en Manuel per l'atenció i afecte constant que m'han ofert en la tesi i fora d'ella. No vull deixar de pensar en els meus companys de pis, i tots els meus amics, aquesta tesi és també una mica seva.

Per acabar, aquesta tesi escrita entre Son Ganxo i Barcelona, no hauria estat possible sense els ànims i càlida confiança dels meus pares i la meva germana, els quals m'ho han donat tot, i són el model que he intentat seguir per arribar aquí. Aquesta tesi és una mica el reflex de la vostra constància i dedicació cap a jo. Finalment, he d'agrair de manera molt especial el recolzament i afecte constant de na Berta en aquest llarg viatge. Has viscut els moments bons i també dolents d'aquesta tesi, i de tots n'has sigut el far que ha il·luminat el bon camí. Per tot això i molt més, gràcies.

Contents

List of Figures

List of Tables

List of Acronyms

ASK Amplitude-Shift Keying

DB Data Base

CMOS Complementary Metal-Oxide Semiconductor

CRC Cyclic Redundancy Code

DoS Denial of Service

EEPROM Electrically Erasable Programmable Read-Only Memory

EPC Electronic Product Code

ETSI European Telecommunications Standards Institute

FF Flip-flop

FPGA Field-Programmable Gate Array

FSPL Free-Space Path Loss

GE Gate Equivalent

Gen2 Generation 2

HF High Frequency

IC Integrated Circuit

IEC International Electrotechnical Commission

ITF Interrogator Talks First

ISO International Organization for Standardization

ISP In-System Programming

JTAG Joint Test Action Group

LCG Lineal Congruential Generator

LFSR Linear Feedback Shift Register

LSB Less Significant Bit

MHz Mega Hertzs

MOSFET Metal-Oxide Semiconductor Field-Effect Transistor

MSB Most Significant Bit

NIST National Institute for Standards and Technology

NVM Non-Volatile Memory

PC Protocol Control

PCB Printed Circuit Board

PRNG Pseudo Random Number Generator

PSK Phase-Shift Keying

QoS Quality of Service

RFID Radio Frequency Identification

RN16 16-bit Random Number

RNG Random Number Generator

ROM Read-Only Memory

RS-232 Recommended Standard 232

SPICE Simulation Program with Integrated Circuit Emphasis

UART Universal Asynchronous Receiver / Transmitter

UHF Ultra High Frequency

USB Universal Serial Bus

VLSI Very Large-Scale-Integration

XOR Exclusive OR

1

Introduction

Radio Frequency Identification (RFID) technology is an automatic identification method for retrieving digital information without physical contact or line-of-sight, that is revolutionizing the manner in which objects and people can be identified by computers [16]. Tagging objects or even people with smart labels (the so called RFID tags) emitting identifying information in form of binary modulated signal, is the way computers can actually understand the presence of objects. RFID technology is the closest approach to the ubiquitous computing [84] or the future *Internet of Things*.

RFID labels are frequently referred as the next generation barcodes. Although the utility is the same (the identification of an object), RFID offers two main advantages over conventional barcode systems. On the one hand, optical barcodes only indicates the generic product, whereas a RFID tag can identify the item (being able to distinguish

different objects from the same product). On the other hand, there is no need of line-of-sight. Thus, while optical barcodes must be identified one by one, RFID tags can be read much faster, without human intervention and in large quantities [16, 48].

The unassisted wireless identification makes the RFID very attractive in areas like product traceability, inventorying or personal identification, but it also creates setbacks. Like the rest of wireless information technologies, RFID information transferred between sender and receiver is not completely secure. The air interface is much more insecure than the wired one, because the only presence of an attacker in the communication area gives him the opportunity to obtain information in a malicious way. The scarce available energy on tags, and tag computational capabilities are also determinant for security in RFID.

In addition, RFID is very related with personal identification. Let us imagine a postal user sending a RFID enabled letter, with some easy techniques. It will not be difficult to link sender and receiver by eavesdropping the letter identification. Privacy issues must, therefore, also be considered.

1.1 Motivation and Research Objectives

This research dissertation is focused in low-cost passive RFID. This is the case of the Electronic Product Code Class 1 Generation 2 (EPC Gen2) [25] for UHF, designed by EPCglobal [26] and developed in the MIT Auto-ID labs. This technology is being widespread in the retail industry [64], and also other sectors [78], thanks to the reduced price of their tags. EPC Gen2 was designed giving priority to reduce the price by means of a very simple performance [48].

Indeed, the price is the main reason for the industry to adopt or to refuse a technology. It is not a coincidence that EPC technology appearance coincided with the explosion of RFID adoption in the retail industry [86], because tag price should not increase the product cost [48]. It can be said that a small area chip (thus a few logical gates) and no battery on-board (thus using radio frequency waves to energize the tag) will be a cheap tag. But that also means that there is almost no place for additional capabilities in the chip like security mechanisms. In fact, security measures implemented on those devices are scarce and are basically reduced to the use of Pseudo-Random Number Generators (PRNG) and small passwords [16].

Providing security for RFID systems is not a simple task. Established approaches for protecting communications cannot be applied without modification of the specific protocols because of the special characteristics of RFID systems [29]. The singularity of RFID systems lies in the properties of communication, which makes the communication channel (the air-interface) very susceptible to security threats. Specially for EPC Gen2 systems due to its simplicity and specific differences in the link's properties (forward and backward channel).

Additionally, in the low-cost RFID there is a big difference in the computing capabilities of readers and tags, what makes the implementation of security protocols challenging. Computation of cryptographic primitives is generally much more computationally expensive compared to the simple tasks of a simple tag. Hence, security measures have to be designed considering the special requirements of RFID systems into account. These low-cost tags are expected to be deployed on a large scale. Hence, there is a potential danger of obtaining sensible information from the tags. For example an adversary can link a specific tag identification with the price of the object, thus, prioritize actions regarding the economic target. Furthermore, pri-

vate information about people can be obtained by stealthily scanning the tags that they carry [16].

We focus our research in PRNGs as main security tool for low-cost RFID. The use of weak PRNGs that allow the predictability of the outgoing sequences introduce important security flaws in any communication system. For example, it might allow an adversary to bypass the security of the password-protected commands defined in the EPC Gen2 standard (e.g., the *access* and the *kill* commands). This is indeed possible if the on board tag's PRNG is predictable, e.g. due to its bad statistical properties, since it then suffices to apply a simple XOR operation with the predicted sequences and the contents of the messages transmitted over the reader-to-tag channel to decrypt the remainder ciphertext (e.g., the protected password values).

Our motivation is to improve the security state-of-the-art in low-cost passive RFID by the analysis and improvement of pseudo-random number generators. PRNGs are necessary in the current version of the most extended low-cost RFID technology: the EPC Gen2 Standard. But PRNG can also be used as a core elements for future cryptographic primitives implementation on RFID tags. Thus cryptographically secure PRNGs are necessary for the current and future technology. Our goal is to design a suitable PRNG to be adaptable to the possibilities of the technology, offering the secure behavior while not exceeding the low-cost requirements.

1.2 Main Contribution of the Dissertation

To summarize, the main contributions of this dissertation can be concentrated in four main points:

Attack to EPC Gen2 PRNG proposal: We have demonstrated the vulnerability of the EPC Gen2 PRNG proposed by Che *et al.* [18]. The vulnerability is related to the inherent linearity of linear feedback shift registers, which is translated into a predictability threat. The Che *et al.* scheme is theoretically predictable after only 128 bits with a confidence of 42%, as demonstrated in [60]. The attack was proved to be implementable in EPC Gen2 compatible devices, using a novel method for eavesdropping pseudo-random sequences from EPC Gen2 tags presented in [59].

Analysis of EPC Gen2 commercial PRNGs: Additionally to the study of proposals in the scientific literature, we have analyzed the pseudo-random sequences generated by ICs attached to commercial tags. Using statistical testing, we have found evidences of non-randomness in the analyzed sequences. Furthermore, we propose a different approach to measure the sequences' randomness quality, based on the frequency of the sequences regarding the size of the dataset.

New PRNG proposal based on multiple-polynomial: We have proposed a new PRNG design based on a linear feedback shift register (LFSR) for security applications, compliant with the EPC Gen2 technology. Our proposal handles the LFSRs inherent linearity with the multiple-polynomial architecture, being the first time it is used for security applications. The PRNG proposal has been described at a logical level and also following a hardware scheme using registers, logic gates and non-volatile memory. Due to the severe power and area restrictions in the EPC Gen2 tags, an evaluation of primitive polynomials of degree 16 with suitable characteristics for its hardware implementation, is also presented in this dissertation. A preliminary version of the proposal was presented in [31].

Evaluation of our PRNG proposal based on EPC Gen2 requirements: Based on our PRNG proposal, a complete evaluation has been performed. Regarding that our PRNG is intended to work as a cryptographic tool in a resource-restricted device, the aim of this evaluation is to demonstrate the suitability of our proposal to the EPC Gen2 technology. The evaluation includes statistical and formal security analysis, hardware complexity (area) regarding similar proposals presented in the literature, the power consumption of the tag's digital scheme based on a SPICE simulation, and finally a time execution evaluation.

1.3 Document Organization

The rest of this dissertation is organized as follows:

Chapter 2 is focused in the EPC Gen2 technology characteristics, specially focused on the EPC Gen2 tags on-board main security tool: the PRNG. Furthermore, different security mechanisms based on stream ciphers for constrained devices are introduced, as well as scientific proposals for PRNGs designed for low-cost RFID. We also review the evaluation tools used in this dissertation, including a testing device prototype and statistical evaluation tests.

Chapter 3 presents a practical implementation attack on a weak PRNG designed specifically for EPC Gen2 tags. We demonstrate that it is feasible to eavesdrop a small amount of pseudo-random values by using standard EPC commands and using them to determine the PRNG configuration that allows to predict the complete output sequence in a theoretical way, and performing a real attack with EPC Gen2 compatible devices. Also in this chapter, we present an analysis of the pseudo-random sequences generated from real commercial

tag integrated circuits (ICs). Specifically, ICs from vendors such as *NXP*, *Alien* and *Impinj* are tested. The main analyzed parameter is the probability of appearance of any single 16-bit pseudo-random sequence, evaluating the statistical properties of the commercial ICs PRNG.

Chapter 4 proposes a new pseudo-random number generator design for EPC Gen2. This generator is based on a 16-bit linear feedback shift register with multiple feedback primitive polynomials fed by a physical source of randomness. The proposed PRNG successfully handles the inherent linearity of linear feedback shift register based PRNGs and satisfies the hardware requirements imposed by the UHF RFID technology. This chapter includes a high level logical description of the method used to obtain the randomness on the tag, and a preliminary hardware specification implementing the designed technique.

Chapter 5 evaluates the suitability of our design to the specifications required for the EPC Gen2 standard for RFID. Statistical analysis of the pseudo-random sequences from our generator confirms the validity of the proposed technique. Furthermore, a formal security analysis is presented, evaluating the PRNG security using attack scenarios with different capabilities. Regarding the PRNG power consumption, an electronic circuit simulation of the proposed hardware is presented, as well as an execution time analysis evaluation according to the current technology implementing EPC tags.

Finally, concluding remarks and a discussion about the possible future research directions to follow are given in Chapter 6.

2

State of Art

Low-cost radio frequency identification (RFID) tags are becoming a successful technology to increase the efficiency and productivity in the logistics sector. As the costs of tags are dropping, logistics departments are taking more attention to the possibility to integrate this real-time technology in the business processes in order to improve the visibility and accuracy of the logistic operations [64].

The deployment of the RFID technology is becoming more important thanks to the standardization process through the Electronic Product Code (EPC) Class 1 Generation 2 (hereinafter denoted as Gen2) tag standard [25] promoted by EPCglobal. EPCglobal standardization covers the whole RFID architecture, from tag data structure to network communication specifications. EPC tags are not provided with on-board batteries, but are passively powered through radio-frequency waves.

The remainder of this chapter is organized as follows: Section 2.1 introduces the EPC Gen2 technology characteristics, specially focused on the EPC Gen2 tags on-board main security tool: the *pseudo-random number generator* (PRNG). Section 2.2 introduces different security mechanisms based on stream ciphers for constrained devices, and Section 2.3 overviews scientific proposals for PRNGs designed for low-cost RFID. Finally, Section 2.4 introduces the evaluation tools used in this dissertation, including a testing device prototype and statistical evaluation tests.

2.1 The EPC Gen2 Standard

The EPC technology is based on the use of RFID. This technology is intended to be the successor of the nowadays ubiquitous barcodes. Designed in the Massachusetts Institute of Technology Auto-ID Labs, and developed by the EPCGlobal consortium [26], the EPC technology represents the key component of an architecture known as EPCglobal Network [25]. The main components of the RFID system are the electronic labels or tags, the readers and the Information Systems (IS) e.g middlewares, databases and servers. The main goal of this architecture is the object-in-motion automatic identification in the supply chain and factory production.

The EPC tags (cf. Table 2.1) are passive devices powered by the electronic field generated by the reader, due to the absence of on-board batteries. They work worldwide on the ultra high frequency (UHF) band between 860 and 960 MHz, depending on the RF regulations for each continent. The communication range between tags and readers depends on the electric field, thus, it may vary depending on the power supply and antenna design, but also on the kind of sur-

face where the tag is placed. RFID tags are intended to be deployed widely so they must be cheap. EPC Gen2 Tags are composed by two main elements, the *Integrated Circuit* (IC) and the *antenna*. The IC is based on a state machine model and processes and stores the RFID information. The antenna is intended to receive and transmit RFID signals, and also to energize the IC. In a low-cost RFID system, like EPC Gen2, the tags are very simple and resource limited, allowing to reduce their cost under the 10 cents of US dollar [89]. This reduction on the tag cost is proportional to the size of the silicon IC. The typical measure of space in silicon ICs is the *gate equivalent* (GE) that is equivalent to a boolean *two-input NAND gate*. The estimations on available GE for EPC Gen2 implementations are around 10,000 GE [85].

The EPC Gen2 system communication model is common to other low-cost RFID systems where the interrogator (reader) talks first (ITF). EPC Gen2 tags are passive and power dependent from the reader to respond the queries. The communication between tag and reader in the EPC Gen2 system is organized in three stages. In the *Selection* and *Inventorying* stages, the reader initiates the communi-

Table 2.1: EPC Gen2 tags main properties

Identification	96 bit
Communication range	$\sim$ 5 m
Tag power consumption	$\sim$ 10 μW
Frequency (Europe)	865-868 MHz (UHF)
Tags Tx ratio	40 - 640 kbps
Tags Rx ratio	26.7 - 128 kbps
Identifications per second	$\sim$ 200

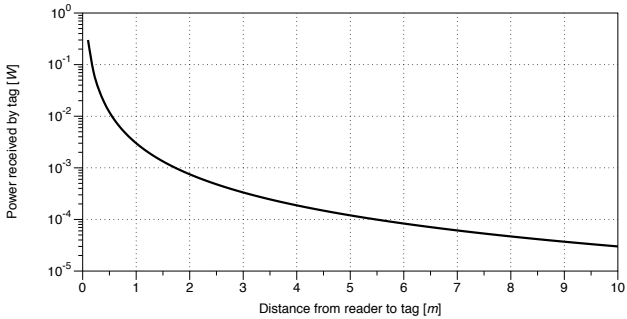


Figure 2.1: At five meters an EPC Gen2 tag receives around 100 μW from the reader

cation sending identification queries. The available tags in the communication range respond with a 16-bit provisional identifier (hereinafter denoted as RN16) extracted from the implemented PRNG on-board. When the reader acknowledges the provisional identifier, each single tag sends an identification sequence. The EPC Gen2 standard defines the identification sequence with 96 bits [25], but other identification sizes can be used depending on the tag manufacturer. If the reader manages to access or modify the tag memory content at this point, the *Access* stage is started. In the remainder of this section we introduce the main properties of the EPC Gen2 technology used in this dissertation.

2.1.1 Main Properties

EPC Gen2 tags do not have a power source. Instead, tags are passively powered following an ITF protocol, thus, tags can only respond after a message is sent by the reader. Regarding the physical layer, the reader powers up the tag by transmitting a radio frequency (RF) continuous wave to the tag, and the tag backscatters a signal to the

reader using the modulation of the reflection coefficient of its antenna. RFID passive tags are powered through the electromagnetic waves received from the interrogator. Only a small fraction of the power emitted by the interrogator is received by the RFID tag antenna, inducing a voltage to the RFID tag IC. The European Telecommunications Standards Institute (ETSI) [45] regulates the RF spectrum for the European region. It allows for the RFID UHF communication a maximum transmission power of 2 W from EPC Gen2 readers. According to the *Friis transmission equation* (cf. Equation 2.1) [79], the signal power received by a RFID tag IC depends on the power signal from the reader, the gain of the antennas of both tag and reader and the inverse of the free-space path loss (FSPL) equation.

$$P_{RX,tag} = P_{TX,reader} G_{reader} G_{tag} \left(\frac{\lambda}{4\pi d} \right)^2 \tag{2.1}$$

The FSPL for the UHF frequency, which in the equation is represented by its wavelength (λ), decline quadratically (order of magnitude) with the distance (d) to the interrogator antenna. The communication distance d for the RFID tags depends on the factors included in the Equation 2.1 and it is usually considered of about 5 meters, that is, the maximum distance where the signal power is sufficient to activate the tag IC. Figure 2.1 shows the approximated tag received power curve depending on the distance between reader and tag. This distance is considered in ideal conditions but on real RF environments there are mitigation factors reducing such distance. Signal reflection, absorbing materials or inadequate antenna orientation are possible factors for reducing the communication distance. The communication is half-duplex thus, simultaneous transmission and reception is not allowed.

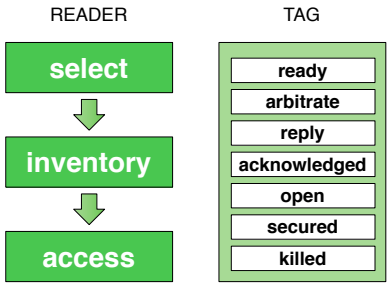


Figure 2.2: Reader stages and tag states for the EPC Gen2 protocol

The communication stages in the EPC Gen2 protocol are organized in three stages (cf. Figure 2.2):

- **Select**: In this stage, the reader selects a subset of the tag population in the communication range for inventory and access using one or more *Select* commands.
- **Inventory**: The process by which a reader identifies tags. An inventory round is initialized by the reader sending *Query* commands. One or more tags may reply, thus, the tags use an anti-collision protocol to avoid collisions. After *selection* the tag loads a random slot counter between *zero* and $2^Q - 1$ (with $0 \leq Q \leq 15$, automatically adjusted or user-defined) decreasing one unit for each *Query* command reception. When the counter reaches the value *zero*, the tag initiates the reply. If the reader detects a single tag reply, it requests the identification from the tag. Figure 2.3 shows an example of a reader inventorying a single tag.

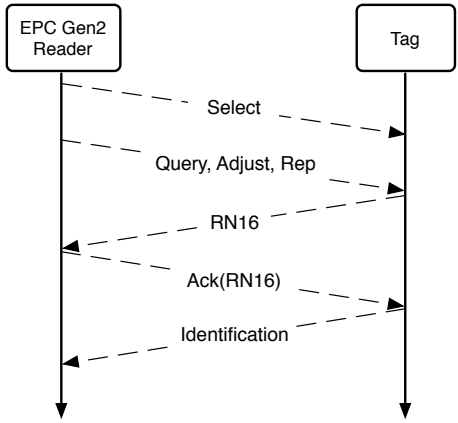


Figure 2.3: Example of *Select* and *Inventory* process

- **Access**: The process by which a reader modifies or reads individual tags' memory areas. This stage can only be initiated after a successful inventory process.

The tag memory is logically divided into four banks (cf. Table 2.2):

- **Reserved**: This memory block shall contain the 32-bit access and kill passwords. If these passwords are not specified, a logic *zero* is stored on that memory area. Tags with a *non zero* access password have to receive that value before transitioning to a secure state.
- **EPC**: This block contains the Protocol Control (PC) bits and the 96-bit identification code (denoted as EPC) that identifies the tag. This memory block also contain a CRC-16 (defined in ISO/IEC 13,239) checksum of the PC and EPC codes.

- **TID**: This area of memory shall contains an 8-bit ISO/IEC 15,693 class identifier. Moreover, sufficient information to identify the custom commands and optional features supported by the tag is also specified in this memory block.

- **User**: This memory block is not mandatory thus, the block size is not specified in the standard. Instead, the User memory is factory-configured depending on the manufacturer.

The communication between reader and tags in the EPC Gen2 protocol is organized in identification stages and tag states. The following paragraphs describe each of the possible tag states (cf. Figure 2.4):

Table 2.2: EPC Gen2 tag's memory logic map

User:	Optional
TID:	TID [15:0]
	TID [31:16]
EPC:	XPC_W1 [15:0]
	EPC [15:0]
	⋮
	EPC [95:79]
	PC [15:0]
	CRC [15:0]
Reserved:	Access Password [15:0]
	Access Password [31:16]
	Kill Password [15:0]
	Kill Password [31:16]

- **Ready**: After being energized, a tag that is not killed enters in the *ready* state. The tag shall remain in this *ready* state until it receives a *Query* command. Tag loads a Q-bit number from its PRNG, and transitions to the *arbitrate* state if the number is *non-zero*, or to the *reply* state if the number is *zero*.

- **Arbitrate**: A tag in an *arbitrate* state shall decrement its slot counter every time it receives a *QueryRep*, transitioning to the *reply* state and backscattering a RN16 when its slot counter reaches *zero*.

- **Reply**: A tag shall backscatter a RN16, once entering in *reply* state.

- **Acknowledged**: If a tag in the *reply* state receives a valid acknowledge (*Ack*), it shall transition to *acknowledge* state, backscattering its PC, EPC, and CRC-16. Otherwise, the tag returns to the *arbitrate* state.

- **Open**: After receiving a *Req_RN* command, a tag in *acknowledge* state whose access password is *non-zero* shall transition to *open* state. The tag backscatters a new RN16 that both reader and tag shall use in subsequent messages. Tags in an *open* state can execute all access commands except *Lock* and may transition to any state except *acknowledge*.

- **Secured**: A tag in *acknowledge* state, which access password is *zero*, shall transition to *secured* state, upon receiving a *Req_RN* command. The tag backscatters a new RN16 that both reader and tag shall use in future messages. A tag in the *open* state, which access password is *non-zero*, shall transition to a *secured* state, after receiving a valid access command, which includes the same *handle* that was previously backscattered when it

transitioned from *acknowledge* state to the *open* state. Tags in *secured* state can execute all access commands and may transition to any state except *open* or *acknowledge*.

- **Killed**: Once a *kill* password is received by a tag in either *open* state or *secured* state, it shall enter the *killed* state. *Kill* permanently disables a tag. A tag shall notify the reader that the killed operation was successful, and shall not respond to any reader thereafter.

2.1.2 Security in the EPC Gen2 Standard

As in many other emerging technologies, attacks against different services of the EPC architecture may expose its users to security risks and privacy violations. Indeed, if countermeasures are not handled properly at the lowest level of the architecture, where the exchange of information between Gen2 tags and readers is carried via insecure wireless connections, attacks such as data disclosure, cloning, impersonation, and denial of service may succeed. However, the development of Gen2 security countermeasures faces several challenging constraints such as cost, compatibility regulations, power consumption, and performance requirements. From the approximately 10,000 available GE in EPC Gen2 ICs only 2,000 to 5,000 can be devoted to security tasks [85]. Thus, on-board security tools must be necessarily simple.

The EPC Gen2 communication protocol includes basic security mechanisms for the *Access* stage in the tags. The EPC Gen2 standard includes in its specification a 32-bit password to protect the tag memory access. Moreover, the standard includes a 32-bit password for the *kill*

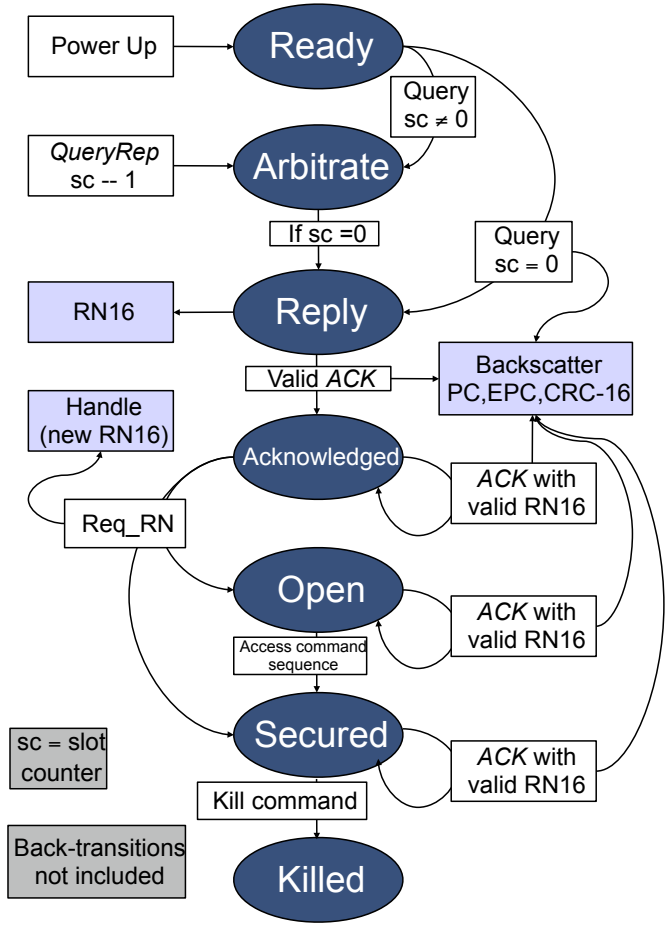


Figure 2.4: EPC Gen2 tag state diagram extracted from [25]

command execution. This command allows to permanently deactivate the tag performance (*kill*), or to unblock specific tag memory areas previously blocked (*recomission*), depending on the command

codification [25]. The kill and access passwords are stored in the tag reserved memory area (Table 2.2 describes the EPC Gen2 tags different memory areas).

To avoid revealing sensible information in the *reader-to-tag* channel (e.g. passwords or new identifiers) susceptible to be eavesdropped from a non authorized reader, the EPC Gen2 tags include a PRNG to encrypt the transmitted information on that channel. Hence, when the EPC reader requests the *Access* to a tag, the tag sends 16-bit nonces in plaintext to encrypt the content to be sent by the reader, by means of a bitwise exclusive OR (XOR) operation.

To execute the kill command to an EPC Gen2 tag for example, the reader shall previously identify it. Once the tag has sent the 96-bit identification, the reader switches to the *Access* stage (cf. Figure 2.5). To successfully switch to the *Access* stage the reader requests a 16-bit nonce to the tag to be used as a session key for the *Access* stage. When the tag provides the RN16 (represented as *Handle*), the reader requests a new nonce (RN16') to start the *kill* password encryption. This process is repeated for the two halves of the kill password (Pwd [31:16] and Pwd [15:0]) with a new RN16" key. To confirm the operation success the tag sends a last message containing a header and the *Handle* code.

Security paradigm

Traditional keystream generators share a secret k as a key for the PRNG *one-time pad* communication between *sender* and *receiver*. The use of deterministic PRNGs to generate exactly the same sequence in both sender and receiver sides is commonly used.

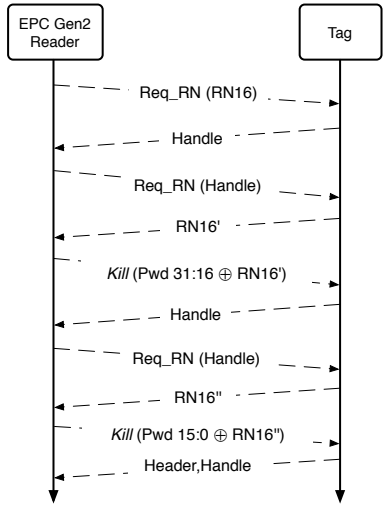


Figure 2.5: EPC Gen2 *kill* command protocol

However, the specific communication model of EPC Gen2 systems uses another paradigm in which *sender* and *receiver* cannot share any secret k. Instead, the low-power tag-to-reader communication is used to transmit in plaintext the nonces to be used as a keystream for the reader-to-tag communication. The next paragraphs detail the keystream generator characteristics of the EPC Gen2 technology.

On-board PRNG

Existing commercial EPC Gen2 tags implement an on-board PRNG, as required by the EPC specifications [25]. However, companies are reluctant to provide their designs [75]. Manufacturers simply refer to testbeds that show the accomplishment of some compatibility requirements. They fail to offer convincing information about the security

of their designs. This is mostly security through obscurity, which is always ineffective in security engineering, as it has been shown with the disclosure of the PRNG used in the MIFARE Classic chip [30] that has shown a vulnerable PRNG.

The on-board PRNG generates 16-bit pseudo-random sequences, and shall have the ability to provide RN16s to the Q anti-collision system, to acknowledge other Gen2 specific operations (e.g., memory writing, decommission of tags, and self-destruction), and as a source for the *one-time-pad* based *cover-coding* ciphering system for the access stage of the tag identification. PRNGs are, therefore, the crucial components that guarantee Gen2 security. As specified in the standard the PRNG is supposed to meet the following randomness criteria:

1. The probability that any single 16-bit sequence j drawn from the generator shall be bounded by $\frac{0.8}{2^{16}} < Prob(j) < \frac{1.25}{2^{16}}$.

2. Among a tag population of up to ten thousand tags, the probability that any two tags simultaneously generate the same 16-bit sequence shall be less than 0.1%.

3. The chance of guessing the next 16-bit sequence generated by a tag shall be less than 0.025% even if all previous outputs are known to an adversary.

2.2 Symmetric Cryptography Based on Stream Ciphers

In this section, we introduce an overview of cryptographic primitives with suitable characteristics for its implementation on RFID devices.

Section 2.3 introduces specific security proposals built from the cryptographic primitives shown in this section.

2.2.1 Cryptography Goals

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication. The application of cryptographic primitives in information systems is based on the following goals [61]:

- *Confidentiality* is used to keep information accessible only to the authorized entities. Secrecy is a term synonymous with confidentiality and privacy. There are numerous approaches to provide confidentiality, from physical protection to mathematical algorithms which render data unintelligible.

- *Data integrity* addresses the unauthorized alteration of data. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties. Data manipulation includes insertion, deletion, and substitution.

- *Authentication* is closely related to identification. This function applies to both entities and information itself. Two parties entering into a communication should identify each other. Information delivered over a channel should be authenticated as to origin, date of origin, data content, time sent, etc. Data origin authentication implicitly provides data integrity.

- *Non-repudiation* prevents an entity from denying previous actions. When disputes arise due to an entity denying that certain

actions were taken, a means to resolve the situation is necessary. For example, one entity may authorize the purchase of property by another entity and later deny such authorization was granted.

2.2.2 Symmetric Cryptography Primitives

Cryptosystems are divided between *secret-key* or *symmetric*, *public-key* or *asymmetric* and *unkeyed* systems. Asymmetric cryptography is built around mathematical hard problems [61] such as factorization, discrete logarithms or elliptic curves. Some approaches to public-key cryptography for RFID based on re-encryption and Elliptic Curve Cryptography are detailed in [49, 11]. Nevertheless, current EPC Gen2 tag computation capabilities are unsuited to asymmetric cryptography deployment [27]. The unkeyed cryptography main example is the *Hash* functions. The cost of these hash functions in hardware implementation is considered expensive in the literature [27]. Due to the excessive consumption of hash functions regarding computational power and memory resources it is recommended to minimize its use in RFID implementations and consider alternative security tools [29].

Symmetric cryptography is based on the secret-key exchange between participants. There are two distinct types of symmetric encryption; *stream ciphers* and *block ciphers* and their functionality is described as follows [72]:

- A *block cipher* transforms blocks of plaintext into ciphertext under the action of a key. This is typically a relatively complicated transformation but, apart from the reused key, the encryption of one block is independent of another.

- A *stream cipher* generates a keystream by sampling a constantly evolving cipher state. The state is typically initialized under the action of a key and an initialization vector. The sampling operation and the operation used to update the state are usually computationally lightweight. The plaintext stream is then encrypted by combining it directly (typically using bitwise XOR) with the keystream to give the ciphertext stream.

The Data Encryption Standard (DES) algorithm has been the most important block cipher for decades. Due to its limited key size (56-bit) this algorithm was extended with the Triple-DES and now it has been replaced by the Advanced Encryption Standard (AES) algorithm which uses 128, 192 and 256-bit keys. The AES algorithm was standardized by the NIST [4] in 2001. Other interesting block ciphers for resource constrained devices are the Tiny Encryption Algorithm (TEA) [99] and the Present Algorithm [13]. The latter is currently the most hardware efficient block cipher with 1,570 GE. TEA, DES and AES block ciphers can be implemented from around 2,000 GE [72].

Together with the block cipher, symmetric cryptography includes the stream cipher. The remainder of this section deepens into this category of ciphers and the keystream generators used by the stream ciphers, usually pseudo-random number generators.

2.2.3 Pseudo-Random Number Generators

The security of many cryptographic systems depends upon the generation of unpredictable data sequences [61], for example the keystream in the *one-time-pad* or the challenges used in challenge-response identification systems. In any case, the generated sequences must be of

sufficient size and be random in the sense that any generated sequence must have the same probability of appearance, to avoid the possibility of a search strategy based on such probability from an adversary.

A random number generator (RNG) is a device or algorithm which outputs a sequence of statistically independent and unbiased binary digits. However RNG is an inefficient procedure in most practical environments [61]. A PRNG is a deterministic algorithm which given an initial state (or seed) of length k, outputs a binary sequence of length $l \gg k$ which looks like a random sequence. It is usually a deterministic one-way algorithm that generates sequences which statistical properties are indistinguishable from truly random numbers, but its output is often predictable [16]. PRNGs are characterized by its period, lineal complexity and implementability, and must fulfill the three Golomb's postulates [87, 36]:

1. The number of 1's in every period must differ from the number of 0's by no more than one.

2. In every period, half the runs must have length one, one quarter must have length two, one eighth must have length three ... as long as the number of runs so indicated exceeds one. Moreover, for each of these lengths, there must be just as many runs of 1's and 0's.

3. Suppose we have two copies of the same sequence of period p which are off-set by some amount d. Then for each d, $0 \leq d \leq p-1$ we can count the number of agreements between the two sequences, A_d, and the number of disagreements, D_d. Then, the auto-correlation coefficient for each d, defined by $(A_d - D_d)/p$, must be a bi-valuated function.

Many cryptographic algorithms and protocols require the generation of random values such as keys (or keystreams) and nonces, hence, the generated random sequences need to be unpredictable in practice. As specified in the EPC Gen2 Standard the tags should include a PRNG, whose main function is the generation of RN16s to be used as one-time pad nonce for data encryption.

A PRNG functionality is typically based on some unknown secret to the adversary, for example the generator internal state. The PRNG computes the next output sequence as a one-way function of its internal state, adapting the internal state in a deterministic manner. Good PRNGs are designed in such a way that they satisfy the following properties [16]:

- The adversary cannot compute the internal state of the PRNG, even if many outputs of the PRNG have been observed.
- The adversary cannot compute the next output of the PRNG, even if many previous outputs of the PRNG have been observed.
- If the adversary can observe or even manipulate the input samples that are fed in the PRNG, but the internal state of the PRNG is not known, then the adversary cannot compute the next output and the next internal state of the PRNG.
- If the adversary has somehow learned the internal state of the PRNG, but the input samples that are fed in the PRNG cannot be observed, then the adversary cannot figure out the internal state of the PRNG after the re-keying operation.

PRNGs play an important role when trying to encrypt plaintext messages in an information constrained environment. PRNGs are generally fast and have a simply hardware circuitry, solving the absence

of buffering and being suitable for highly probable error-transmission channels [61]. Next, two common PRNGs are introduced: the Lineal Congruential Generator (LCG) and the Linear Feedback Shift Register (LFSR).

Lineal Congruential Generator

Lineal Congruential Generators (LCG) presented by Lehmer in 1951 [54] are pseudo-random number generators defined by equation 2.2, in which X_i is the i_{th} number of the sequence, and X_{i-1} is the previous number of the sequence. Variables a, b and m are constants and X_0 is the seed or key of the system.

$$X_i = (aX_{i-1} + b) \mod m \tag{2.2}$$

If a, b and m are properly chosen, then the generator will be a maximal period generator and the output of the sequence will have period of m [90]. Regardless of its good statistical properties and easy implementation in software, LCGs are easily predictable (even if parameters a, b and m are unknown) because only a few integers produced by the LCG are necessary to predict the remainder of the sequence [77]. Hence, LCGs are not suitable for cryptographic applications [73].

Linear Feedback Shift Registers

A linear feedback shift register (LFSR) (cf. Figure 2.6) is a digital circuit of n bits (or cells) that contains a shift register and a feedback function. The shift register is composed of a sequence of binary cells that share the same clock signal. Each time a bit is needed, the con-

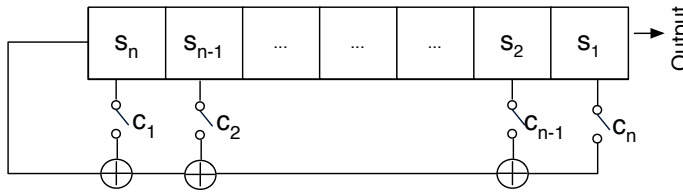


Figure 2.6: Linear feedback shift register scheme

tent of the register is shifted one cell, obtaining the most significant bit of the register in the previous state, given by the expression:

$$s_{n+1} = c_1 s_n + \cdots + c_n s_1 \tag{2.3}$$

The feedback function computes a new bit using the current state of the register, obtaining the less significant bit to be filled in the new state of the register. The feedback function of a LFSR can be represented as a polynomial function of degree n:

$$C(x) = 1 + c_1 x^1 + c_2 x^2 + \cdots + c_n x^n \tag{2.4}$$

The feedback polynomial function is defined by the value of the coefficients $c_i \in \{0, 1\}$. If $c_i = 1$, the content of the cell $n - i$ (named *tap*) is used to compute the new most significant bit of the register. Tap's content is processed with an exclusive OR logical operation (XOR).

The LFSR can then be determined by this polynomial function. In turn, the sequences of the LFSR can be determined by the polynomial function of the LFSR and the initial state of the register cells (often referred as *seed*). Equation 2.5 shows the mathematical expression of the LFSR where n is the number of cells of the LFSR, i is the

position in the register and k is the number of shifts. It is important to notice that an initial state with zeros in all the cells leads to an absorbing state, generating zeros for the following shifts of the LFSR.

$$s_{n+k} = \sum_{i=1}^{n} c_i s_{n+k-i} \tag{2.5}$$

The period (quantity of different possible states) of a LFSR with n cells is up to $2^n - 1$ when taps configuration follows a primitive polynomial function, generating sequences with optimum statistical properties. A primitive polynomial is a polynomial that generates all elements of an extension field from a base field. Primitive polynomials are also irreducible polynomials. For example, the number of primitive polynomials of degree n over the finite field GF(2) (i.e., with coefficients either 0 or 1) is defined by the Equation 2.6, where $\phi(n)$ is the *totient* function [12].

$$a_2(n) = \frac{\phi(2^n - 1)}{n} \tag{2.6}$$

LFSRs are the most common type of shift registers used in cryptography. They lead to efficient and simple hardware implementations. They have, however, important drawbacks that must be handled. First, the sequences of a LFSR are predictable [41, 19]. For example, let s_{k+1}, $s_{k+2}, \cdots, s_{k+2n}$ be a sequence of $2n$ consecutive bits generated from a LFSR. Let c_n, $c_{n-1}, \cdots, c_1$ be the feedback function of the LFSR. Then, the feedback function can be easily computed by solving the following equation system:

$$\begin{bmatrix} s_{k+1} & s_{k+2} & \cdots & s_{k+n} \\ s_{k+2} & s_{k+3} & \cdots & s_{k+n+1} \\ \vdots & \vdots & \ddots & \vdots \\ s_{k+n} & s_{k+n+1} & \cdots & s_{k+2n-1} \end{bmatrix} \begin{bmatrix} c_n \\ c_{n-1} \\ \vdots \\ c_1 \end{bmatrix} = \begin{bmatrix} s_{k+n+1} \\ s_{k+n+2} \\ \vdots \\ s_{k+2n} \end{bmatrix} \tag{2.7}$$

By solving Equation 2.7 we obtain the feedback polynomial coefficients. Therefore, a n-bit (cells) LFSR with period $2^n - 1$ can be determined with only $2n$ values. The *Berlekamp-Massey* [57] algorithm can also be used to solve the feedback coefficients of a LFSR with only $2n$ values. Since any periodic sequence can be generated with a LFSR, the *linear complexity* of a sequence is defined as the shortest number of LFSR cells that can generate the sequence. In other words, if we define $L(s)$ as the linear complexity of an infinite binary sequence s, then [61]:

- if s is the zero sequence $s = 0, 0, 0, \ldots$ then $L(s) = 0$
- if no LFSR generates s, then $L(s) = \infty$
- otherwise, $L(s)$ is the length of the shortest LFSR that generates s.

This linearity problem must be handled before using LFSRs to build PRNGs.

2.2.4 Stream Ciphers

Stream ciphers (cf. Figure 2.7) are an important class of encryption algorithms. They encrypt individual values (usually bits) of a plaintext message one at a time, using an encryption transformation

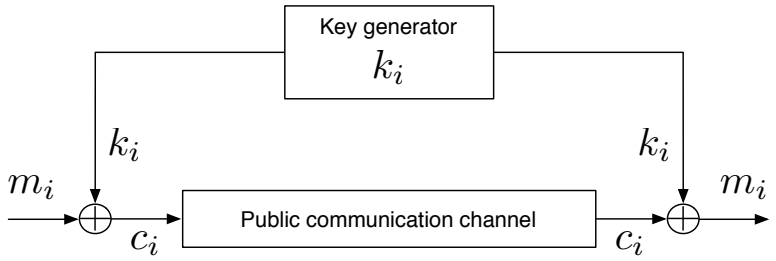


Figure 2.7: Working principle of stream cipher

which varies with time [61]. The stream cipher algorithms are the most simple symmetric cryptographic primitives. The encryption is applied to a single bit for each cycle (usually applying a XOR operation) with a key stream that depends on the current internal state. Pseudo-random number generators are the main components of the stream ciphers, since long keys with good statistical properties are necessary for the suitable performance of stream ciphers.

The classic stream cipher example is the *Vernam Cipher*, defined by Equation 2.8. Where $m_1, m_2, m_3, \ldots$ are the plaintext digits, $k_1, k_2, k_3, \ldots$ are the keystream digits and $c_1, c_2, c_3, \ldots$ are the ciphertext digits. The symbol $\oplus$ denotes the XOR operation (bitwise addition modulo 2), and the decryption is defined by $m_i = c_i \oplus k_i$. If the keystream digits are generated independently and randomly, the Vernam Cipher is called the *one-time-pad*, and is unconditionally secure against a ciphertext-only attack. Shannon proved that a necessary condition for a symmetric-key encryption scheme to be unconditionally secure is that the uncertainty of the secret key must be at least as large as the uncertainty of the plaintext [92]. The one-time-pad is unconditionally secure regardless of the statistical distribution of the plaintext.

$$c_i = m_i \oplus k_i \tag{2.8}$$

An obvious drawback of the one-time-pad is that the key should be as long as the plaintext, which represents a problem for the key management or distribution. This motivates the design of stream ciphers where the keystream is pseudo-randomly generated from a smaller secret key. Such stream ciphers do not offer unconditional security (since the key is smaller than the plaintext), but they are designed with the aim to be computationally secure [61].

Stream ciphers based on LFSRs

LFSRs are used as keystream generators due to their suitable properties for hardware implementation, producing sequences with large periods and good statistical properties. Since well-designed systems should be secure against known-plaintext attacks, a LFSR should never be used by itself as a keystream generator because of its predictability. Nevertheless, LFSRs are desirable because of their very low implementation costs, large periods and good statistical properties. Two general methodologies for avoiding the linearity properties of LFSRs are used:

- Using a nonlinear filtering function on the contents of one (or more) LFSRs
- Using the output of one (or more) LFSRs to control the clock of one (or more) other LFSRs.

Filters use a nonlinear feedback function as an input to the register. The filter should not be too simple to be weak but neither

too complex, otherwise it would become the bottleneck of the generator. However recent attacks to the MIFARE PRNG [30] have demonstrated the vulnerability of this kind of generators when the nonlinear function is not taken carefully. Another example of nonlinear filter generators is the *Knapsack generator* developed by Merkle and Hellman [62].

Another approach to break the linearity of a LFSR is to use a nonlinear combination of multiple LFSRs to generate a unique output. Generally, the output of one LFSR is used to select or combine the output of one or more LFSRs, in the same or different clock times. Known examples of this approach are the Geffe [33], A5 [90] or the Shrinking generator [90]. The output generated from this constructions is statistically weak, being vulnerable to correlation or side-channel attacks [47]. Also the irregular output data rate from some of these constructions (e.g. the shrinking generator) is not suitable for PRNG used in security environments.

Another strategy for the security improvement of LFSR based stream ciphers is to keep the feedback connection polynomial secret. For known connections, the secret key generally consists of the initial contents of the component LFSRs. For secret connections, the secret key for the keystream generator generally consists of both the initial contents (or initialization vector) and the connections polynomial.

2.3 Stream Ciphers for RFID Constrained Devices

In the remainder of this section some stream ciphers and pseudo-random number generators specifically designed for constrained devices (e.g. EPC Gen2 devices) are introduced.

Since the ratification of the EPC Gen2 specification [25] and the ISO standards ISO/IEC 18000-6C [6], in which the usage of on-tag PRNGs on low-cost RFID devices is presented as mandatory, the number of stream ciphers for security solutions has increased in the industry and academic research. Cryptographic suitable PRNGs designs must satisfy, in addition to a low hardware complexity, some other relevant properties. A crucial property that stream ciphers designed for security purposes must address is unpredictability. Indeed, an external adversary who eavesdrops the communication cannot compute the PRNG internal state, even if many outputs of the generator have been observed. The adversary cannot either compute the next sequence, even if many other previous sequences have been observed. If the adversary can observe, or even manipulate, the input samples that are fed by a PRNG, but its internal state is not known, the adversary must not be able to compute the next sequence or the next internal state of the PRNG. Finally, if the adversary has somehow learned the internal state of the PRNG, but the input samples that are fed in cannot be observed, then the adversary should not figure out the internal state of the PRNG after the re-keying operation.

Focused on RFID low-cost technologies like EPC Gen2, the design of PRNGs for security applications is not an easy task due to the computational and memory restrictions that these tiny devices imply. Capabilities of this type of tags are so small that security features for the EPC Gen2 standard are expected to be implemented with a small amount of equivalent logic gates (GE), defined in the literature between 2,000 and 5,000 [85]. This is a extremely small value if we consider that a standard hash function (the most simple cryptographic transformation), like SHA-1, needs at least 8,120 GE to be implemented [28]. Existing commercial Gen2 tags do implement a PRNG, as it is an EPC standard mandatory, but companies are

often reluctant to present the design of their PRNGs. Manufacturers simply refer to testbeds that show the accomplishment of some expected requirements, most of them for compatibility purposes.

In the literature, some suitable PRNG designs for resource constrained devices are proposed. We specially focus on designs motivated by security purposes, that is, where the PRNG is used as a part of the stream cipher algorithm. The following paragraphs introduce PRNGs for low-cost RFID based on LFSRs and other algorithms.

2.3.1 LFSR-based PRNGs for low-cost RFID

Implementations of PRNGs based on LFSRs for lightweight RFID applications have been proposed in the scientific literature [53, 39, 30]. Regarding the EPC Gen2 technology, to the best of our knowledge, only the Che *et al.* PRNG [18] describes a hybrid approach that combines the use of Linear Feedback Shift Registers (LFSR) and physical properties to build random sequences (see Section 3.1 for a detailed description of their scheme).

Apart from the Che *et al.* proposal, there are not many references in the literature that combine true random data and LFSR to obtain a good PRNG. The main reason, as stated in Section 2.1, is that the obtained PRNG cannot be used as an additive stream cipher for a standard sender-receiver communication model due to the infeasibility of reproducing the same sequence at both communication parts since the cipher sequence will be affected by a true random source.

Mifare Crypto1

Crypto1 stream cipher (cf. Figure 2.8)is a proprietary encryption algorithm developed by NXP Semiconductors [69] to be used in the

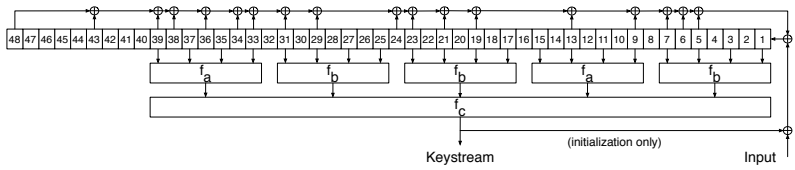


Figure 2.8: Crypto1 uses a 48-bit LFSR with non-linear filters as stream cipher (image extracted from [30])

Mifare Classic HF RFID tag. The security of the card relies partly on the secrecy of the Crypto1 algorithm, which is commonly known as "security by obscurity". However the stream cipher was uncovered with reverse engineering [68] and cryptanalysis [30].

The pseudo-random number generator producing the keystream is a 48-bit LFSR with the feedback polynomial $x^{48}+x^{43}+x^{39}+x^{38}+x^{36}+x^{34}+x^{33}+x^{31}+x^{29}+x^{24}+x^{23}+x^{21}+x^{19}+x^{13}+x^{9}+x^{7}+x^{6}+x^{5}+1$, with two layers of nonlinear filters f_a, f_b and f_c defined by:

$$\begin{aligned} f_a(x_a, x_1, x_2, x_3) &= ((x_0 \vee x_1) \oplus (x_0 \wedge x_3)) \oplus \\ &\quad (x_2 \wedge ((x_0 \oplus x_1) \vee x_3), \\ f_b(x_a, x_1, x_2, x_3) &= ((x_0 \wedge x_1) \vee x_2) \oplus ((x_0 \oplus x_1) \wedge (x_2 \vee x_3)), \\ f_c(x_a, x_1, x_2, x_3, x_4) &= (x_0 \vee ((x_1 \vee x_4) \wedge (x_3 \oplus x_4))) \oplus \\ &\quad ((x_0 \oplus (x_1 \wedge x_3)) \wedge ((x_2 \oplus x_3) \vee (x_1 \wedge x_4))), \end{aligned}$$

where $\vee$, $\wedge$ and $\oplus$ denote logical OR, AND and XOR operations respectively. A 32-bit LFSR is used for nonce generation and cipher initialization, but the feedback polynomial $x^{16}+x^{14}+x^{13}+x^{11}+1$ has degree 16, thus, in fact the LFSR period is only 2^{16}. Moreover, the

LFSR is always initialized with the same values thus, a synchronization attack with precomputed values is enough to predict the Crypto1 cipher state (which is the secret of the stream cipher), therefore also the sequences generated by the stream cipher.

Self-Shrinking Generator

Lee and Hong propose in [53] the self-shrinking generator to be used in low-cost RFID tags. The self-shrinking generator is the variant of a PRNG proposed by Coppersmith *et al.* in [21], known as the shrinking generator. The shrinking generator is a well studied cryptographic design that combines two clocked LFSRs (cf. Figure 2.9). The output sequence of the first LFSR is used to discard some bits from the output sequence of the second LFSR [61]. In other words, the linearity of the output sequence of the second LFSR is perturbed by the output sequence of the first LFSR.

The self-shrinking generator, proposed by Meier and Staffelbach in [58], simplifies the previous scheme. It uses only one single LFSR whose output sequence is partitioned into pairs of bits. Lee and Hong estimate that the hardware implementation of the self-shrinking generator on a low-cost RFID tag requires 1,435 logic gates, and 517 clock cycles at 100 kHz. Regarding specific security issues, it is worth mentioning that some techniques presented in [58, 63] can be used to attack the scheme, provided that feedback polynomials are known. In this sense, no evidences of how their proposal controls the irregularities of the generator's output rate (an important drawback inherent to the shrinking generator [34]) are provided. If this problem is not properly handled, it would difficult its hardware implementation, and it can also give hints about the state of the main LFSR.

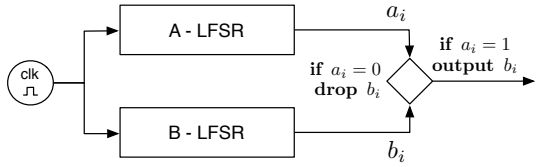


Figure 2.9: The Shrinking Generator has shown that is not suitable for cryptographic operations

Meier and Staffelbach [58] also proved that if the length of the LFSR used in the self-shrinking generator is n, then the period and linear complexity of the output sequence of the self-shrinking generator are at least $2^{\frac{n}{2}}$ and $2^{\frac{n}{2}} - 1$, respectively.

Grain

The ECRYPT Network of Excellence started the eSTREAM project [71] with the aim to improve the current state-of-the-art of stream cipher algorithms. The project ended with new promising stream cipher proposals. *Grain* [39] is one of these proposals, designed for hardware efficient implementation.

Grain is based on the combination of two feedback shift registers, one of which uses LFSR while the other uses Nonlinear-LFSR. *Grain*'s key size is 80-bit and its initialization vector size is 64-bit, that is the same size as the two registers. The authors claim that the linear complexity and period of their design corresponds to the size of the 80-bit LFSR, thus, 2^{80} and $2^{80} - 1$, respectively. However, because of the NFSR and the fact that the input to this is masked with the output of the LFSR, the exact period will depend on the key and the initialization vector used. Figure 2.10 shows a schematic overview of *Grain*.

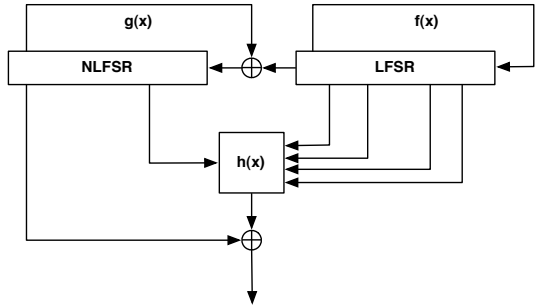


Figure 2.10: *Grain* is a hardware efficient stream cipher [39]

Since this generator is bit-oriented, one bit is generated each clock cycle. There are different implementations of the *Grain* stream cipher increasing the output data rates by also increasing the hardware complexity of the system. The most hardware-efficient implementation of *Grain* takes 1,294 GE.

Che's PRNG

Che *et al.* present in [18] a new PRNG intended for EPC Gen2 applications. Similarly to the shrinking generator discussed above, their proposal uses a LFSR whose linearity is supposed to be perturbed by a physical source of randomness. This hybrid combination leads to an efficient generator with low consumption and hardware complexity. More specifically, their scheme exploits the initial state of a 16-bit LFSR combined with the addition of a physically-generated truly random bit for each cycle ring. According to the authors, the addition of only a truly random bit in the cycle ring as a random number seed, makes the LFSR output sequence to be unpredictable and irreproducible, just like a true random number generator (TRNG).

Hence, the period and linear complexity should be considered as infinite (∞). Section 3.1 presents a detailed analysis of the Che *et al.* proposal.

2.3.2 Other low-cost RFID PRNGs

Apart of LFSR based PRNGs, other algorithms and techniques are proposed in the literature for lightweight RFID applications [22, 42, 10, 51, 75].

Trivium

The Trivium stream cipher [22] is also part of the eSTREAM Project [71]. It is designed to provide a flexible trade-off between speed and gate count in hardware, as well as a reasonably efficient software implementation. Trivium's 288-bit internal state is organized in three different registers of different lengths. The most hardware-efficient implementation uses 1,857 GE to output one bit for each clock cycle. More expensive hardware implementations can increase the output data rate up to 64-bit. Because of the fact that the internal state of Trivium evolves in a nonlinear way, its period is hard to determine [22]. However the authors consider that their design generates a period between 2^{80} and 2^{93} from the 288-bit combination of key and initial state.

Invertible Mappings

Similarly, Klimov et al. present in [51] invertible bit transformations of 32 or 64-bit, suitable for PRNG applications, specially in software

implementations. The authors propose forward invertible mappings by updating the Equation 2.9.

$$x_i = \left(x_{i-1} + (x_{i-1}^2 \vee 5)\right) \mod 2^{64} \tag{2.9}$$

The authors claim that the pure PRNG produced by the equation are not cryptographically secure in themselves, but can serve as excellent building blocks in software based generators (in the same way as LFSRs are insecure as stand-alone designs, but they can serve as excellent components in hardware based generators) [51]. Specifically, the upper halves (32-bit) of the numbers defined by Equation 2.9 are statistically random [4].

FERNS

Holcomb *et al.* propose in [42] a method to derive truly random data using the initial state of tag Static Random Access Memory (SRAM). Their measurements show that initialization of SRAM produces a physical fingerprint. They propose a system of Fingerprint Extraction and Random Numbers in SRAM (FERNS) that harvests static identity and randomness from existing volatile memory storage.

The FERNS method produces 0.103 bit of entropy per raw bit of memory. That means that a truly random bit can be generated from ten bits of memory. Nevertheless, experiments extracting 128-bit patterns from a testing device with 2,048-bit of memory fails to pass entropy tests [42]. The authors propose the use of Universal Hash Functions (cf. Section 2.2) to improve the output randomness. Despite of these preliminary results, the FERNS method is an interesting and power efficient TRNG that can be useful in combination with other PRNG methods.

440 nA TRNG

Balachandran *et al.* propose in [10] the extraction of randomness by sampling radio signals. Although it is designed to solve possible collision problems, its method of truly random bit jittery-clock-generation can be useful in combination with other PRNG methods, in a similar manner as the Holcomb *et al.* method. The low-power consumption of this proposal (440 nA) is suitable for its implementation on resource-constrained devices.

LAMED

Peris *et al.* present in [75] a low-cost PRNG specifically designed to meet the EPC Gen2 requirements. Based on a genetic programming methodology, the authors provide the EPC Gen2 specifications as the input parameter of their methodology. As a result, they automatically derive a set of functions that, eventually, build a 32-bit PRNG. The resulting generator, referred as LAMED, is accomplished by assembling the set of functions that are obtained. The set of functions mainly consists of bit rotations, bitwise operations, and modular algebra. Since the EPC Gen2 specification requires the use of 16-bit PRNGs, the authors propose an alternative 16-bit version of their PRNG, referred as LAMED-EPC. Figure 2.11 overviews the LAMED diagram.

To reduce the output length from 32 to 16-bit, Peris *et al.* divide the 32-bit output of LAMED in two halves, namely $MSB_{31:16}$ and $LSB_{15:0}$. Then, an additional XOR operation before the final output sequence XORes, these two halves to obtain the 16-bit output sequence. The hardware implementation of LAMED is estimated by the authors to 1,566 logic gates, 186 clock cycles at 100 kHz, and

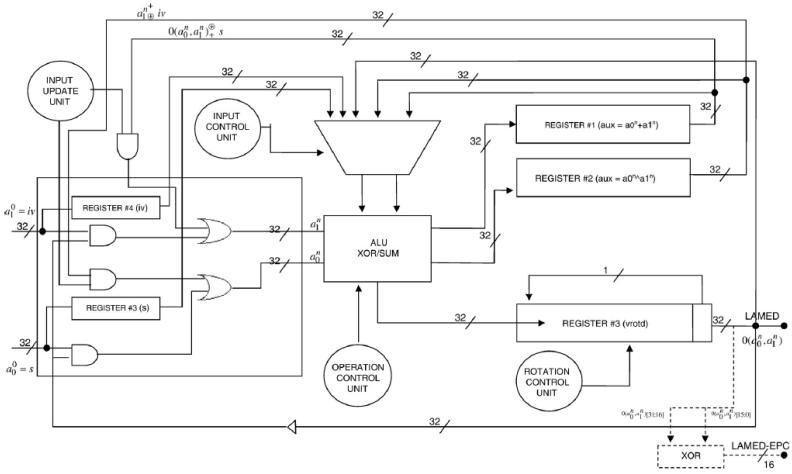


Figure 2.11: LAMED PRNG for EPC Gen2 tags extracted from [75]

64-bit of memory. LAMED-EPC requires some more circuitry and clock cycles, being the final count estimated to 1,584 GE and 194 clock cycles. An exhaustive statistical analysis confirms that both proposals successfully satisfy the initial expectations. No evidences of further achievements other than hardware complexity and statistical behavior are provided. Moreover, the inherent peculiarity of their construction methodology makes it difficult to compare with other designs in the literature.

2.4 Evaluation Tools

In the remainder of this section, the security evaluation tools used in this dissertation are presented. On one hand, the IAIK UHF Demo Tag [3] is a programmable device with an EPC Gen2 RF front-end,

capable to incorporate new algorithms and designs to the official EPC Gen2 protocol for communications. On the other, the National Institute for Standards and Technology (NIST) Statistical Test Suite for Random and PRNG for Cryptographic Applications 800-22 [4] is used to check the statistical properties of the PRNGs analyzed in this dissertation.

2.4.1 IAIK UHF Demo Tag

The *IAIK UHF Demo Tag* [3] is a programmable device intended for developing new commands or functionalities to the EPC Gen2 standard. It allows, moreover, to verify the new functionality using compliant EPC Gen2 readers, by modifying the code inserted into the Demo Tag. Thus, new developments can be implemented and tested in real environments.

The Demo Tag (cf. Figure 2.12) consists on four main components: an antenna, a radiofrequency (RF) front-end, a programmable microcontroller, and a firmware library. The antenna harvests (i.e., converts) the energy emitted by the reader into the appropriate voltage that powers up the RF front-end. The RF front-end rectifies the voltage and demodulates the information encoded in the original signal. The resulting data feeds the programmable microcontroller which, in turn, computes the appropriate responses. To compute the responses, the programmable microcontroller executes a software implementation of the EPC Gen2 protocol, provided as a firmware library. The responses are then modulated by the RF front-end and eventually backscattered to the reader. We present in the sequel a condensed background on these four components. More details can be found in [3, 7].

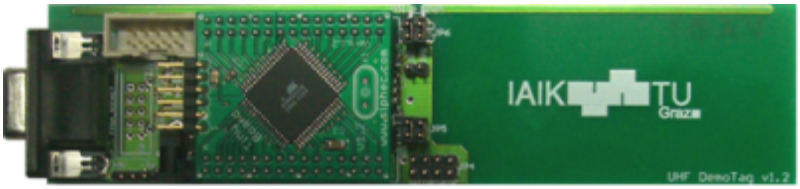


Figure 2.12: Image of UHF Demo Tag from the website of IAIK [3]

Antenna and RF front-end

The antenna connected to the RF front-end is composed of a dipole antenna of about 17cm of overall length. The RF front-end uses a 2-stage charge-pump rectifier to perform amplitude-shift keying (ASK) demodulation. It extracts the information stored by the reader in the signals' amplitude which is transmitted by the reader-to-tag channel. It does, indeed, rectification, voltage multiplication, and envelope detection all at once [7]. The power extracted by the rectifier from the RF field emitted by the reader from most compliant EPC Gen2 readers amounts to having about 2.4mW of power. This does not allow to power the microcontroller. For this reason, the Demo Tag adopts a semi-passive approach, meaning that although the analog parts are powered by the energy harvested from the reader, the digital parts (e.g., the programmable microcontroller) are powered by an external power supply or by an on-board battery. The backscattering of the information computed by the programmable microcontroller is in fact the reflected power of the antenna. This power is indeed generated according to the transmitted data. The RF front-end of the Demo Tag combines both ASK and PSK (phase-shift keying) to modulate the information. The backscattering components used by the Demo Tag to modulate the tag-to-reader signals are a resistor, a capacitor, and a fast-switching transistor placed close to the antenna. These

components are controlled by the programmable microcontroller and the functionalities implemented on it.

Programmable microcontroller and firmware Library

The programmable microcontroller connected to the RF front-end of the Demo Tag consists on an Atmel AVR ATmega128 [1]. It contains all the logic and memory necessary for the Demo Tag. The ATmega128 is an 8-bit microcontroller based on the AVR architecture. The memory banks of the microcontroller, of 128KB of flash memory and 4KB of data memory, can be addressed by three independent 16-bit registers. The rest of the registers of the ATmega128 consist on 32 registers of 8-bit. All 32 registers can be accessed as the destinations of the ATmega128 arithmetic operations. The microcontroller operates exactly one instruction per clock cycle, at frequencies up to the order of 16 MHz. An external crystal oscillator connected to the Demo Tag provides the 16 MHz signal to the microcontroller. Three main signals connect the microncontroller to the RF front-end. A first signal, called DEMOD, provides the demodulated UHF signal from the reader-to-tag channel. A second signal, called MOD, allows the ATmega128 to control the backscatter used to generate the tag-to-reader responses. Finally, a third signal, called RF ON, provides a boolean value to detect the presence of the RF field.

The original IAIK UHF Demo Tag already provides an appropriate implementation of the EPC Gen2 protocol for the ATmega128. The protocol implementation is provided as a firmware library stored in the flash memory of the microcontroller. This library contains, indeed, all the appropriate set of functions necessary to process the readers' standard queries and to compute the appropriate responses. The microcontroller is connected, via an UART module, to a serial-

interface connector. This serial interface allows us to interact with the Demo Tag, in order to provide some basic operations such as memory mapping, EPC Gen2 values' configuration, visualization of queries and responses exchanged with compliant readers, and execution of user defined operations. This latter possibility allows, in fact, to complement the original protocol implementation with new functionalities defined at a user level. By using the JTAG connector provided by the Demo Tag, it is possible to upload new functionalities to the flash memory of the microcontroller, as well as to perform program debugging. A combination of C code and assembly code can be used to complement or modify the original firmware library. An appropriate JTAG download cable allows the transfer of new functionalities or firmware updates. Some other modules connected to the Demo Tag allow more complex programming possibilities, such as FPGA-based UHF protocol implementations. We refer the reader to [3, 7], and citations thereof, for more information.

2.4.2 NIST Randomness Statistical Test Suite

It is impossible to give a mathematical proof of the correctness of a PRNG. Nevertheless, specific statistical tests can help to detect certain kinds of weaknesses a generator may have [61]. A statistical test determines whether a randomly generated sequence possesses certain attribute that a truly random sequence would be likely to show. For example, a random binary sequence is supposed to have the same number of zeros and ones. If the analyzed sequence fails to pass some specific statistical tests, the generator is *rejected* as being non-random, or subjected to further testing. On the other hand if the sequence passes all the tests it is considered *non-rejected*, since passing the tests is not a mathematical proof of the generator randomness [61].

The National Institute for Standards and Technology (NIST) provides in its website [4] the documentation and source code for the Statistical Test Suite for Random and PRNG for Cryptographic Applications 800-22. This suite of tests is designed for testing random or pseudo-random number generated sequences, to detect possible statistical deviations to a given degree of significance.

The Normal and χ^2 distribution

The normal and χ^2 distributions are used for statistical analysis. The normal distribution arises when a large number of independent random variables having the same mean and variance are added, and it is defined by Equation 2.10. X is said to be $N(\mu, \sigma^2)$. If X is $N(0, 1)$, then X is said to have a *standard normal distribution* [61].

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \quad -\infty < x < \infty \tag{2.10}$$

The χ^2 (chi-square) distribution can be used to compare the *goodness-of-fit* of the observed frequencies of events to their expected frequencies under a hypothesized distribution. The χ^2 distribution with v degrees of freedom arises when the squares of v independent random variables having standard normal distributions are summed, and its probability density function is defined by Equation 2.11 , where Γ is the gamma function [61]. The mean and variance of this distribution are $\mu = v$, and $\sigma^2 = 2v$.

$$f(x) = \begin{cases} \frac{1}{\Gamma(\frac{v}{2})2^{\frac{v}{2}}} x^{\frac{v}{2}-1} e^{\frac{-x}{2}} & , \quad 0 \leq x < \infty \\ 0 & , \quad x < 0 \end{cases} \tag{2.11}$$

Hypothesis testing

NIST testing algorithms use a hypothesis test considering the randomness of the sequence as the *null hypothesis* H_0, and the non-randomness as the alternative hypothesis, H_a. Tests are performed regarding a level of significance or critical value, denoted as α hereinafter.

NIST tests produce *P-values* summarizing the strength of the hypothesis. If *P-values* $\geq \alpha$, H_0 is accepted. Even in the situation of some results under the level of significance, additional numerical experiments should be done to determine whether the result was a statistically anomaly or a clear evidence of non-randomness. The NIST recommendation for acceptable uniformity of *P-values* is then applied. The proportion of test over the significance level, must fit in the interval

$$\hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}} \tag{2.12}$$

where $\hat{p} = 1 - \alpha$, and m is the sample size. To statistically confirm the randomness of the analyzed data, one would expect one in 100 sequences to be rejected (represented as $\alpha = 0.01$), being a common value in cryptography [4]. *P-values* passing α give a confidence of 99.9% of the randomness of the evaluated sequence (if 100 sequences are evaluated, results should pass 0.9615 as defined in Equation 2.12).

Tests included in NIST

Some tests included in the NIST suite are in fact common tests used in statistical testing, for example the Frequency Test (or Monobit Test), the Serial Test or the Runs Test [61]. The following is the list of the specific tests included in the NIST suite:

- The Frequency (Monobit) Test
- Frequency Test within a Block
- The Runs Test
- Tests for the Longest-Run-of-Ones in a Block
- The Binary Matrix Rank Test
- The Non-overlapping Template Matching Test
- The Overlapping Template Matching Test
- Maurer's *Universal Statistical* Test
- The Linear Complexity Test
- The Serial Test
- The Approximate Entropy Test
- The Cumulative Sums (Cusums) Test
- The Random Excursions Test
- The Random Excursions Variant Test

These tests focus on a variety of different types of non-randomness that could exist in a sequence. NIST recommends to run the *Frequency* test first, since this supplies the most basic evidence for the existence of non-randomness in a sequence, specifically, non-uniformity [4]. If the *Frequency* test fails, the likelihood of other tests failing is high. Also NIST recommends the size of the sequences to be tested, being 10^7 the larger size to be tested. A number of test in the suite have the *standard normal* and χ^2 as a reference distributions. If the sequence under test is in fact non-random, the calculated test

statistic will fall in extreme regions of the reference distribution. The *standard normal* distribution is used to compare the value of the test statistical obtained from the random or pseudo-random number generation with the expected value of the statistic under the assumption of randomness. The test statistic for the standard normal distribution takes the form $z = (x - \mu)/\sigma$, where x is the sample statistic value, and μ and σ^2 are the expected value and the variance of the test statistic.

The χ^2 distribution is used to compare the *goodness-of-fit* of the observed frequencies of a sample measure to the corresponding expected frequencies of the hypothesized distribution. The test statistic takes the form $\chi^2 = \sum((o_i - e_i)^2/e_i)$, where o_i and e_i are the observed and expected frequencies of occurrence of the measure, respectively.

In the following paragraphs we describe the purpose of some NIST tests that are used later on this dissertation. Specifically, we describe the purpose and function of the Frequency, Runs, Binary Matrix Rank, Overlapping Template, Serial, Approximate Entropy and Cumulative tests. If the computed *P-value* of the described tests is lower than α, then we can conclude that the sequence is non-random. Otherwise, we can conclude that the sequence is random.

Frequency (Monobit) test The purpose of this test is to determine whether the number of 0's and 1's in the analyzed sequence are approximately the same, as would be expected for a truly random sequence. The test assesses the closeness of the fraction of ones to 1/2, that is, the number of ones and zeros in a sequence should be about the same.

Frequency Test within a Block This test is similar to the *Frequency* test, but considering the proportion of ones within M-bit blocks. The purpose of this test is to determine whether the frequency of symbols in a M-bit block is approximately $M/2$, as would be expected under an assumption of randomness. For block size $M = 1$ this test is equal to the Frequency (Monobit) test.

Runs test The focus of this test is the total number of runs in the sequence, where a run is an uninterrupted sequence of identical bits. A run of length k consists of exactly k identical bits and is bounded before and after with a bit of the opposite value. The purpose of the runs test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such zeros and ones is too fast or too slow.

Binary Matrix Rank test The focus of the test is the rank of disjoint sub-matrices of the entire sequence. The purpose of this test is to check for linear dependence among fixed length substrings of the original sequence. To do this a number of matrix are created from the analyzed sequence, and the rank for each matrix is determined, expecting a specific range of ranks to be bounded by the χ^2 distribution. That is, if linear dependences between the values of the analyzed sequence are found (e.g. if the sequence is generated from a LCG or a LFSR without modifications), the test will conclude that the analyzed sequence is non-random.

Overlapping Template test The focus of the Overlapping Template Matching test is the number of occurrences of pre-specified target strings. This test uses a m-bit window to search for a specific

m-bit pattern. If the pattern is not found, the window slides one bit position. A χ^2 distribution of the analyzed sequence measures how well the observed number of template *hits* matches the expected number of template *hits* (under an assumption of randomness).

Serial test The focus of this test is the frequency of all possible overlapping m-bit patterns across the entire sequence. The purpose of this test is to determine whether the number of occurrences of the $2m$ m-bit overlapping patterns is approximately the same as would be expected for a truly random sequence. Random sequences have uniformity, that is, every m-bit pattern has the same chance of appearing as every other m-bit pattern. Note that for $m = 1$, the Serial test is equivalent to the Frequency test.

Approximate Entropy test Like the Serial test, the focus of this test is the frequency of all possible overlapping m-bit patterns across the entire sequence. Specifically, the purpose of the test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (m and $m+1$) against the expected result for a truly random sequence.

Cumulative test The focus of this test is the maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted ($0 = -1$, $1 = +1$) digits in the sequence. The purpose of the test is to determine whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small in relation to the expected behavior of that cumulative sum for random sequences. This cumulative sum may be considered as a random walk. For a random sequence, the excursions of the random walk should be near zero. For certain types of non-random sequences, the

excursions of this random walk from zero will be large. Hence, if s is the sequence to be analyzed we would expect a cumulative sum result of zero: $s = 1+(-1)+1+(-1)+(-1)+1+(-1)+1+1+(-1) = 0$.

2.5 Chapter Summary

PRNGs are used in stream ciphers as a keystream generators due to their suitable properties for hardware implementation, producing sequences with large periods and good statistical properties. Stream ciphers based on PRNGs are, thus, a suitable option for cryptographic implementation in resource constrained devices.

In this chapter we have introduced the main characteristics of the EPC Gen2 standard for low-cost passive RFID tags, which implement a one-time-pad stream cipher using a PRNG as a keystream generator. Despite of its importance for the security, nor the EPC Gen2 standard, neither the scientific literature, focus on the design and analysis of suitable PRNG schemes for security applications.

In the next chapter, we present a theoretical and practical attack to a PRNG proposal for low-cost RFID based on a linear feedback shift register. Furthermore, an empirical analysis of PRNGs used in commercial EPC Gen2 tags is presented, thanks to a novel eavesdropping technique based on commercial products and using standard EPC Gen2 commands.

3

Security Analysis for EPC Gen2 PRNGs

The Electronic Product Code Generation 2 (EPC Gen2) is an international standard that proposes the use of Radio Frequency Identification (RFID) in the supply chain. It is designed to balance cost and functionality. As a consequence, security on board of Gen2 tags is often minimal. It is, indeed, mainly based on the use of on-board pseudo-random number generators (PRNGs), used to obscure the communication between readers and tags; and to acknowledge the proper execution of password-protected operations.

In this chapter, we present a practical implementation attack on a weak PRNGs designed specifically for EPC Gen2 tags [18]. We show that it is feasible to eavesdrop a small amount of pseudo-random values by using standard EPC Gen2 commands and using them to

determine the PRNG configuration that allows to predict the complete output sequence [59].

Furthermore, we also perform an analysis of the pseudo-random sequences generated from real commercial tag integrated circuits (ICs). Specifically, ICs from *NXP* [70], *Alien* [95] and *Impinj* [43] are tested. The main analyzed parameter is the probability of appearance of any single pseudo-random sequence, evaluating the statistical properties of the commercial ICs PRNG.

3.1 Theoretical Attack to a PRNG EPC Proposal

In [18], Che *et al.* present a new PRNG for application in RFID tags. Their system relies on an oscillator-based Truly RNG (TRNG), and exploits the thermal noise of two resistors to modulate the edge of a sampling clock. Authors state the final system prevents potential attackers to perform any effective prediction about the generated sequence (even if the design is known) based on a white noise based cryptographic key generation.

After describing its TRNG oscillator-based core, the authors focus on design considerations specially regarding *power consumption* and *output data rate* trade-offs. Knowing the fact that the higher the frequency oscillation of the system, the higher the current (thus also power) consumption, the authors look for system level optimization in order to reduce the power consumption due to the low-power restrictions of RFIDs.

The optimization proposed by Che *et al.* relies on the combination of the TRNG and a linear feedback shift register (LFSR) (cf. Figure

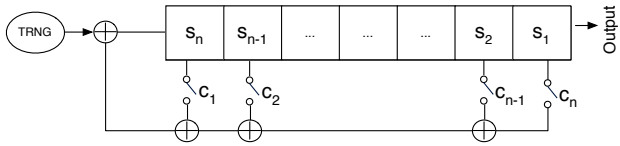


Figure 3.1: PRNG scheme based on the Che *et al.* specifications.

3.1). Adding a LFSR to the TRNG lets the system reduce the clock frequency proportionally to the number of cells of the LFSR. Specifically, exploiting the initial state of a 16-bit LFSR combined with the addition of the generated truly random number (*trn*) for each cycle ring, allows the system to decrease the clock frequency with a $\frac{1}{16}$ factor.

Authors claim that [18]: *"If we add 1-bit truly random number in the cycle ring as a random number seed, the output sequence of the LFSR will also be unpredictable and irreproducible as a TRNG."*.

We show in the remainder of this section that this claim does not hold. It is worth mentioning that Strüker *et al.* also cite in [93] functional weaknesses of the Che *et al.* scheme, although, no results nor proofs are given in their paper.

3.1.1 Analyzing the Che *et al.* Proposal

In this section, we present an analysis of the PRNG proposal presented by Che *et al.* described above. We give the details of a statistical analysis, performed over the output data, based on the NIST statistical test for pseudo-randomness. Based on these results, we describe an attack that, given a small number of output bits, can determine the whole sequence [60].

Che *et al.* Statistical Analysis

Since the main property of a PRNG is to ensure the forward unpredictability of its generated sequence, the correctness of a PRNG can be measured with statistical tests applied to the output sequence. For this purpose, we use the NIST Statistical Test Suite for Random and PRNG for Cryptographic Applications 800-22 [4] which has been introduced in Section 2.4.

The Che *et al.* scheme has been implemented by strictly following the specifications stated in [18]. The code has several configurable parameters, such as the size of the LFSR, the feedback polynomial, and the seed values. The sequences of true random bits are obtained from [37].

In order to evaluate the randomness quality of the sequence produced by the Che *et al.* scheme, we have generated 230 MB of output data from an implementation of their proposed PRNG. This amount of data represents more than 100 million 16-bit pseudo-random sequences (RN16). Such data has been divided in ten different data sequences (T_i) that have been independently analyzed using the NIST test suite.

NIST test results for the Che *et al.* random generated data are presented in Table 3.1. Each column represents 23 MB of pseudo-random data (11.5 million RN16s) generated with different *seed* and true random source. Each row refers to a test included in the NIST test suite. The first nine tests are represented with the numerical value of the uniformity of *P-values*. The last five tests are, in fact, a set of different tests, thus, in order to represent each of the values, an achievement ratio is represented following the same decision rule of the first tests (proportion of 0.96 for 100 runs, Equation 2.12). Tests

refusing randomness hypothesis are denoted with bold letters in the table.

Results show a statistical evidence of non randomness for the *Binary Matrix Rank Test* (cf. Table 3.1). Such test constructs binary matrices from the analyzed data and checks for linear dependence among the rows or columns of the constructed matrices. The fact that the *Binary Matrix Rank Test* fails for all the sequences, gives a clear evidence of non-randomness due to linearity problems.

3.1.2 Exploiting the linearity weaknesses of the scheme

As we have pointed out in Section 2.2, the main vulnerability of a PRNG based on a linear feedback shift register comes from its easy predictability due to its linearity properties.

Results presented in Section 3.1.1 show that the Binary Matrix Rank Test from the NIST statistical test suite fails for the Che *et al.* scheme, providing information that the scheme does not succeed in breaking the linearity of the underlying LFSR. In fact, a specific attack to break the Che *et al.* PRNG based on the inherent linearity of the LFSR has been presented in [59] and [60] and is next briefly described.

Notice that in the Che *et al.* scheme the pseudo-random sequence is produced by a LFSR XORed in its first cell with a *truly* random bit (cf. Figure 3.1). That means we can find a $2n$ pseudo-random output sequence of the proposed scheme identically equal to the one of the $n-$bit LFSR (without of the XORed true bit) in case that 2 consecutive random bits are 0. Such event will occur with probability 1/4 assuming bits are true random.

Table 3.1: Che *et al.* PRNG NIST test results

Runs	100									
Sequence	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
Frequency	0.99	0.98	0.96	1.00	1.00	0.97	0.97	0.99	0.96	0.97
BlockFrequency	1.00	1.00	0.97	0.98	1.00	0.98	1.00	0.99	0.98	0.99
Runs	0.98	1.00	1.00	0.99	0.99	0.99	0.96	0.98	0.98	1.00
LongestRun	**0.96**	**0.96**	0.98	0.97	**0.94**	0.96	**0.95**	0.98	0.99	**0.94**
Binary Matrix Rank	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
OverlappingTemplate	0.96	**0.94**	0.99	0.98	0.98	**0.93**	0.97	0.98	0.98	**0.95**
Universal	1.00	0.99	0.98	0.99	1.00	0.97	0.99	0.99	0.99	0.98
ApproximateEntropy	0.99	0.97	1.00	0.99	0.98	0.98	0.99	1.00	1.00	1.00
LinearComplexity	0.99	1.00	0.99	0.99	**0.95**	1.00	1.00	0.99	0.99	1.00
CumulativeSums	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$
NonPeriodicTemplate	$\frac{148}{148}$	$\frac{148}{148}$	$\frac{148}{148}$	$\frac{148}{148}$	$\frac{148}{148}$	$\frac{148}{148}$	$\frac{148}{148}$	$\frac{148}{148}$	$\frac{\mathbf{147}}{\mathbf{148}}$	$\frac{148}{148}$
RandExcursions	$\frac{\mathbf{7}}{\mathbf{8}}$	$\frac{\mathbf{7}}{\mathbf{8}}$	$\frac{8}{8}$	$\frac{\mathbf{7}}{\mathbf{8}}$	$\frac{8}{8}$	$\frac{\mathbf{7}}{\mathbf{8}}$	$\frac{8}{8}$	$\frac{\mathbf{6}}{\mathbf{8}}$	$\frac{8}{8}$	$\frac{8}{8}$
RandExcursionsVar	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$
Serial	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$

Predictability of the Scheme

We have detailed above that the main vulnerability of a PRNG based on a linear feedback register comes from its easy predictability due to its linearity properties. We will show that the randomness introduced in the Che *et al.* scheme is not enough to mask the linearity of the scheme.

Following the Che *et al.* scheme (cf. Figure 3.1) the pseudo-random sequence is produced by a LFSR XORed in its first cell with a *truly* random bit (generated in the oscillator) for each register cycle in order to be unpredictable and irreproducible [18]. The pseudo-random output sequence for a n cell LFSR can be represented as in Equation 3.1, where the LFSR output s_i is XORed ($\oplus$) with a true random bit.

$$\begin{aligned} s_{k+1} \oplus trn_1, s_{k+2} \oplus trn_1, s_{k+3} \oplus trn_1, \ldots s_{k+n} \oplus trn_1, \\ s_{k+n+1} \oplus trn_2, \ldots s_{k+2n} \oplus trn_2, s_{k+2n+1} \oplus \ldots \end{aligned} \tag{3.1}$$

Since the LFSR seed is modified with the trn_i bit, the LFSR output will also be modified regarding the trn values. If we assume that the trn_i bits are generated by a true random generator, then the probability that $trn_i = 0$ or $trn_i = 1$ is equal to $p = \frac{1}{2}$. Then, since the trn_i value is only XORed for each cycle, when two consecutive 0's are generated by the true random generator, $trn_i = trn_{i+1} = 0$, then the $2n$ bits output stream of the system will be exactly the same as the one produced by the LFSR. This situation can represent a threat for the unpredictability of the system, since these $2n$ values can be used to obtain the feedback polynomial of the LFSR.

Attack Description

The vulnerability defined above has been implemented to support the theoretical analysis with practical results.

Our scenario is composed by a Che *et al.* system that produces pseudo-random bits. Only a part of the pseudo-random output sequence, denoted by s_a is known to the attacker, besides the size n of the LFSR. Moreover, the *seed* (initial state) and the feedback polynomial coefficients remain secret to the attacker. The attack will succeed if the attacker can provide the LFSR feedback polynomial.

To generalize the attack, we also assume that the attacker cannot determine the first bit of the sequence, that means he has no information if a given s_a sequence, with $|s_a| = 2n$ (the length of the sequence), has been affected by exactly two trn values (that means the attacker finds two exact LFSR rounds) or the sequence has been modified by three trn values.

With probability $\frac{1}{n}$, the sequence, s_a with $|s_a| = 2n$ has been affected by exactly two trn and, in this case, the probability to obtain the $2n$ values of the LFSR despite the XORed trn is $\frac{1}{4}$ (two consecutive zeros). That means that, with probability $\frac{1}{4n}$, we can obtain $2n$ values of the LFSR that composes the system and with this sequence we are able to compute the feedback polynomial and the whole pseudo-random sequence.

Now, we assume that $|s_a| = 3n - 1$ (cf. Figure 3.2). If the sequence is divided into n subsequences of length $2n$, we can ensure that one of these subsequences has been affected by exactly two trn. The remainder $n-1$ subsequences, have been affected by three trn. However, notice that if the three trn are zeros, the n vectors of length $2n$ will give the same feedback polynomial. The probability of such

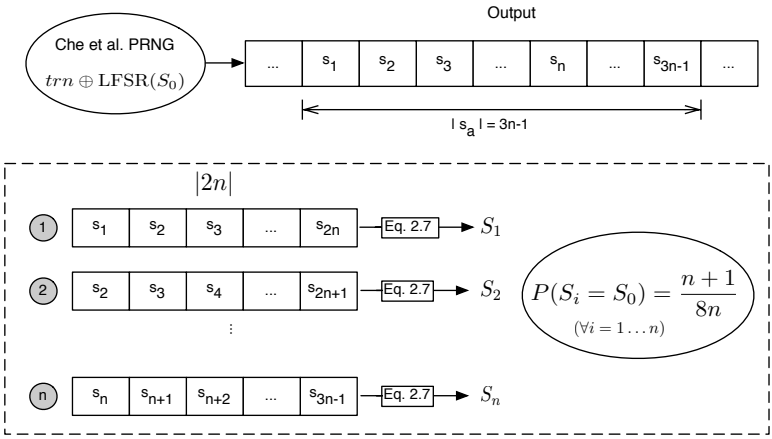


Figure 3.2: Scheme of the attack to Che *et al.* PRNG

event is $\frac{1}{8}$. Then, Equation 3.2 provides the probability of success of an attack that analyzes a sequence with $|s_a| = 3n - 1$:

$$P_{\text{success}}(3n-1) = \frac{1}{4}\left(\frac{1}{n}\right) + \frac{1}{8}\left(\frac{n-1}{n}\right) = \frac{n+1}{8n} \qquad (3.2)$$

Obviously, the probability of success increases with $|s_a|$ since increasing the $|s_a|$ implies that more trn bits affect the sequence and then the probability of finding three consecutive zeros also increases. Figure 3.3 shows the probability of success of an attack with s_a length for a particular system with a LFSR of length $n = 16$. Notice that only 160 bits ($10n$) are enough to perform a successful attack with probability higher than 50%, and 464 bits ($29n$) implies more than a 90% of success probability.

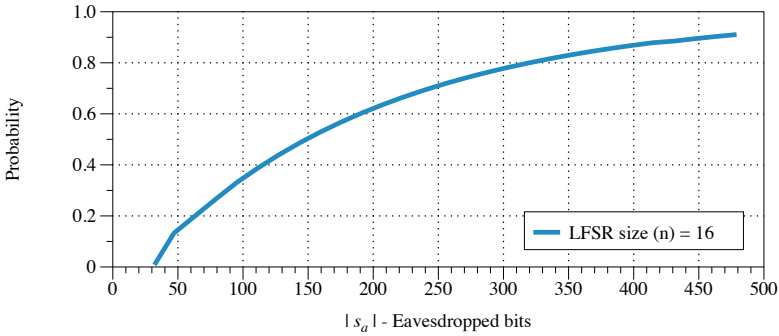


Figure 3.3: Reliability on the Che *et al.* attack regarding $|s_a|$.

Obtained Results

To test the correctness of the theoretical evaluation, the described attack has been implemented over the same ten pseudo-random se-

Table 3.2: Attack success rate for $|s_a| = 3n - 1$.

Sequence	T_1	T_2	T_3	T_4	T_5
attack success (%)	0.132	0.137	0.131	0.126	0.139
Sequence	T_6	T_7	T_8	T_9	T_{10}
attack success (%)	0.137	0.129	0.137	0.138	0.128

Table 3.3: Value of $|s_a|$ for a successful attack in the worst case after 10 tests

Sequence	T_1	T_2	T_3	T_4	T_5
$\|s_a\|$	238	254	254	190	510
Sequence	T_6	T_7	T_8	T_9	T_{10}
$\|s_a\|$	158	254	286	238	222

quences (T_i) used to execute the NIST tests (cf. Section 3.1.1).

The first analysis validates that the probability of finding the feedback polynomial matches the one described in Equation 3.2. In this case, the algorithm takes $|s_a| = 3n - 1$ bits from T_i starting at a random position and tries to attack the system by finding n equal feedback polynomials. The operation is repeated one thousand times for each test sequence T_i. Attack success rates are reported in Table 3.2. Notice that they are close to the theoretic value $\frac{(n+1)}{8n}$ with $n = 16$ $\approx 0,132$.

The second analysis provides the number of bits needed to achieve a successful attack. Ten different attacks have been performed for every T_i data sequence taking the first bit of s_a at random. Results presented in Table 3.3 show the number of bits for a successful attack in the worst case, that is the attack that needs a greater number of bits. Notice that, although considering the worst case, the number of bits is significantly lower than the whole period $2^{16} - 1$.

3.2 Attack Implementation and Empirical Results

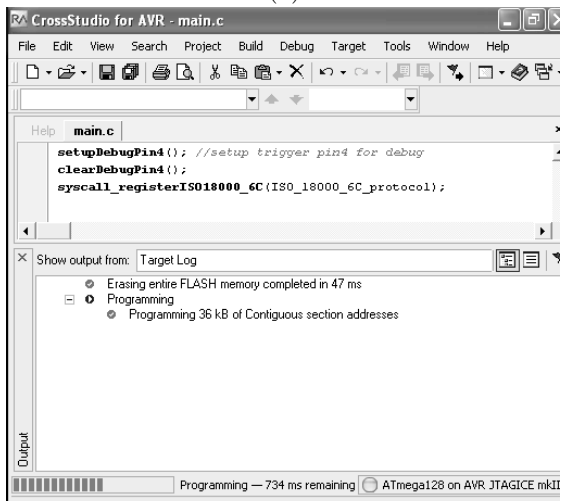
We present in this section a practical attack based on a real EPC Gen2 setup. First, we present the RFID devices used to implement the attack. Second, we describe the implementation scenario and the techniques used to eavesdrop the PRNG from the RFID communication [59]. Finally, we present the empirical results.

3.2.1 Che *et al.* implementation and experimental setup

In Section 3.1.2, we have seen how to attack the pseudo-random number generator proposed by Che *et al.* once a sufficient number of pseudo-random values are collected. We show in this section the results of a practical attack against the vulnerable scheme on a real EPC Gen2 setup. The attack is based on the eavesdropping of the communication between a standard EPC Gen2 reader and the IAIK UHF Demo Tag (cf. Section 2.4). Indeed, we show how it is possible to obtain an appropriate set of random queries generated by an on-board PRNG, based on the Che *et al.* scheme, to eventually predict the generation of pseudo-random sequences that will be generated later over the Demo Tag.

The Che *et al.* scheme has been implemented in ANSI C using the Crossworks IDE for AVR from Rowley Associates [5]. The original scheme provided in [18] has been adapted into a code-optimized EPC Gen2 version that can be executed over the microcontroller of the IAIK UHF Demo Tag. Arithmetic efficient functions such as *bit shifts*, logic operators (*AND*, *OR* and *XOR*) and *modulo 2*, are used to implement the LFSR in the Demo Tag [90]. The *trn* addition is extracted from the less significant bits of the analogical to digital conversion in the microcontroller. Since the generation of pseudo-random sequences is a mandatory operation specified in the EPC Gen2 protocol, an existing PRNG function is already included in the original firmware. By using the Crossworks IDE, we code and merge the PRNG based on the Che *et al.* scheme with the general firmware library to replace the existing PRNG. The JTAG programmer that we use to transfer and to debug the updated firmware merged with the new PRNG implementation is an AVR JTAG MKII programmer [1].

(a)



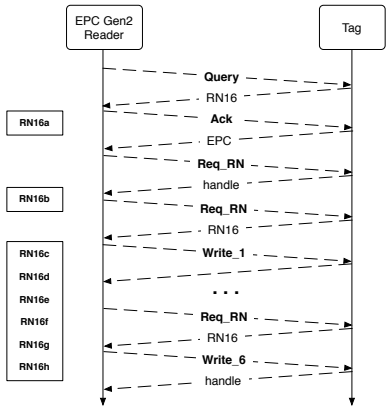
(b)

Figure 3.4: Experimental setup. In (a), we can see the CAEN A829EU Reader, the AVR JTAG MKII Programmer, and the IAIK Graz UHF Demo Tag. In (b), we can see the Crossworks IDE GUI for AVR, uploading the updated firmware over the Demo Tag.

The queries are generated from a standard EPC Gen2 RFID reader. The RFID reader we use is a short-range reader CAEN A829EU [2]. Figure 3.4 shows our experimental setup. The reader is controlled by a desk computer over a USB serial port. For the generation of queries, we use a .NET application that controls the communication process with the reader. This application enables us to generate the set of queries required to proceed with the eavesdropping attack. Finally, we use Matlab [44] to decode the set of responses generated over the Demo Tag. This operation enables us to isolate the pseudo-random queries computed at the Demo Tag. When the number of sequences collected by the application reaches an appropriate threshold, it proceeds to execute the implementation of the attack we presented in Section 3.1.2. We provide in the sequel further details about the collection of pseudo-random sequences and the practical results.

3.2.2 Eavesdropping of pseudo-random sequences and practical results

Due to the Gen2 RF power range characteristics, a realistic attack should only consider reader-to-tag queries because they are much easier to be eavesdropped (cf. Section 2.1). Some reader-to-tag queries include pseudo-random sequences that are computed from the on-board PRNG included on the EPC tags. Table 3.4 shows the mandatory operations for Gen2 reader-to-tag protocol and the minimum number of RN16s involved in each operation. Notice that the *write* command generates a minimum of eight RN16s for its proper execution. For a full EPC code writing, up to six RN16s must be generated to cover the reader-to-tag communication, besides the two previously generated pseudo-random sequences for the inventory query and the handle descriptor [25]. This strategy is used later on this chapter



(a) EPC Gen2 *write* protocol.

```
=> QUERY ...
=> ACK ...
=> Req_RN RN:14438
=> Req_RN RN:44282
=> Write (data) 27698
=> Req_RN RN:44282
=> Write (data) 47380
=> Req_RN RN:44282
=> Write (data) 44282
=> Req_RN RN:44282
=> Write (data) 60868
=> Req_RN RN:44282
=> Write (data) 32656
=> Req_RN RN:44282
=> Write (data) 34674
```

(b) Real capture of six *write* cycles.

Figure 3.5: *Write* process for EPC Gen2 and the PRNG utilization. In (a), we can see the six cycles of the EPC Gen2 *write* command. In (b), we can see a real sample of six *write* cycles captured from the reader-to-tag channel.

to eavesdrop RN16s from EPC Gen2 compliant commercial ICs (cf. Section 3.3).

A *write* operation is an access command used to modify specific areas of a Gen2 tag memory (cf. Section 2.1). The reader first identifies the tag with *select* and *inventorying* commands (which shifts the tag from *ready* to *acknowledged* state). Once the tag is *acknowledged*

Table 3.4: The *Write* Access command uses the maximum number of RN16s when writing the full EPC identification code

Operation	Inventory	Access			
Command	Identification	Read	Write	Lock	Kill
Number of RN16s	1	2	8	2	4

(meaning that the tag has sent its EPC identification) the reader requests a new RN16 to the tag for establishing an *access* session. The new RN16 (denoted as *handle*) acts as a session key, and must be used to link all the *access* actions to a specific tag. Let us observe that all access commands can be executed both in the *open* or *secured* tag state [25]. If the accessed tag is in the secured state, it means a 32-bit password (exchanged as two 16-bit half-passwords XORed with two RN16s) is necessary to allow the reader to access the tag. In our experiments, we assume that the tag is in the open state, i.e., we do not consider the capture of PRNGs derived from the exchange of the two half-passwords. In this way, an inventoried tag transitions directly to the access mode. For a *write* operation, once the reader gets the *handle*, it initiates a round of writes of 16-bit data sequences (obscured with previously requested RN16s) to the tag. Thus, if a new EPC identification is written to the tag, six write cycles are performed, as we picture in Figure 3.5 (a). The eight generated RN16s represent 128 consecutive bits generated from the PRNG of an EPC Gen2 tag.

$$\begin{aligned} \text{EPC}_{\,0\ldots15} &= 0_{0\ldots15} \oplus \text{RN16}_1 \\ \text{EPC}_{16\ldots31} &= 0_{0\ldots15} \oplus \text{RN16}_2 \\ &\vdots \\ \text{EPC}_{80\ldots95} &= 0_{0\ldots15} \oplus \text{RN16}_6 \end{aligned} \tag{3.3}$$

As we pointed out in Section 3.1.2, the Che *et al.* scheme can be predicted with a reasonable small amount of data. We can now demonstrate this property in our real Gen2 environment, by simply performing an appropriate series of *write* challenges to the adapted

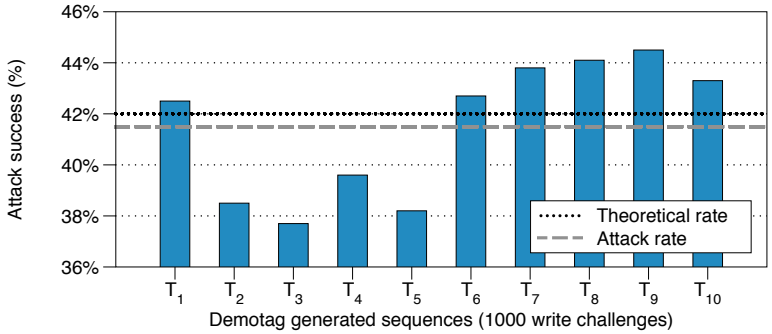


Figure 3.6: Che *et al.* PRNG attack for real Gen2 environment.

Che *et al.* PRNG implemented over the Demo Tag, and analyzing the resulting RN16s. More precisely, we show that simply collecting 128 bits (generated from a series of eight RN16s associated to each *write* challenge) is enough to obtain the feedback polynomial of the LFSR with a confidence of about 42%. This is consistent with the analytical results we anticipated in the previous section. Figure 3.5 (b) shows a simple example where six write cycles are captured. These captures allow us to collect 96 pseudo-random bits generated from the on-board Che *et al.* PRNG. If the two previously generated RN16s are added, the total amount of eavesdropped bits is 128. The sequences are parsed from the Matlab code that we feed with the serial interface output of the Demo Tag. Only reader-to-tag challenges are shown. The reader writes the EPC identification to 0 (cf. Equation 3.3), to obtain the RN16s directly from the ciphered data field of the *write* challenges.

The complete set of experiments that we summarize in Figure 3.6 consists of ten series of *write* commands. Each of these series generates a total of 1,000 *write* challenges that are sent from the A829EU reader

to the Demo Tag. As a result, 8,000 RN16s, i.e., 128,000 pseudo-random bits, are captured in total. These pseudo-random bits are computed from the Che *et al.* PRNG implementation deployed over the Demo Tag. Once stored, the pseudo-random sequences are processed by the Matlab code that contains the attack implementation. Let us recall that the attack applies the analysis of the linearity relation for each single *write* challenge. We show that the attack finds the appropriate feedback polynomials of the LFSR each 128 bits with a total ratio of success of 41.5%. This result is very close to the 42% that we predicted in Section 3.1.2. Therefore, we are able to confirm the vulnerability of the Che *et al.* PRNG for EPC Gen2 environments.

3.3 Empirical Analysis of Commercial EPC Gen2 PRNGs

The main security tool in the EPC Gen2 communication between readers and tags is the bitwise exclusive OR (XOR) *one-time-pad* encryption of specific *reader-tag* communication commands (cf. Section 2.1). To encrypt the information, EPC Gen2 tags are provided of an on-board pseudo-random number generator, transmitting 16-bit pseudo-random sequences to the *one-time-pad cover-coding* system (cf. Section 2.1). However, EPC Gen2 ICs manufacturers do not provide information about their pseudo-random number generators design [74]. Hence, the only references regarding this issue are the EPC Gen2 random number generation requirements defined in the EPC Gen2 standard [25].

The empirical analysis of the random sequences generated from EPC Gen2 commercial tags presented in this section is, to our best knowl-

edge, the first work done in this direction. Scientific literature and technical reports regarding real tags testing are orientated towards other goals and objectives.

Well known research areas involving commercial tags analysis are mostly related to the general performance of the tags. In this sense we can find works reporting read rates, reader-to-tag communication distance, read time analysis and antenna performance (e.g. [15, 32, 8, 82, 67]). Another research area related to the RFID performance is the analysis of read ratios of tags placed on specific objects or surfaces with bad properties for the radio-frequency communication [17].

EPC Gen2 protocol security issues is also subject of evaluation in the scientific literature. Research works evaluating the *kill command* activation [52, 93], or the use of TID (tag manufacturer information) numbers [55] are examples of empirical security research works involving EPC Gen2 technologies. Apart of EPC tags, empirical security analysis of other RFID technologies such as HF tags can be found in the literature [38, 30].

EPC Gen2 tag IC market is currently provided by five major manufacturers [55], only three of which have a representative piece of the market share [46]: *NXP* [70], *Alien* [95] and *Impinj* [43]. In this section we analyze the statistical properties of the EPC Gen2 commercial pseudo-random number generators based on the data extraction method introduced in Section 3.2.

3.3.1 Experimental Setup

The RN16 sequences sent between readers and tags in the EPC Gen2 technology are included in the communication protocol commands. In the identification stage, the tag uses the PRNG to generate RN16

sequences that are sent to the reader. If the reader acknowledges the RN16 sequence, the tag sends its full EPC identification. If the reader requests access to the reserved memory contents, or if he wants to modify the memory content, the tag generates RN16 to cipher the *kill* or *access* passwords, as well as to cipher the new memory contents for some commands (cf. Section 2.1). Note that the RN16 sequences are not a parameter to be set up by the user or the middleware. They are used in the lower layers of the communication. Therefore, there is no possibility to access the RN16 values from the reader software or middleware.

Eavesdropping Technique Description

In order to obtain the RN16 sequences from the communication between readers and tags, the IAIK UHF Demo Tag [3] introduced in Section 2.4.1 is used. The Demo Tags produced by IAIK TU Graz are programmed for delivery with an *ISO 18,000-6C Example Application*. This example application uses the functions included in the Demo Tag firmware to act like an EPC Gen2 reader, with some extended functionalities. Some of these functionalities are related to the UART module that enables the visualization of information through a serial terminal.

Two of the extended functions are used for the eavesdropping technique (cf. Section 3.2), the *Verbose Buffer* and the *Tag Normal / Silent Mode* [3]. The *Verbose Buffer* is a first-input first-output stack that stores the EPC Gen2 protocol commands exchanged between reader and tag. The buffer stores both the commands received from the reader and the tag responses, and it can be visualized through the serial terminal. The *Tag Normal / Silent Mode* allows the deactivation of the Demo Tag responses but without turning it off. This command is activated through the serial terminal.

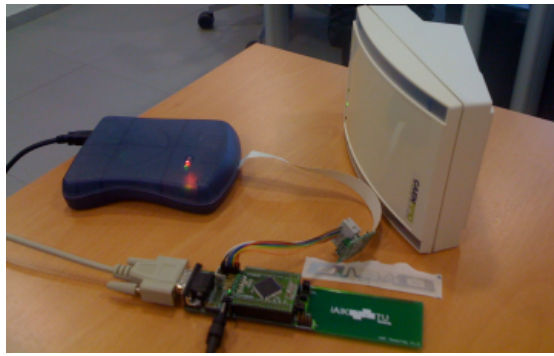


Figure 3.7: EPC Gen2 compliant ICs RN16s extraction method setup. The CAEN A829EU reader performs write queries to a commercial tag, and the *reader-tag* communication is eavesdropped by the IAIK Graz UHF Demo Tag.

The combination of these two extended functionalities allows us to use the Demo Tag to obtain the *reader-tag* communication between the reader and the commercial tags. Figure 3.7 shows the eavesdropping hardware setup. The CAEN A829EU reader is used to interrogate a commercial tag, and the Demo Tag (with the *Verbose Buffer* on and the *Tag Silent Mode* on) is placed in the vicinity of the communication range to eavesdrop the *reader-tag* commands. The reader writes the EPC identification to 0 as specified in Equation 3.3, to obtain the RN16s directly from the ciphered data field of the *write* challenges. In this way, the Demo Tag captures the queries from the reader but it does not generate responses, thus not affecting the communication between the CAEN A829EU reader and the commercial tag.

The eavesdropping software setup is shown in Figure 3.8. The CAEN A829EU Java software version has been modified to allow automatic consecutive queries from the reader. Some additional functions to interact with the UART module of the Demo Tag have been added.

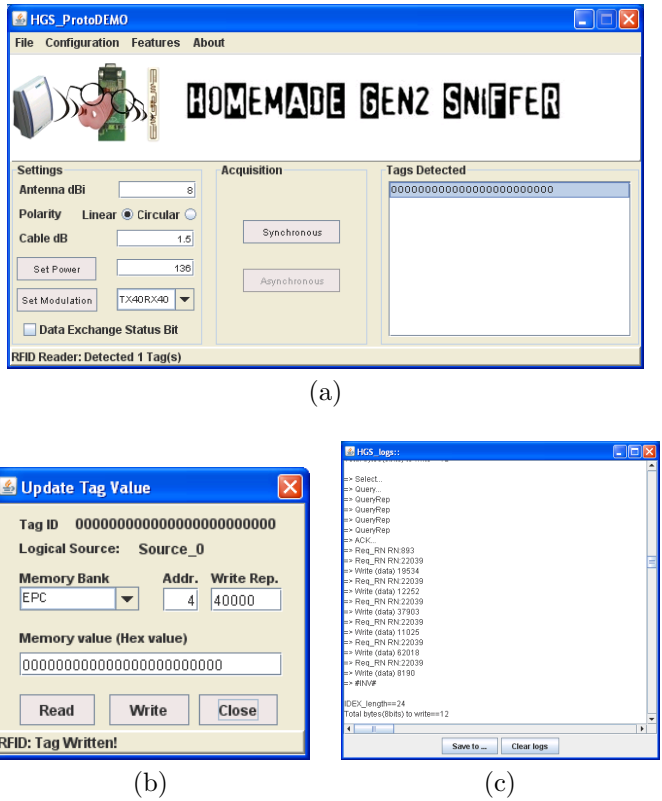


Figure 3.8: Screen captures of the software used for the RN16 extraction. In (a), we can see a screen capture of the CAEN A829EU Middleware, that has been modified to allow the automatic storage of RN16s. In (b), we can see the *Update Tag Value* screen ready to perform 40,000 *write* commands. In (c), we can see the IAIK Graz UHF Demo Tag extended *Verbose Buffer* with one write sequence eavesdropped from the *reader-tag* channel.

The *Tag Silent Mode* is activated and the *Verbose Buffer* is captured for each operation. A Matlab function [44] is later used to extract the random sequences from the rest of buffered data.

3.3.2 Commercial Analyzed ICs

EPC Gen2 Integrated Circuit manufacturers commercialize their ICs and inlay together (that is, the tag ready to work), or as single product to add to other inlay manufacturers. Hence, there are several different tags (even from different manufacturers) sharing the same IC.

EPC Gen2 commercial IC manufacturers bring information of some parameters of their products (cf. Table 3.5). They usually offer information regarding the physical properties of the IC such as wafer information, data retention cycle or working temperature range. Information about the memory mapping is usually provided, as well as information regarding the level of compliance with both the EPC Gen2 [25] and the ISO 18,000-6C [6] standards (e.g. memory mapping and optional commands implementation).

Precisely one of the parameters implicitly related to the accomplishment of the EPC Gen2 standard is the on-board pseudo-random number generation. PRNG is the basic security tool for the EPC Gen2 tags. Since the manufacturers declare the compliance of their products with the standard, their PRNG method satisfies the three requirements specified in the standard for that issue (detailed in Section 2.1).

To address the lack of information regarding this issue we have analyzed the PRNG behavior of three different IC models (NXP Ucode G2XL, Alien Higgs 3 and Impinj Monza 3) used in EPC Gen2 compliant commercial tags. Table 3.5 describes the parameters of the analyzed ICs, and Figure 3.9 shows the layout of three commercial tags using the analyzed ICs.

Table 3.5: Commercial ICs main parameters

IC model	Ucode G2XL	Higgs 3	Monza 3
Manufacturer	NXP	Alien	Impinj
EPC ID (bits)	96/240	96/496	96
TID (bits)	64	64	32
Access Pwd. (bits)	32	32	32
Kill Pwd. (bits)	32	32	32
User memory (bits)	No	512	No
EPC optional commands	Yes	Yes	Yes
Custom commands	Yes	Yes	Yes
Class 1 Gen2 compatible	Yes (v. 1.1.0)	Yes (v. 1.2.0)	Yes (v. unspecified)
ISO 18,000-6C compatible	Yes	Yes	Yes
Reading sensitivity	- 15 dBm	- 18 dBm	- 15 dBm
Release date	September 2006	April 2008	March 2008
Sample tag	Confidex Cassey [20]	Alien Squiggle [96]	TRACE Inlay [97]

(a)

(b)

(c)

Figure 3.9: Sample tags using the analyzed ICs. *UPM Short Dipole* (a) use NXP Ucode G2XL. *Trace Inlay* (b) use Impinj Monza 3. *Alien Squiggle* (c) uses Alien Higgs 3.

NXP Ucode G2XL

The main NXP IC product line compliant with ISO 18,000-6C or EPC Gen2 is the UCODE family for long range communication. Released in September 2006, the Ucode G2XL and G2XM ICs are currently the leading products for the NXP UHF RFID. The analyzed IC in this section is the Ucode G2XL [70]. G2XM IC is equivalent to the G2XL model but it also incorporates a 512-bit user memory bank. The ICs are EPC Gen 2 1.0.9 certified and compliant to EPC Gen2 1.1.0. UPM Short Dipole [81] and Confidex Cassey [20] are examples of EPC Gen2 commercial tags using the G2XL IC.

Alien Higgs 3

The Alien Technology market is focused on the UHF RFID technology, including readers, tags and ICs along with software solutions. The IC analyzed in this section is the latest Alien's IC for UHF RFID, the Higgs 3, which was preceded by the Higgs 2 IC. The Higgs 3 IC is compatible with the EPC Gen2 specifications (v. 1.2.0) and ISO 18,000-6C standard.

The Higgs family is the fourth generation of UHF RFID IC from Alien, being preceded by the Quark, Omega, and Lepton. Higgs 3 ICs are manufactured using CMOS process and EEPROM memory technology. Alien Squiggle or Short models [96]are examples of EPC Gen2 commercial tags using the Higgs 3 IC.

Impinj Monza 3

The Impinj market focus are the global supply chain management, asset tracking and identification applications. Impinj produces different RFID products like readers, antennas and tag ICs. In this section, the EPC Gen2 and ISO 18,000-6C Impinj Monza 3 IC is analyzed. Impinj has recently (2010) released a new version of the Monza series, the Impinj Monza 4, which adds user memory and custom commands capabilities. TRACE Inlay [97] or UPM Dogbone [80] are examples of EPC Gen2 commercial tags using the Monza 3 IC.

Random.org

Random.org [37] offers true randomly generated data through its web service since 1998. The randomness comes from atmospheric noise, compared with the PRNGs that uses deterministic algorithms to gen-

erate the randomness. The true random data from *Random.org* is used as a reference data to compare with the results obtained from the analyzed ICs.

3.3.3 Capture of Pseudo-random Sequences

The *write* command, used to write information in the tag memory, uses the one-time-pad cover-coding to cipher the new information to be written with the RN16s. Some tag ICs have relatively big user memory areas where a few hundreds of bits can be stored, but the user memory is not EPC Gen2 standardized. In fact nor the *NXP G2XL* neither the *Impinj Monza 3* IC are not provided with user memory. In order to equally compare the randomness of each IC we shall look for a compatible bit-extraction method for the three IC models. Since the 96-bit EPC identification memory is the largest memory area recognized by the EPC Gen2 Standard, it is chosen for the RN16 extraction method. Writing the full EPC identification memory area allows an equivalent number of generated RN16 for the three analyzed ICs, thus it is the most recommended for the pseudo-random generated bit extraction method.

The EPC Gen2 memory is organized in 16-bit blocks, and in the same way the PRNG of the EPC Gen2 tags shall provide the RN16. That is, six RN16s are used each time the EPC identification memory area is written (cf. Equation 3.3). Before these six RN16s, two additional RN16s are generated. One previous identifier to the EPC, and an acknowledgement of the *Access* stage. Thus at the end of the *write* process up to 8 RN16 have been generated, which means 128 bits.

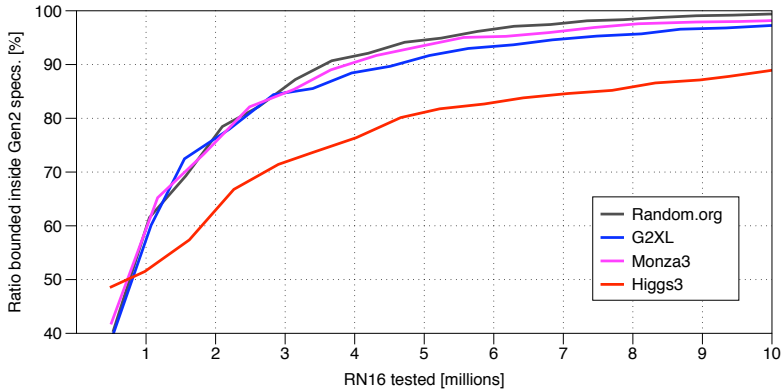


Figure 3.10: After 10 million of RN16s generated from EPC Gen2 commercial tags the fulfillment of the EPC Gen2 Standard's first requirement for random number generation is bounded between 89% and 98%

Albeit we cannot ensure the consecutiveness of the eavesdropped data. The same process is strictly applied to all the commercial tags to be able to compare the results in the same conditions.

3.3.4 Reference Data

A specific number of randomly generated bits shall be decided to eavesdrop to the commercial tags for the later analysis. Since the target of our analysis is the randomness of EPC Gen2 commercial tags, we take as a threshold decision the fulfillment of the first EPC Gen2 requirement for random number generation. The EPC Gen2 standard specifies that any single RN16 generated by an EPC Gen2 tag shall be bounded by $P_{min} = 0.8$ as the minimum probability; and $P_{max} = 1.25$ as the maximum probability, with respect to the mean of all the RN16 values (cf. Section 2.1).

Hence, while the rate parameter is under the specified fulfillment percentage, the data size to analyze S is increased with 16-bit sequences. *Random.org* [37] true random generation service is used to establish the data size for analysis, since the random sequences obtained from this service are truly random. We obtain that about 10 million of RN16 sequences (160 Mb) are necessary to reach a 99% of RN16 values ensuring the EPC Gen2 first requirement for random number generation. We consider this result as a proper size value for the evaluation of the commercial ICs.

In order to evaluate the commercial ICs PRNG we shall choose a tag using the specified IC. We generate 10 million RN16 sequences with the Confidex Cassey tag [20] for the G2XL IC, the Alien Squiggle tag [96] for the Higgs3 IC, and the TRACE Inlay tag [97] for the Monza3 IC.

Figure 3.10 shows the rate evolution of the *Random.org* truly random data and the RN16s generated by the analyzed ICs PRNGs, regarding the number of RN16 analyzed sequences. We can see that both *Monza3* and *G2XL* random generator rates grow parallel to the *Random.org* reference with approximately a 1% of difference between them for larger values of analyzed RN16s. For 10 million analyzed sequences both rates are close to the 99% reached by the *Random.org* sequences. For the same number of analyzed RN16 sequences the *Higgs3* random generator reaches a 89% of sequences within the specified boundaries [25]. Table 3.6 shows a resume of the ICs PRNGs properties for 10 million RN16 sequences. P_{max} and P_{min} refer in the table to the maximum and minimum probability of RN16s, with respect to the mean of all the RN16 values.

EPC Gen2 PRNGs main goal is to provide sequences of random data to be used as a cipher keystream. Since the PRNGs are devoted to

security tasks it is important to bring good statistical properties, as well as forward unpredictability. In the next section we analyze the statistical properties of the EPC Gen2 PRNGs generated datasets, and the *Random.org* data.

3.3.5 Statistical Evaluation

Since the main property of a PRNG is to ensure the forward unpredictability of its generated sequences, the correctness of a PRNG can be measured with statistical tests applied to the output sequence. We use the National Institute of Standards and Technology (NIST) test suite to evaluate the randomness deviations of a binary random sequence [4] (cf. Section 2.4). Tests are applied to the reference *Random.org* sequences and to the three analyzed EPC Gen2 compliant ICs (*NXP G2XL*, *Impinj Monza 3* and *Alien Higgs 3*). For each analyzed IC we have analyzed two sequences of 10 million RN16s, generated from the same tag. Test results are presented in Table 3.7. The NXP Ucode G2XL IC is represented hereinafter as *G2XL*, the Alien Higgs 3 IC is represented hereinafter as *Higgs3* and the Impinj Monza 3 IC is represented hereinafter as *Monza3*.

Table 3.6: Impinj Monza 3 brings the closest result to the true random sequences (*Random.org*) after 10 million sequences

PRNG Generator	Random.org	NXP G2XL	Alien Higgs 3	Impinj Monza 3
Rate (%)	99.45	97.35	89.40	98.67
P_{max}	1.33	1.50	16.36	1.43
P_{min}	0.69	0.62	0.63	0.65

Table 3.7: Commercial EPC Gen2 labels fail to pass specific NIST tests

Runs	100							
PRNG	Random.org		G2XL		Monza3		Higgs3	
Sequence	I	II	I	II	I	II	I	II
Frequency	1.00	1.00	**0.30**	**0.39**	**0.34**	**0.38**	0.96	0.99
BlockFrequency	0.98	0.99	**0.95**	0.98	0.97	0.97	**0.56**	**0.52**
Runs	1.00	1.00	**0.50**	**0.66**	**0.66**	**0.66**	0.99	0.98
LongestRun	1.00	1.00	0.98	1.00	1.00	0.99	1.00	0.99
Rank	0.99	0.99	0.99	0.98	0.98	1.00	1.00	1.00
OverlappingTemplate	0.99	1.00	0.99	0.98	0.99	0.99	**0.87**	**0.95**
Universal	1.00	0.99	0.98	0.99	1.00	1.00	0.98	0.97
ApproximateEntropy	1.00	1.00	0.98	0.98	1.00	0.98	**0.00**	**0.00**
LinearComplexity	0.96	0.98	0.98	1.00	1.00	0.98	0.99	1.00
CumulativeSums	$\frac{2}{2}$	$\frac{2}{2}$	$\mathbf{\frac{0}{2}}$	$\mathbf{\frac{0}{2}}$	$\mathbf{\frac{0}{2}}$	$\mathbf{\frac{0}{2}}$	$\mathbf{\frac{1}{2}}$	$\frac{2}{2}$
NonPeriodicTemplate	$\mathbf{\frac{146}{148}}$	$\frac{148}{148}$	$\mathbf{\frac{139}{148}}$	$\mathbf{\frac{144}{148}}$	$\mathbf{\frac{142}{148}}$	$\mathbf{\frac{141}{148}}$	$\mathbf{\frac{111}{148}}$	$\mathbf{\frac{126}{148}}$
RandExcursions	$\frac{8}{8}$	$\mathbf{\frac{7}{8}}$	$\frac{8}{8}$	$\mathbf{\frac{5}{8}}$	$\mathbf{\frac{6}{8}}$	$\mathbf{\frac{7}{8}}$	$\frac{8}{8}$	$\frac{8}{8}$
RandExcursionsVar	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\mathbf{\frac{10}{18}}$	$\mathbf{\frac{13}{18}}$	$\mathbf{\frac{17}{18}}$	$\frac{18}{18}$	$\frac{18}{18}$
Serial	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\mathbf{\frac{0}{2}}$	$\mathbf{\frac{0}{2}}$

For each test, 100 stream runs are executed. Tests refusing the randomness hypothesis, that is, performing a proportion under 0.96 (cf. Equation 2.12) of tests over the significance level are denoted with bold letters in the table (see Section 3.1 and Section 2.4 for details regarding the NIST tests). The results demonstrate the good statistical properties of the *Random.org* sequences, since only one over 8 RandomExcursions test, and two over 148 NonPeriodicTemplate tests fail, over the two analyzed sequences. The commercial ICs PRNGs fail specific tests. *NXP G2XL* fails the *Frequency*, the *Runs* tests, the *CumulativeSums* test and partially fails the *BlockFrequency* test. *Impinj Monza3* fails the *Frequency*, the *CumulativeSums* tests and the *CumulativeSums* test. *Alien Higgs 3* fails the *BlockFrequency* test, the *OverlappingTemplate*, the *ApproximateEntropy* and *Serial* tests.

It is worth mention that the pseudo-random nonces are generated by the IC, but since the PRNG mechanism is not revealed by the IC manufacturers we do not know the inlay influence in the RN16s generation. For this reason, we tested additional sequences generated from different inlays and tag models but using the same IC. Statistical tests offer similar results to the ones shown in Table 3.7. Hence, our experimental tests show similar statistical properties for each specific IC, regardless of the analyzed inlay.

Tests Discussion

Results show evidences of statistical deviations in the generated random sequences from EPC Gen2 commercial ICs. The *Frequency* test is the reference test for the subsequent statistical test suite. If the *Frequency* test fails, the likelihood of other tests failing is high [4]. This is the case for the *G2XL* and *Monza3* analyzed sequences, which are under the significance level for this specific test. Hence, *G2XL*

and *Monza3* pseudo-random generated data have a gap with respect to the number of 1's and 0's binary representation (supposed to be the same for a random sequence). This is an evidence for the existence of non-randomness in the analyzed sequence. To dimension this statistical deviation we have sized the number of 0's and 1's for each analyzed sequence. As it can be seen in Table 3.8 *G2XL* and *Monza3* show the widest gap with respect to the number of both symbols in the sequence. Specifically, the PRNGs associated to these ICs generate about a 0.59% more of 0's than 1's.

The *Runs* test also fails for the *G2XL* and *Monza3* sequences. The focus of this test is the number of runs in the sequence, where a run is an uninterrupted sequence of identical bits. In particular, these tests determines whether the oscillation between different lengths of runs is likely to belong to a random sequence. Thus, the *G2XL* PRNG tends to generate runs of bits statistically deviated from what is expected from a random sequence.

The *CumulativeSums* test fails for *Monza3* PRNG, and for *G2XL* PRNG. This test focuses on cumulative sums of binary sequences, where $1 = +1$ and $0 = -1$. For a random sequence, the excursions of the cumulative sums should be near zero. Hence, the result of this test is another evidence of the gap in the number of 0's and 1's from the analyzed sequences failing this test.

Table 3.8: G2XL and Monza3 PRNGs generate more approximately 0.59% more of 0's than 1's

Sequences length	10 million RN16			
PRNGs	Random.org	G2XL	Monza3	Higgs3
$\frac{\text{Num. } \mathbf{0's} - \text{Num. } \mathbf{1's}}{\mathbf{1.6 \cdot 10^8}}$	0.018%	0.590%	0.593%	- 0.050%

Evidences of non-randomness are also found in the *Higgs3* pseudo-random generated sequence. Specially relevant is the *ApproximateEntropy* tests result, where none of the runs are over the significance level. In this case the *ApproximateEntropy* test focuses on the frequency of all possible overlapping m-bit patterns (for $m = 10$) across the entire sequence, comparing the frequency of overlapping blocks of two adjacent lengths (m and $m + 1$) against the expected result for a truly random sequence.

Higgs3 PRNG fails to pass two other statistical tests. The *OverlappingTemplate* test also measures how well pre-specified pattern strings match the expected number of these patterns under the assumption of randomness. The *Serial* test is similar to the Frequency test but using different uniform m-bit patterns.

Security Weaknesses Associated to the Tests Results

If a frequency analysis of the obtained RN16 values is performed, we may expect a margin of repetitions centered at an average of 152 repetitions for each of the 10 million analyzed RN16 values. Based on the EPC Gen2 standard first requirement for random number generation the maximum repetition frequency shall be approximately 190 ($P_{max} = 1.25$) and the minimum repetition frequency shall be approximately 122 ($P_{min} = 0.8$). Figure 3.11 shows the RN16 frequency analysis. The horizontal axis represents the RN16 values (from zero to $2^{16} - 1$) and the vertical axis represents the number of repetitions for a dataset size of 10 million RN16 sequences.

We can observe that *Higgs3* PRNG significantly exceed the number of repetitions for some specific RN16 values with regard to the parameters allowed by the EPC Gen2 standard. The appearance of *peaks* for some specific RN16 values (over 2,000 repetitions for some

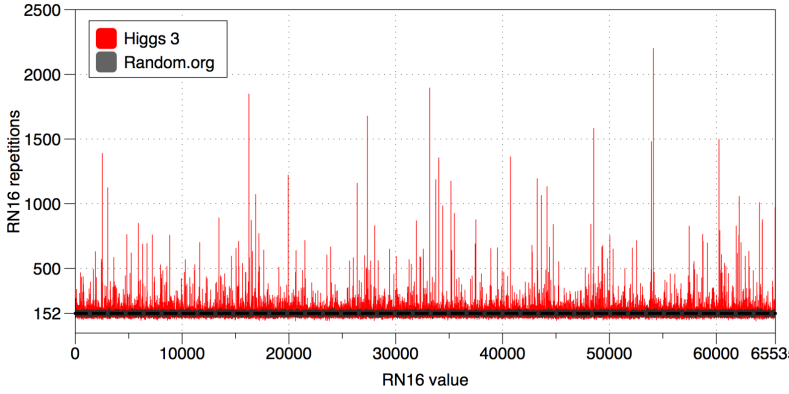


Figure 3.11: Frequency analysis for individual RN16 values show consistency with previous results obtained from statistical tests

cases) generates a surplus of bit-patterns for the most repeated RN16 values, and consequently other bit-patterns are less represented in the binary sequence. This fact is consistent with the previously obtained statistical analysis results, where the *Higgs3* sequence failed to pass tests regarding the matching frequency with pre-established bit patterns (*Overlapping Template* and *Approximate Entropy*), compared with the values expected from a random sequence.

Figure 3.12 focuses in the RN16 frequency analysis of *G2XL* and *Monza 3* PRNGs. The black dashed lines represent the frequency mean of single RN16 appearances. We can observe that there are two different frequency means. For the first 2^{13} RN16 values, the mean value is placed at approximately 169 repetitions. For the rest of the RN16 values the mean is placed at 150 repetitions. Based on this information we can observe that the first 2^{13} RN16 values are a 12.7% more likely to appear than the rest of RN16 values for the *G2XL* and *Monza3* ICs PRNGs.

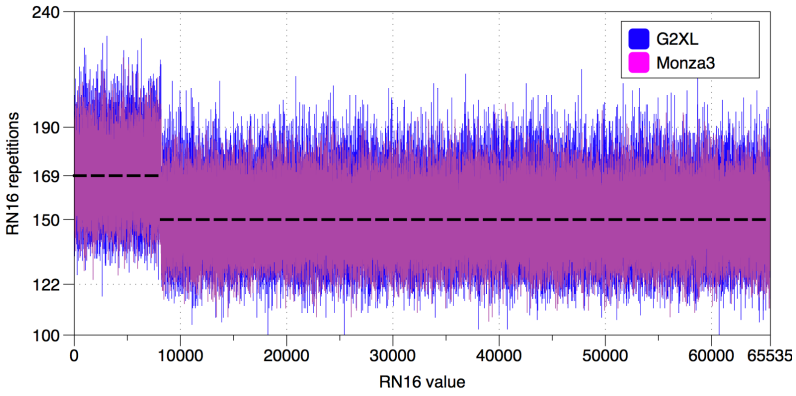


Figure 3.12: *G2XL* and *Monza 3* shows a different mean frequency value for approximately the first 2^{13} RN16 values

Based on the statistical analysis the *G2XL* and *Monza3* appearance deviation for the first 2^{13} RN16 values is consistent with the *Frequency (Monobit)* failed test results. *G2XL* and *Monza3* analyzed binary sequences show a 0.59% more of 0's than 1's. This result show evidences that the first 2^{13} RN16 values (where three first bits are always 0) are repeated 12.7% more than the rest of RN16 values.

The goal of a cryptography-based PRNG is to be unpredictable and irreproducible [61]. The NIST statistical tests [4] applied to the sequences generated by the analyzed ICs PRNGs (*G2XL*, *Monza3* and *Higgs3*) give evidence of statistical deviations, associated to the hypothesis of non-randomness. Specific tests such as the appearance frequency (Figures 3.11 and 3.12) of each RN16 value are consistent with the statistical tests results, demonstrating that even being compliant with the EPC Gen2 requirements for random number generation, certain RN16 values have a significantly higher frequency of appearance compared with the mean values.

3.3.6 Analysis Conclusions

Thanks to a novel bit-extraction method using the Demo Tag [3] we have been able to obtain information from the reader-to-tag communication using the EPC Gen2 protocol. The obtained data are the RN16 sequences generated from the PRNGs which EPC Gen2 tags implement on-board. We have focused our analysis on the *NXP G2XL*, *Alien Higgs 3* and *Impinj Monza 3* ICs, using truly random data sequences from *Random.org* as a reference data.

10 million RN16 sequences (160 Mb), are obtained from each commercial IC. These sequences have been analyzed with the NIST statistical test suite for random and pseudo-random number generators for cryptographic applications [4]. Since evidences of statistical deviations for the analyzed data have been found, a new analysis focused on the RN16 values frequency is performed. From these analysis, we have observed statistical deviations on the analyzed PRNGs, generating specific RN16 values more likely than others. Even meeting the EPC Gen2 Standard requirements, this vulnerability can lead to PRNG prediction threats.

From our evaluation, we conclude that the EPC Gen2 first requirement for random data generation must be revised. Indeed, since the probability boundaries depend on the number of analyzed RN16s, we propose to specify a new probability boundaries range linked to a specific number of generated RN16 sequences, similarly to the boundaries obtained from the analysis of *Random.org* and commercial ICs. In Figure 3.13 we represent the P_{max} and P_{min} for each of the analyzed sequences, with respect to the number of generated RN16s. The plot specifies the EPC Gen2 probability boundaries defined at $P_{min} = 0.8$ and $P_{max} = 1.25$. Notice that the longer the number of analyzed sequences, the closer the results to these boundaries. Specifically,

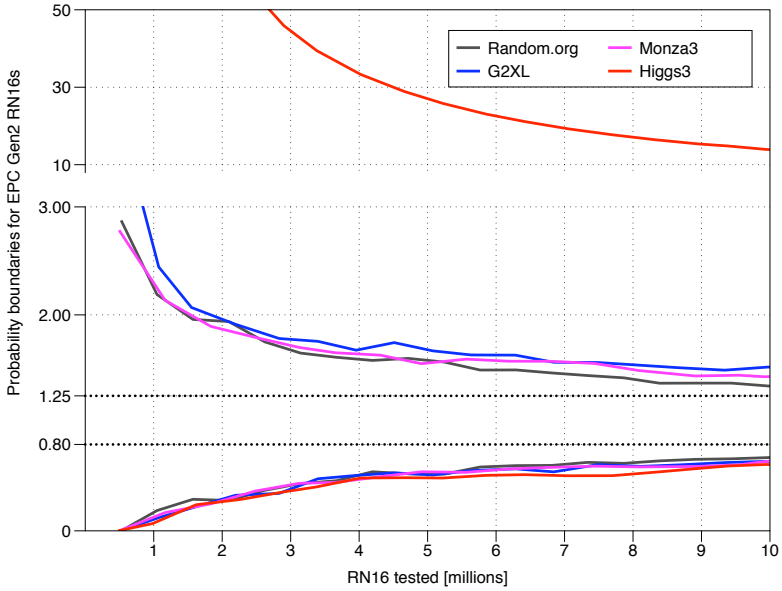


Figure 3.13: P_{max} and P_{min} approach to the EPC Gen2 Standard requirements with a large quantity of analyzed RN16s

the suitable probability boundaries for 10 million analyzed sequences would be around $\frac{0.69}{2^{16}} < P_{10M}(RN16) < \frac{1.33}{2^{16}}$.

3.4 Chapter Summary

Pseudo-random number generators are the crucial components that guarantee the confidentiality of EPC Gen2 [25] RFID communications. In this chapter, we have described the problems of using linear feedback shift registers as underlying mechanisms for the implementation of low-cost PRNGs. Without appropriate measures that increases their complexity, the linearity of LFSR-based PRNGs lead to

insecure implementations. We have analyzed a cost-effective PRNG proposal for EPC Gen2 devices presented by Che *et al.* [18]. The proposal combines thermal noise signal modulation and an underlying LFSR. We have indeed demonstrated that the proposal does not handle the inherent linearity of the resulting PRNG properly.

We have described an attack to obtain the feedback polynomial function of the LFSR. This allows us to synchronize and to predict the resulting sequences generated by the Che *et al.* PRNG. We have presented the implementation of a practical attack in a real EPC Gen2 scenario. By means of a compatible Gen2 reader, and a programmable Gen2 tag [3] implementing the Che *et al.* PRNG, we have shown that an attacker can obtain the PRNG configuration with a confidence of 42% by only eavesdropping 128 bits of pseudo-random data (that can be obtained with only one command). Although the attack implementation has been applied to a specific PRNG proposal, the procedure used to obtain the data is based on standard EPC Gen2 commands and it can be applied to any EPC Gen2 tag communication to eavesdrop the output of the PRNG.

Using the eavesdropping technique designed for the attack to Che *et al.* scheme, three different EPC Gen2 compliant ICs have also been evaluated. *NXP Ucode G2XL*, *Alien Higgs 3* and *Impinj Monza 3*. Their underlying randomness has been evaluated by analyzing the statistical properties of their PRNGs with the NIST statistical test suite. Statistical evidence of non-randomness in the analyzed sequences has been found.

Our main contributions include:

- A new method for extracting RN16 sequences from the reader-to-tag EPC Gen2 communications using a Real EPC Gen2 setup.

- A linearity vulnerability found in a PRNG proposal by Che *et al* [59].
- An attack to the Che *et al.* PRNG proposal to synchronize its output with a 42% confidence by simply eavesdropping 128 generated bits.
- An analysis of EPC Gen2 commercial ICs pseudo-random number generators based on the NIST statistical test suite for randomness, finding evidence of non-randomness in the analyzed sequences.
- A new proposal for the measurement of the first requirement for random number generation regarding the number of analyzed sequences.

4

New PRNG Proposal

A new pseudo-random number generator (PRNG) scheme is introduced in the remainder of this document. Our scheme follows the requirements established in Chapter 2 for resource constrained devices, and has been specifically designed to meet the Electronic Product Code Class 1 Generation 2 (EPC Gen2) specifications. Several PRNGs specifically designed for resource constrained devices have been presented in the state of the art (Chapter 2). An attractive approach intrinsic to these proposals relies on the use of linear feedback shift registers (LFSRs). The simplicity of using LFSRs and their low hardware complexity, while still providing efficient statistical properties, guarantees the ease and efficient adaptation of these schemes to meet the requirements of the EPC Gen2 specifications. However, without the appropriate handling of the inherent linearity of the LFSR circuitry, the resulting schemes might still lead to insecure designs. This is the case of the approach described by Che *et al.*

in [18], which consists on a PRNG specifically designed for EPC Gen2 tags. The Che *et al.* scheme combines a 16-bit LFSR and a physical source of randomness perturbing the LFSR linearity. However, the approach is insecure, as it is shown in Section 3.1.

The PRNG design proposed in this chapter, is based on a 16-bit LFSR that contains multiple feedback primitive polynomials fed by a physical source of randomness [60].

4.1 High Level Description

The combination of deterministic modifications of LFSRs is useful for keystream generators where *sender* and *receiver* can share a secret k as a key for the PRNG one-time pad communications. On the contrary, in the specific communication model of EPC Gen2 systems, *sender* and *receiver* do not share any secret k. Instead, the low-power tag-to-reader communication is used to transmit in plain text the nonces to be used as a keystream as depicted in Figure 4.1. This allows other strategies for the linearity avoidance of LFSRs, while maintaining the simplicity of a single LFSR. This is the case of the Che *et al.* scheme [18], regardless of its vulnerability.

In our proposal, randomness is used in a different way than the Che *et al.* scheme in order to truly mask the linearity of the LFSR. Our proposal successfully handles the vulnerabilities found in the Che *et al.* scheme [59]. Our strategy can be seen as a *keystream generator with memory*, in the sense of linearity stream modification.

For the construction of our generator, we decide to use the same strategy taken in the Che *et al.* scheme, which consists on perturbing the LFSR output by a physical source of randomness. Similarities of

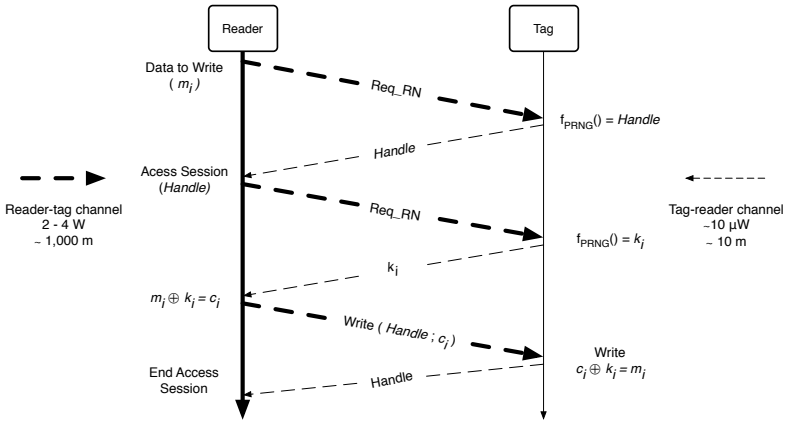


Figure 4.1: EPC Gen2 keystream supply strategy. Bold lines denote the long reader-to-tag signal. Thin lines denote the short tag-to-reader signal, which transfers the keystream to the reader

our scheme with the one of Che *et al.* end here. We use the trn bits to modify the characteristic polynomial of the LFSR rather than the LFSR output.

A first idea is to replace the static feedback polynomial (cf. Equation 4.1) with a dynamic one, that depends on the true random data (cf. Equation 4.2) where only the most significant cell is always switched on to set the function to n_{th} order. However, such an approach does not produce a good pseudo-randomness output sequence since not all feedback polynomials randomly generated are primitive. Feedback polynomials of a LFSR must be primitive to guarantee good pseudo-random properties (cf. Section 2.2). Using primitive polynomials as feedback polynomials must, therefore, be enforced.

$$C(x) = 1 + c_1 x^1 + c_2 x^2 + \cdots + c_n x^n \tag{4.1}$$

$$C(x) = 1 + (trn_j)x^1 + (trn_{j+1})x^2 + \cdots + (trn_{j+n})x^{n-1} + x^n \quad (4.2)$$

Taking different primitive polynomials as the feedback polynomial of a LFSR has already been used in non-security related scenarios. In [40], for instance, Hellebrand *et al.* apply this technique for the design of *Built-In Self Tests* (BIST) operations. These operations are intended for testing chip designs, generating test vectors and evaluating test responses. The use of several polynomial configurations are applied to the LFSR, depending on an input parameter. To do so, the so-called multiple-polynomial LFSR contains a programmable feedback function that solves the linear dependency problem by choosing between different LFSR configurations for the encoding of each test. The resulting construction guarantees complete fault coverage tests while minimizing test application time, test overhead and data storage [88]. Hellebrand *et al.* show in their work that, despite the simplicity of their design, the obtained generator offers remarkably sound statistical properties and a long output period.

Following the Hellebrand *et al.* design, we construct our generator as a multiple-polynomial LFSR. We, therefore, substitute the static feedback polynomial configuration by a multiple feedback primitive polynomials configuration. The different feedback primitive polynomials are connected to the LFSR module through a *decoding matrix unit*, which is in charge of selecting each single feedback polynomial. After a number of LFSR cycles, the *Polynomial Selector* shifts its position towards a new feedback polynomial configuration, following a *wheel* behavior as depicted in Figure 4.2. The number of shifts, and so, the corresponding selection of each primitive polynomial at a certain LFSR cycle, is determined by the *trn* bit derived from the

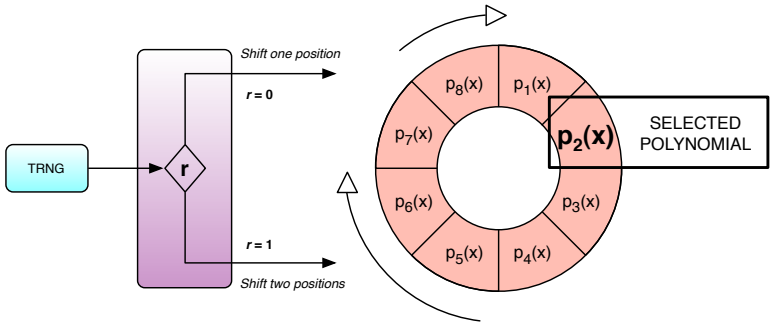


Figure 4.2: Polynomial Selector function. The polynomial *wheel* shift one or two positions regarding the true random value r

physical source of randomness. If the *trn* bit turns out to be set to 0, the *Polynomial Selector* shifts the *wheel* one position. Otherwise, if the *trn* bit turns out to be set to 1, the *Polynomial Selector* shifts the *wheel* two positions. The way how the *trn* bit decides the selection of the following feedback polynomial avoids, moreover, two consecutive selections of the same feedback polynomial.

Besides the multiple-polynomial selection to break the LFSR inherent linearity, the proposed PRNG is also designed to discard bits from the LFSR prior to output them as the pseudo-random sequence (RN16). The aim of this measure is to increase the flexibility of the system and to reduce the predictability of the generated sequences.

4.2 Logical Components

The main challenge to obtain an efficient PRNG is how to design the pseudo-random generation scheme in order to achieve the best efficiency with the least computational complexity. We look for hard-

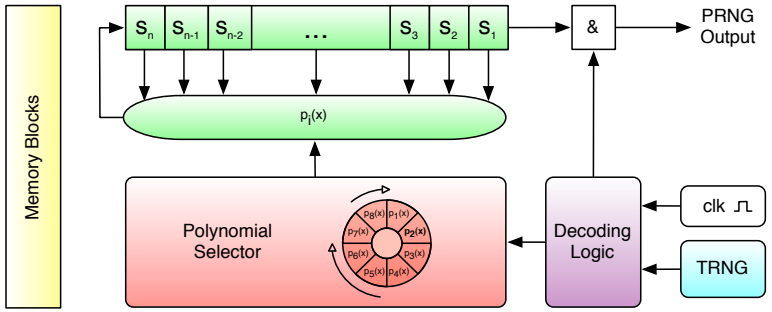


Figure 4.3: Block diagram of our PRNG proposal

ware highly efficient components with desirable cryptographic properties, e.g., boolean operators (XOR, AND, OR), and flip-flop registers. In the remainder of this section we introduce the internal modules which enable the PRNG performance, as well as the parameters characterizing our proposal.

4.2.1 Internal PRNG Modules

The main idea of our PRNG proposal is the generation of pseudo-random data from a LFSR perturbed by a source of truly generated random bits. The whole design of our PRNG proposal can be divided in *LFSR*, *Polynomial Selector*, *Decoding Logic*, *Thermal-noise TRNG*, *Memory* and *Storage Control*. Figure 4.3 shows a block diagram description of the proposed PRNG.

Linear Feedback Shift Register

Our proposal relies on a LFSR core (green block in Figure 4.3) perturbed by a physical source of randomness. We keep the LFSR core

for several reasons. LFSR schemes are very fast and efficient in hardware implementations as well as simple in terms of computational requirements [61], as specified in Section 2.2. This makes the use of LFSRs an ideal system for both energy and computational constrained environments. Moreover, a LFSR follows the same hardware scheme as cyclic redundancy check (CRC) functions. These functions are included in the EPC Gen2 standard [25]. Therefore, current EPC Gen2 tags (which include CRC operations) are able to execute LFSR-based functions in the same hardware.

Input: The LFSR input comes from the feedback function generated by the Polynomial Selector. Hence, the feedback polynomial taps are implemented in the Polynomial Selector, including those which are always enabled. This is the case of the S_1 tap, which determines the polynomial function to the n_{th} order. The shift cycles are determined by the Decoding Logic which manages the internal clock.

Output: The Decoding Logic unit manages to discard d bits each n LFSR shifts, that is, for each full LFSR cycle. Hence, the output of the LFSR module are the PRNG bits prior to the bit-discarding stage. The LFSR module also serves the tap bits to the Polynomial Selector, used to compute the feedback bit.

Parameters: The main parameter of the LFSR block is:

- Size of the LFSR = n bits

Polynomial Selector

The Polynomial Selector (red block in Figure 4.3) is the responsible of the linearity avoidance in our scheme. As introduced in the High Level Description (cf. Section 4.1), the use of primitive polynomials in the system is enforced to obtain the optimal period and statistical properties.

A set of primitive feedback polynomials is selected and implemented in the Polynomial Selector, and each single feedback polynomial is used depending on the value of the truly random bit provided by the TRNG module. We recall that the feedback polynomials are implemented as a *wheel*, which rotates depending on the bit value given by the TRNG module as depicted in Figure 4.2. If the truly random bit is a logic 0, the *wheel* rotates one position, that is, it selects the next feedback polynomial. Instead, if the truly random bit is a logic 1, then the *wheel* rotates two positions, that is, the Polynomial Selector jumps one feedback polynomial and selects the next one.

Input: There are two Polynomial Selector inputs, the Decoding Logic and the LFSR taps. The Decoding Logic is in charge of the rotation of the polynomials *wheel*, that is, the feedback polynomial selection. Furthermore, the LFSR taps provide the information to generate the new feedback bit to be stored in the most significant bit (MSB) of the LFSR.

Output: The Polynomial Selector output is the LFSR feedback bit, generated from the selected polynomial function.

Parameters: The main parameters of the Polynomial Selector block are:

- Number of implemented feedback polynomials on tag = m

Decoding Logic

The Decoding Logic (purple block in Figure 4.3) is the responsible of managing the internal PRNG clock. It activates and deactivates the PRNG modules for its proper performance. The internal PRNG modules have different activation and deactivation timings. Depending on the internal clock frequency, some modules such as the LFSR or the TRNG need different activation cycles. For example, the trn sampling in the TRNG module is activated only once for each PRNG output (16-bit pseudo-random sequence).

The Decoding Logic also manages the truly random bit (trn) obtained from the TRNG module, rotating the Polynomial Selector with regard to the trn value. This action is performed every l LFSR shifts (where $l < n$), to avoid pseudo-random sequences generated from a single feedback polynomial. Finally, the Decoding Logic is in charge of discarding a specific number of bits from the LFSR. That means the PRNG output might not be the same as the LFSR output because some bits are dropped.

Input: The Decoding Logic has two data inputs, the internal clock and the truly random bit obtained from the TRNG module. The internal clock is obtained from the integrated circuit (IC), being used to activate and deactivate the different blocks composing the PRNG. The TRNG module also supplies the truly random bit value, modi-

fying the feedback polynomial selection. Furthermore, the Decoding Logic would be the receiver of the IC signal requesting a new RN16.

Output: The output clock signals are conducted to the Polynomial Selector, the LFSR shifter and the LFSR output.

Parameters: The main parameters of the Decoding Logic block are:

- Internal clock frequency = f_{clk}
- Number of LFSR discarded bits = d
- Polynomial Selector update period = $l < n$

Thermal-noise TRNG

Regarding the physical source of true randomness (*trn*), there are different proposals to derive true random sequences of bits from the hardware of a radio-frequency identification (RFID) tag. Some examples of on-tag *trn* acquisition are, for instance, taking advantage of thermal noise [18], high frequency sampling [23, 101] or fingerprint data [42] in circuits. Some commercial tags include, moreover, some extra functionalities (e.g., *received signal strength indicator*, RSSI [65]) that can be useful for *trn* addition techniques.

The chosen technique to include in our design is the oscillator-based high frequency sampler from Che *et al.* design [18] (blue block in Figure 4.3), due to its simplicity and suitability for low-resourced designs. The output of the TRNG is fed to the Decoding Logic, which in turn, manages the Polynomial Selector.

Input: The input parameter is the high-frequency oscillation waveform obtained from the RF front-end, sampled by a low-frequency waveform generated from the thermal noise, which produces truly random bits.

Output: The output is the truly random generated bit that is fed to the Decoding Logic block.

Parameters: The main parameter of the TRNG block is:

- *trn* sampling frequency = f_r

Memory and Storage Control

Due to the nature of our proposal, a few bits of memory storage are necessary to ensure good statistical properties (yellow block in Figure 4.3). The linear feedback shift register state is stored after a RN16 generation to be fed as LFSR seed for the next random sequence generator. Furthermore, the Polynomial Selector state needs also to be stored to avoid a possible repetition of the feedback polynomial function, and to maintain the feedback polynomial switching cycle l.

Both input and output of the Memory Blocks are, hence, the LFSR and the Polynomial Selector register's state. The memory block size will be defined by the specific hardware implementation of the PRNG.

4.2.2 System Parameters Summary

Each of the internal modules described so far perform specific actions for the correct generation of pseudo-random bits, and their parameters are resumed in Table 4.1. The design parameters define the

performance and implementation of the whole system. The security of the proposed scheme depends on the *public* or *private* characteristic of the mentioned parameters, as well as the value of the variables used in them. A security evaluation is presented later in Chapter 5.

Since our design is based on a deterministic algorithm (perturbed by a source of true randomness), the size of the components and their design are crucial for the statistical behavior and, therefore, the security of the whole system. Regarding that our PRNG proposal is designed to act as a cipher tool on a RFID tag, its aim is to provide a secure enough communication link. That is, good randomized sequences to avoid unauthorized readers to obtain the encrypted data. Table 4.1 resumes the design parameters for our proposed scheme.

4.3 Detailed PRNG Execution

After the high level description of our PRNG proposal (cf. Section 4.1) and the description of the logical components of the system (cf. Section 4.2), we show an example of the proposed PRNG internal execution. The goal of this section is to provide better comprehension of the PRNG performance details.

Table 4.1: Design parameters summary

Size of LFSR (bits)	n
Number of feedback polynomials on tag	m
Internal clock frequency	f_{clk}
trn sampling frequency	f_r
Polynomial Selector update period	$l < n$
Number of LFSR discarded bits	d

Previous to the PRNG execution example we shall determine specific values for the parameters detailed in Table 4.1. Specifically, the size of the LFSR n, the Polynomial Selector update period l, and the number of discarded bits d are necessary.

Certain parameters shall be defined for this example. Briefly stated, the given values are: size of LFSR $n = 16$, the Polynomial Selector update period is set to $l = 15$, the system does not discard bits ($d = 0$), and the feedback polynomials $\mathcal{P}_{\text{sel}}$ are defined in Table 4.2. Furthermore, some elements of the scheme need a variable assignment. This is the case of the LFSR initial value v_0 and the true random bit value r. To illustrate the PRNG performance two updates are presented. For the first one, the initial value is $v_0 = 0001_h$ (which represents a logical 1 in the less significant bit, in hexadecimal notation) and the trn bit value we use is $r = 0$. The second PRNG update uses the last LFSR state as initial value (v_1) and $r = 1$ as a trn value.

Table 4.2: Example of feedback polynomials ($n = 16$)

Primitive polynomials
$p_1(x) : 1 + x^5 + x^6 + x^{11} + x^{16}$
$p_2(x) : 1 + x + x^5 + x^6 + x^7 + x^{11} + x^{16}$
$p_3(x) : 1 + x^4 + x^5 + x^6 + x^7 + x^{11} + x^{16}$
$p_4(x) : 1 + x + x^3 + x^4 + x^5 + x^6 + x^7 + x^{11} + x^{16}$
$p_5(x) : 1 + x^3 + x^5 + x^6 + x^{10} + x^{11} + x^{16}$
$p_6(x) : 1 + x^5 + x^6 + x^{10} + x^{11} + x^{13} + x^{16}$
$p_7(x) : 1 + x^4 + x^5 + x^6 + x^{10} + x^{11} + x^{16}$
$p_8(x) : 1 + x + x^3 + x^4 + x^5 + x^6 + x^{10} + x^{11} + x^{16}$

Table 4.3 details the LFSR state for 16 shift cycles including the 16 outputted PRNG bits (column Tx) consisting in a PRNG update. We assume that the LFSR initial value (seed) is the last LFSR update using the feedback polynomial P_1, hence, a new feedback polynomial should be used prior to the first LFSR shift of this PRNG update. Since TRNG module transfers a bit with value $r = 0$ to the Decoding Logic module, a consecutive (but different) feedback polynomial is selected in the Polynomial Selector module, that is, P_2 and P_3. This

Table 4.3: LFSR iteration example ($r = 0$)

	Seed v_0																
	S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	S_7	S_6	S_5	S_4	S_3	S_2	S_1	**Tx**
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	
$p_2(x)$	1	0	0	0	1	1	1	0	0	0	1	0	0	0	0	1	
1:	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
2:	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**
3:	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**
4:	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	**0**
5:	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	**0**
6:	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	**0**
7:	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	**0**
8:	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	**0**
9:	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	**0**
10:	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	**0**
11:	1	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	**0**
12:	1	1	1	0	1	0	0	1	1	1	1	1	0	0	0	0	**0**
13:	1	1	1	1	0	1	0	0	1	1	1	1	1	0	0	0	**0**
14:	1	1	1	1	1	0	1	0	0	1	1	1	1	1	0	0	**0**
15:	0	1	1	1	1	1	0	1	0	0	1	1	1	1	1	0	**0**
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	
$p_3(x)$	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	1	
16:	0	0	1	1	1	1	1	0	1	0	0	1	1	1	1	1	**0**

generates 15 LFSR shifts with P_2 and one shift with P_3 feedback polynomial. This is the only specific case where it is necessary to update the feedback polynomial two times in the same RN16 generation for $r = 0$ (different values of l or d would lead to more feedback polynomial updates). This situation is handled using the same value r.

Table 4.4: LFSR iteration example ($r = 1$)

	Seed v_1																
	S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	S_7	S_6	S_5	S_4	S_3	S_2	S_1	**Tx**
	0	0	1	1	1	1	1	0	1	0	0	1	1	1	1	1	
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	
$p_3(x)$	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	1	
17:	1	0	0	1	1	1	1	1	0	1	0	0	1	1	1	1	**1**
18:	1	1	0	0	1	1	1	1	1	0	1	0	0	1	1	1	**1**
19:	1	1	1	0	0	1	1	1	1	1	0	1	0	0	1	1	**1**
20:	1	1	1	1	0	0	1	1	1	1	1	0	1	0	0	1	**1**
21:	0	1	1	1	1	0	0	1	1	1	1	1	0	1	0	0	**1**
22:	1	0	1	1	1	1	0	0	1	1	1	1	1	0	1	0	**0**
23:	0	1	0	1	1	1	1	0	0	1	1	1	1	1	0	1	**0**
24:	0	0	1	0	1	1	1	1	0	0	1	1	1	1	1	0	**1**
25:	0	0	0	1	0	1	1	1	1	0	0	1	1	1	1	1	**0**
26:	0	0	0	0	1	0	1	1	1	1	0	0	1	1	1	1	**1**
27:	1	0	0	0	0	1	0	1	1	1	1	0	0	1	1	1	**1**
28:	1	1	0	0	0	0	1	0	1	1	1	1	0	0	1	1	**1**
29:	1	1	1	0	0	0	0	1	0	1	1	1	1	0	0	1	**1**
30:	0	1	1	1	0	0	0	0	1	0	1	1	1	1	0	0	**1**
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	
$p_4(x)$	1	0	1	1	1	1	1	0	0	0	1	0	0	0	0	1	
31:	1	0	1	1	1	0	0	0	0	1	0	1	1	1	1	0	**0**
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	
$p_5(x)$	0	0	1	0	1	1	0	0	0	1	1	0	0	0	0	1	
32:	1	1	0	1	1	1	0	0	0	0	1	0	1	1	1	1	**0**

The LFSR state after the 16_{th} shift will act as initial state (v_1) for the second PRNG update shown in Table 4.4. The trn value for this PRNG update is $r = 1$, hence, the Decoding Logic rotates the Polynomial Selector one position at shift 31, and another position at shift 32. Then P_3 is used 14 cycles, P_4 is used one cycle, and P_5 is used one cycle in this PRNG update and 14 cycles in the next PRNG update. The Polynomial Selector update is performed in two steps to be compliant with the hardware implementation where two clock cycles are necessary for this issue. The outputted bit at each LFSR cycle to conform the 16-bit pseudo-random sequence is the LSB of the LFSR.

4.4 Hardware Specification

In Sections 4.1 and 4.2 we have defined at a logical level the components and functionality of our proposed PRNG. In these sections, the components have been sized with a generic variable, e.g. the LFSR size n. In this section, we use the same parametrization used in Section 4.3, where we have detailed a PRNG execution example using specific values for the system's parameters and variables.

A numerical definition of the element's size of the PRNG is necessary to provide an implementable system, that is, specific hardware elements to build the proposed scheme. The goal of this section is to provide the approximated hardware complexity of our PRNG, sized in logical gates equivalence (GE), to measure its suitability for resource constrained devices in general, and low-cost RFID in particular. We introduce the parameter's definition of our PRNG proposal, and the technology used for the system integration in integrated circuits (ICs).

4.4.1 CMOS Technology

The ability to improve performance with reduced power consumption made the Complementary Metal-Oxide Semiconductor (CMOS) the dominant technology for integrated circuits [91]. The CMOS efficiency in IC scaling by reducing delay signaling (thus increasing the operating frequency), increasing transistor density and reducing the power consumption, converts this technology in the best candidate for low-cost RFID implementations. Specifically, Metal-Oxide Semiconductor Field-Effect Transistor (MOSFET) technology processes are commonly used for the RFID ICs fabrication.

CMOS technology integrates complementary PMOS and NMOS transistors to create digital logic circuits [9]. For a logical analysis of a CMOS digital circuit we can consider simple switches. If the input gate of a NMOS device is a logic 1 (V_{dd}), it is turned on. On the contrary, if it is a logic 0 (ground) the transistor is turned off. PMOS elements work in the inverse manner.

Figure 4.4 (a) shows a logic *inverter* made with CMOS transistors. The gate labeled with *M1* in the diagram represents a NMOS transistor, and the *M2* represents a PMOS transistor. If a logic 1 is applied to the input A, the *M2* transistor is turned off while the *M1* transistor is turned on, hence the output $\bar{A}$ is a logic 0 because it is connected to the *ground*. If the input A is a logic 0, then the *M2* transistor is turned on, and the *M1* transistor is turned off. Thus the V_{dd} signal generates a logic one in the output $\bar{A}$. The same logic applies to the NAND circuit shown in Figure 4.4 (b).

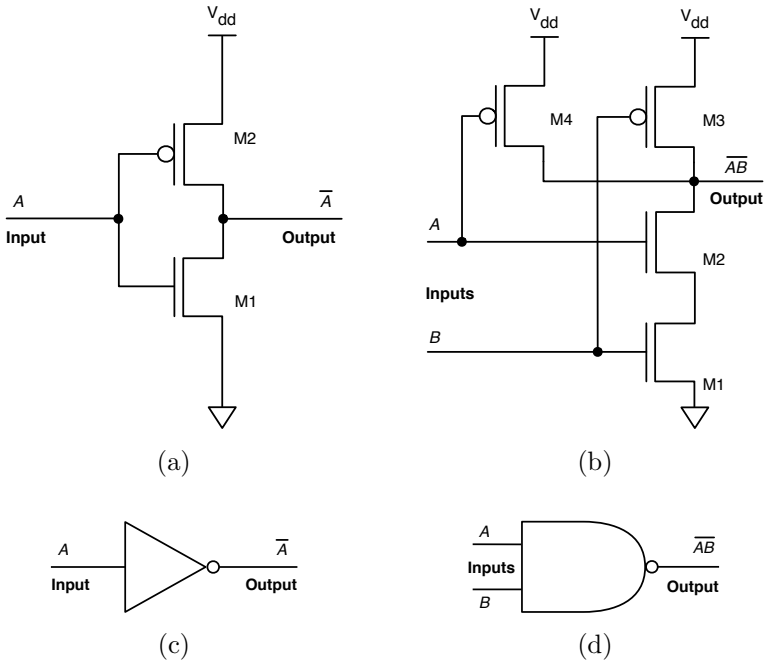


Figure 4.4: Diagram of a logic *inverter* with CMOS transistors (a) and its logic representation (c), and a *NAND* gate with CMOS transistors (b) and its logic representation (d)

Memory on Tags

Ciphers (thus PRNGs) have an internal state which we might refer to as cipher state and key state. The cipher state is fed by the initialization value and the key. Stream ciphers then, use the initialized cipher state to output the keystream. In non resource-constrained environments memory storage does not represent a problem, but in low-cost tag applications it is of main significance. Available memory is minimal for low-cost RFID technologies. Furthermore, read

and write access to the memory module, usually Electrically-Erasable Programmable Read-Only Memory (EEPROM), is highly power consuming. As a consequence, it is preferable to store all intermediate values and variables in registers rather than in external memory. For example, LFSRs typically consist of flip-flops (FFs). Nevertheless, non-volatile memories like EEPROM or CMOS Non-Volatile Memory (NVM) [66, 35] are necessary to store information that shall be available after the tag loses its power supply. Information such as the EPC unique identifier, or the passwords on the tag, are stored in the EEPROM memory [29].

We consider the storage of some register's state to avoid reinitializations with fixed values after tag power-off. Specifically, the LFSR needs 16 bits to store the $n = 16$ FFs of its register, and the Polynomial Selector state needs 3 bits to store which of the $m = 8$ feedback polynomials is being used at that moment. The Decoding Logic unit needs 4 bits to store the current LFSR cycle since the Polynomial Selector shall be updated at the $l = 15_{th}$ cycle. The mentioned variable values shall be stored at the end of the PRNG execution, to be loaded for the next pseudo-random sequence generation, totaling 23 bits of NVM to accomplish this requirement. This is not a strong hardware requirement since current EPC Gen2 ICs include relatively large memory banks like the Monza3 series with more than 500 bits of EEPROM space [43], or prototypes with 256 NVM like the *Zuma* proposal [66, 35], hence, current technology can accomplish this requirement. Albeit the additional 23 bits memory requirement may not suppose a technological problem regarding the area or the cost of fabrication, we do not consider the hardware implementation of non-volatile memory in this dissertation due to the difficulty to obtain hardware non-volatile memory models suitable for our design.

4.4.2 Logic Gates Equivalence

Area requirements for hardware implementation are usually measured in μm^2. This value depends, however, on the fabrication technology. For example HF RFID ICs are implemented with 0.35 μm CMOS processes. Current UHF low-cost RFID is implemented in 180 - 130 nm CMOS processes, which considerably reduces the implementation area for the same hardware [72].

In order to compare the area requirements independently, it is common to state the area as gate equivalents (GE). One GE refers to the area which is required by a two-input NAND gate. Gate equivalents (GE) are a measure that assesses the circuit complexity independently of the used CMOS technology [29]. Table 4.5 shows the gate estimation cost for basic combinatorial logic elements processed with CMOS technology, and the corresponding implementation area for a 130 nm process. A two-input NAND gate is evaluated as one gate equivalent, and an inverter (INV) is only 0.5 gate equivalent. This can also be seen in Figure 4.4 (a) where the inverter is made with two transistors, and Figure 4.4 (b) where the NAND is made with four transistors.

Although the number of gates per layout area and the fabrication cost vary depending on the process technology used, it is generally considered than an average cost per mm^2 is roughly four cents [89]. Hence the area restriction criteria in low-cost RFID is basically an economical issue rather than a technological problem.

4.4.3 EPC Gen2 Compliant Implementation

Although our proposed PRNG can be used as stream cipher in multiple applications, we look for compatibility with the EPC Gen2 [25]

requirements. As defined in Section 2.1 the EPC Gen2 requirements for pseudo-random number generation are limited to the compliance with three statistical properties regarding probability of appearance, simultaneity and predictability. Regardless of these three requirements, the EPC Gen2 Standard [25] states that the PRNG should provide 16-bit sequences, without any kind of hardware specification. On the other hand, the scientific literature quantifies in between 2,000 and 5,000 equivalent logic gates (GEs) the intended area for security operations in EPC Gen2 ICs [85].

As introduced in the Detailed PRNG Execution (cf. Section 4.3) the size and design of each PRNG component implies a specific hardware implementation. Therefore, we have defined a numerical value for the parameters, in order to be able to translate the high level design to a hardware implementation. The LFSR size n and the number of implemented feedback polynomials on tag m (cf. Table 4.1) are the key parameters in our design. Since the EPC Gen2 tags have a limited implementable area, the different elements to implement shall

Table 4.5: Cost estimation for combinatorial logic elements hardware based on static CMOS designs [72]

Logical Element	**Gate Equivalence**	**Area [μm^2] (for 130 nm process)**
INV	0.5	2.6
2 input NAND	1	5.2
2 input NOR	1	5.2
2 input AND	1.33	6.9
2 input OR	1.33	6.9
2 input XOR	2.67	13.9
2-1 MUX	2.67	13.9

be hardware efficient (regarding the implementation area). Since the EPC Gen2 standard requests to generate 16-bit random sequences, we fix the length of the LFSR n to 16 bits. This value offers EPC Gen2 tag compatibility and allows a better comparison with previous works.

The total number of different feedback polynomials is an important parameter regarding the security of the system. The higher the number of implemented feedback polynomials on tag, the higher the unpredictability of the PRNG, but also the higher the hardware area to implement. For the hardware implementation, the number of feedback polynomials on tag m is set to 8. This value gives an appropriate trade off between computational and system complexity. The different polynomials listed in Table 4.2 represent a possible selection of eight primitive feedback polynomials for our construction. All of them are primitive polynomials of degree 16 with the highest number of common elements. From 2,048 possible primitive polynomials, the selected ones have ten common elements and six different ones. With this special selection, only six bits are needed to connect them to the *Polynomial Selector*. Although an increase of the number of feedback polynomials leads to a higher number of different primitive polynomials, it also increases the amount and complexity of logical gates to implement the *Polynomial Selector*.

Linear Feedback Shift Register and Polynomial Selector

Linear Feedback Shift Register module and the Polynomial Selector module have been introduced separately in the Logical Components Section (cf. Section 4.2) due to its specific function inside the scheme. In the hardware design layer, a different approach shall be used. Although the feedback polynomial is an intrinsic part of the LFSR,

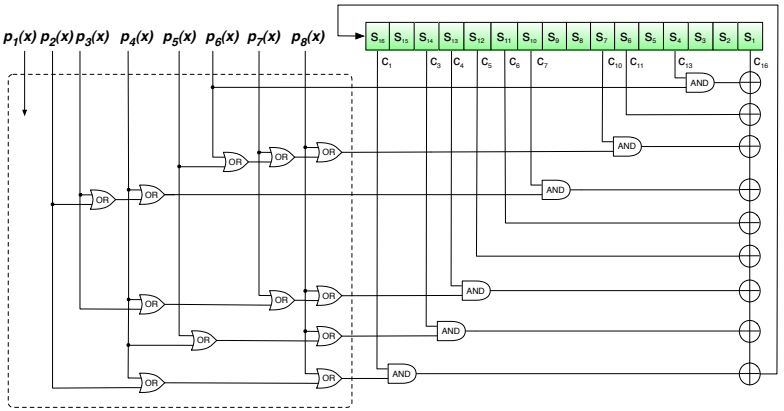


Figure 4.5: Multiple Polynomial selector activates one specific $p_i(x)$ generating the tap of the feedback polynomial. Representation with logical gates.

we consider its implementation in the Polynomial Selector module, due to the multiple-polynomial configuration. Figure 4.5 depicts the LFSR module plus the Polynomial Selector.

LFSRs are implemented in hardware with flip-flops and XOR logic elements. Flip-flops are controlled by the clock signal. The function of a flip-flop is to store a single bit for a time. Hence, flip-flops act like volatile memories of one bit. For the LFSR construction, 16 flip-flops are connected in series, transferring the bit stored in a flip-flop to the following flip-flop of the series at each clock time. For the LFSR implementation purpose we use the *D-flip-flop* (DFF) model specified at [9] and depicted in Figure 4.6 composed by 18 CMOS transistors. Hence, a D-FF can be measured with approximately 4.5 GE.

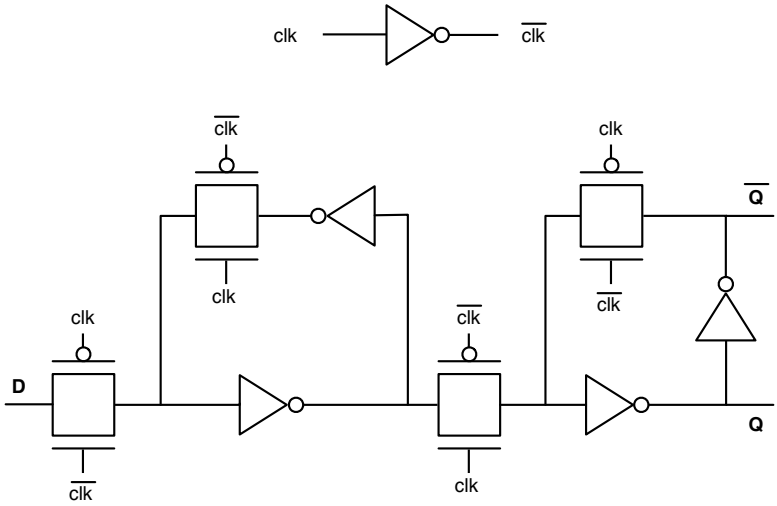


Figure 4.6: D-flip flop implementation with 4.5 GE [9]

The XOR logic elements implement the feedback polynomial function. The XOR elements are connected to the registers specified by the coefficients of the polynomial function (also known as taps). Taps' bits are bit-wise XORed in cascade to the input of the first flip-flop (MSB). The bit stored at the last flip-flop (LSB) is served as the output pseudo-random sequence, or discarded depending on the PRNG configuration.

The aim of the Polynomial Selector is to select a single feedback polynomial $p_i(x)$ from the m implemented on board, for the pseudo-random sequences generation. The Polynomial Selector chooses a single $p_i(x)$ regarding the trn input bit r and the current selected polynomial. Once the new $p_i(x)$ is chosen, the Polynomial Selector needs to transform that selection to feedback taps for XORing the LFSR state. To achieve this following the low-hardware complexity

requirement we need to design a simple circuit, where the LFSR taps would dynamically change by means of logical combinations.

Table 4.6 shows the coefficients' representation of the eight primitive polynomials of degree $n = 16$ previously presented in Table 4.2, that should be implemented on the tag. The coefficients configure the polynomial expression $(c_0 + c_1x + c_2x^2 + \ldots + c_{16}x^{16})$ which determines the feedback bits. The last row represents the elements of the polynomials to be implemented in hardware. The columns with eight ones (denoted as *fixed column* hereinafter) mean that all the eight polynomials feedback these taps from the register, except for the c_0 coefficient which is not represented by any component (it represents the mathematic concept of feedback). The columns with a number in the last row ($1 \leq k < 8$, and denoted as *variable columns* hereinafter) mean the number of polynomials that feedback these taps

Table 4.6: We look for primitive polynomials with the maximum number of common elements to simplify the hardware implementation

	Primitive Polynomials $p_i(x)$															
$C(x)$	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}
$p_1(x)$	1	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1
$p_2(x)$	1	1	0	0	0	1	1	1	0	0	0	1	0	0	0	0	1
$p_3(x)$	1	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	1
$p_4(x)$	1	1	0	1	1	1	1	1	0	0	0	1	0	0	0	0	1
$p_5(x)$	1	0	0	1	0	1	1	0	0	0	1	1	0	0	0	0	1
$p_6(x)$	1	0	0	0	0	1	1	0	0	0	1	1	0	1	0	0	1
$p_7(x)$	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0	0	1
$p_8(x)$	1	1	0	1	1	1	1	0	0	0	1	1	0	0	0	0	1
k		3		3	4	8	8	3			4	8		1			8

from the register. The empty positions in the last row means there is no element on that column for any polynomial.

The last row information from Table 4.6 is useful to translate the polynomials matrix to implementable hardware elements. The taps corresponding to *fixed columns* are directly linked to XOR gates, because all the polynomials feedback these taps. For *variable columns* the tap is feedback depending on the specific polynomial, thus after the XOR gate some logic is necessary to activate or deactivate the tap depending on the selected $p_i(x)$. A network of logical OR and AND gates solves the *variable columns* tap selection.

Figure 4.5 represents with (two input) logical gates the polynomial selection mechanism based on the example of Table 4.6. Fixed columns c_{16}, c_{11}, c_6 and c_5 are directly linked to XOR gates. Variable columns c_{13}, c_{10}, c_7, c_4, c_3 and c_1 must only feedback the tap content for specific polynomials, thus an OR network is added (dashed frame in Figure 4.5) to select the specific set of taps for each polynomial $p_i(x)$. In Figure 4.5, $p_1(x)$ is deliberately grounded because only fixed taps are feedback. Furthermore an AND gate and an XOR gate are also added for each variable column. Table 4.7 resumes the methodology to count the necessary two input logic gates to implement the multiple polynomial selector depending on the polynomials matrix.

This characteristic implies different hardware implementations depending on the specific feedback polynomials implemented on tag. From the EPC Gen2 parameters definition we have fixed the LFSR length (thus also the feedback polynomials) to 16 bits. For a 16-bit polynomial there are 2,048 primitive polynomials (cf. Equation 2.6). Our PRNG combines $m = 8$ of this 2,048 polynomials to give the feedback bits to the register, thus, there are $\binom{2,048}{8}$ possible combi-

nations (binomial coefficient) for primitive polynomials of 16_{th} order for the PRNG, that is almost 2^{73} possible combinations.

A first overview on Table 4.7 formulas allows us to calculate the most expensive case for the hardware implementation of the feedback polynomials, that happens for 15 variable columns with 7 elements each (assuming that this case is an approximation), resulting in a logical gate estimation of 180 GE (cf. Table 4.5). This hardware complexity is considerably low compared with the 2,000 to 5,000 GE estimated in the literature for security operations [85], thus suitable for our PRNG scheme. However, regarding the low-cost resource constrained scenario, the polynomial selection ($\mathcal{P}_{\text{sel}}$) can be optimized to save hardware complexity in the implementation.

Since it is computationally unfeasible to compute the hardware cost of the $\binom{2,048}{8}$ combinations to check its suitability to the hardware design, we can select a set t of feedback polynomials with appropriate hardware parameters. That is, none of the possible combinations of eight polynomials over the set t may suppose an excessive hardware implementation area. From the 16_{th} order 2,048 primitive polynomials we select sets with more than 8 polynomials. Since there are only 7 polynomials with 12 common coefficients, we use sets with 11 com-

Table 4.7: Hardware implementation of Multiple Polynomial selector

Multiple Polynomial selector elements	
XORs	(Fixed columns) + (Variable columns) - 1
ANDs	(Variable columns)
ORs	Sum of (elements - 1) of each Variable column

mon coefficients at much. If, for example, the set t is composed by polynomials with 11 common coefficients, we can combine groups of 8 polynomials from a set of $t = 10$, which means $\binom{2,048}{8} = 45 \simeq 2^{5.5}$ possible combinations with a maximum hardware complexity of 54 GE. Table 4.8 resumes the hardware complexity and the combination possibilities of the sets of polynomials with 7 to 11 common coefficients, plus the approximation of the set including all primitive polynomials of 16_{th} order.

The smaller the set t, the higher the number of possible polynomial combinations to implement in the PRNG scheme. Choosing a specific set of polynomials is a trade-off between hardware efficiency and security. The security improves with the number of implemented polynomials, but it also increases the necessary GEs. If no specific hardware restrictions are applied, all primitive polynomials can be considered. The polynomials shown in Table 4.2 and Table 4.6 are an example of the subsets analyzed in this section.

Moreover, a logical circuitry is necessary to conduct the Decoding Logic selection signal to the selected feedback polynomial. Since we implement eight polynomials on tag ($m = 8$), we need an adder modulo eight (acting as a rotation *wheel*) to select the feedback polynomials (cf. Section 4.2). We implement the rotation *wheel* with a binary adder made of three DFFs, which generates eight different combinations with three bits (2^3). A 3-bit decoder (made of three inverters and 16 two-input AND gates) is also implemented to derive the selection signal to the specific feedback polynomial. Furthermore, the feedback polynomial net is also included in the Polynomial Selector module. It is composed by 12 (two inputs) OR gates which define the specific taps for each cycle, plus the 6 AND gates and 9 XOR gates which compute the feedback bit.

Decoding Logic

The Decoding Logic is the responsible of managing the internal PRNG clock, and activate and deactivate the PRNG modules for its proper performance. To achieve this, the Decoding Logic manages 64 clock cycles from the IC clock signal (f_{clk}), to perform the operations requiring a specific timing. From the 64 clock cycles the Decoding Logic generates specific signals for reading and writing to the memory, shifting the LFSR and optionally discarding bits. To implement these actions in hardware, we use a binary adder composed by six FFs, and 13 logic gates (ANDs, ORs, XORs and inverters).

The Decoding Logic also manages the specific clock cycle to select a new feedback polynomial. Unlike the previously described 64 cycles which are reinitialized for each PRNG update, we should keep the

Table 4.8: Polynomial combination parameters

Common Coefficients	**t**	**Combinations**	**Max. GE**
$p_i(x) : 10000000vv01v11v$ (12)	7	$\binom{7}{8} = 0$	-
$p_i(x) : 10v00000vv01v11v$ (11)	10	$\binom{10}{8} \simeq 2^{5.5}$	54
$p_i(x) : 100v01v00v11vv0v$ (10)	16	$\binom{16}{8} \simeq 2^{13.7}$	70
$p_i(x) : 11vv00v0v00v10vv$ (9)	24	$\binom{24}{8} \simeq 2^{19.5}$	80
$p_i(x) : 11vv0vv0v00v10vv$ (8)	36	$\binom{36}{8} \simeq 2^{25}$	86
$p_i(x) : 10vv1v00vv1vv1vv$ (7)	57	$\binom{57}{8} \simeq 2^{31}$	107
$p_i(x) : 1vvvvvvvvvvvvvvv$ (1)	2,048	$\binom{2,048}{8} \simeq 2^{73}$	~ 180

$l = 15$ bit shifts to sequentially rotate the polynomial selection cycle. This operation is achieved with an independent 15-bit counter implemented with 4 FFs and 10 logical gates which is reinitialized at the 15_{th} cycle.

Finally, de Decoding Logic is also in charge of deriving the *trn* bit (from the TRNG module) to the Polynomial Selector. To achieve this, the Decoding Logic processes the *trn* bit sending one or two clock cycles to the Polynomial Selector binary adder depending on the value r of the TRNG bit. On the one hand, if the *trn* value is $r = 0$, then the Decoding Logic module selects a new feedback polynomial $p_i(x)$ and performs 15 LFSR clock shifts. On the other hand, if $r = 1$, the Decoding Logic module shall jump one position of the polynomials *wheel* and select $p_{i+1}(x)$. This is achieved by selecting one cycle $p_i(x)$, and $p_{i+1}(x)$ for the following 14 clock cycles.

Thermal-noise *trn*

The oscillator-based high-frequency sampler proposed in the Che *et al.* design [18] combines analogical and digital circuitry. A low-frequency oscillator detector is implemented in the analogical circuitry which basically amplifies the voltage difference between two resistors due to the thermal-noise, generating a low-frequency triangular wave signal. The analogical circuitry also receives the high-frequency oscillation from the radio frequency (RF) front-end. On the digital circuitry side, a flip-flop samples the high-frequency signal using the low-frequency thermal-noise signal as a clock reference. The output is a truly random binary sequence. The same proposal has been also modeled by Zhou *et al.* in [101].

4.4.4 Hardware Specification Summary

Hardware complexity in CMOS circuits is directly related to the implemented area, and the execution time of the scheme. Pseudo-random number generators for the EPC Gen2 technology are expected to be implemented with a small amount of equivalent logic gates (GE), defined in the literature between 2,000 and 5,000 [85]. The definition of the different PRNG modules (cf. Section 4.2) allows us to narrow the hardware complexity of our approach using the CMOS technology specified in Table 4.5. We provide in Table 4.9 the GE counting of the three main modules: the LFSR module, the Polynomial Selector module and the Decoding Logic module. These three elements add up to the most representative amount of GEs. They consist of the 16-bit LFSR core and the logic that handles its linearity problems. For the Polynomial Selector module we count the worst case (a combination of feedback polynomials using 180 GE). We then provide the physical source of randomness assumed for our generator (denoted as TRNG in our simulation). For the gate equivalence of this component, we based our estimations on previous works presented in [85, 72]. In this sense, the physical source of randomness that we assume consists of the thermal-noise oscillator presented by Che *et al.* in [18], but specified and modeled in our work as proposed in [76] and [101]. The main elements of the TRNG component are, therefore, a low-frequency oscillator and a Flip-Flop which contains the resulting *trn* output bit.

The remainder GEs mainly consist of the necessary extra circuitry for controlling the different states of the generator. The final amount of GEs is of, at most, about 452 logic gates. This value perfectly matches the EPC Gen2 requirements. Furthermore, it is considered that a complexity of 1,000 GEs can be added to a RFID tag with-

Table 4.9: Logical GE count for our proposed PRNG

Element	**Function**	**GE count**
	LFSR	**73.3**
16 Flip-Flops	16-bit Register	72
1 AND	PRNG output	1.3
	Polynomial Selector	**213.6**
15 ANDs + 15 XORs + 90 ORs	LFSR Feedback	180
14 ANDs + 3 INVs + 3 Flip-Flop	Decoder	33.6
	Decoding Logic	**68**
6 Flip-Flops	64 Cycle Clock	27
5 ANDs + 2 ORs + 2 INVs + 1 XOR	LFSR & Memory Clocking	13
4 Flip-Flop + 4 ANDs + 1 OR + 1 INV	Feedback Selector Cycle	28
	Thermal noise-based $TRNG$	**22**
	Additional control	**76**
	Total	**452**

out perceptible additional cost [29]. At the same time, and regarding other lightweight cryptographic proposals for stream ciphers or PRNGs such as Trivium [22], LAMED [75], or Grain [39], we can conclude that our proposal has a considerably lower hardware complexity (cf. Table 4.10). It is worth mention that this hardware complexity evaluation does not include the 23-bit non-volatile memory hardware (to store the registers state during tag power-off periods) which have been described in the PRNG Logical Description (cf. Section 4.1).

PRNG Execution Time

The EPC Gen2 standard has important temporal requirements, which demand that a given number of tags should be read in a given amount of time [25]. Gen2 readers should be able to read 450 tags/sec. This puts a severe limitation on the maximum number of cycles ($\frac{1}{f_{clk}}$) a tag can spend to generate a RN16 [75]. Therefore, most existing works specify the maximum time consumption of an EPC Gen2 PRNG in terms of the data transmission rate. For example, Ranasinghe and Cole determine in [85] the maximum execution time of an EPC Gen2 PRNG to be bounded by 5 to 10 ms, in order to address a minimum tag reading speed of, at least, 200 labels per second. Peris *et al.* discuss in [75] the need of settling the clock frequency signal f_{clk} of an EPC Gen2 tag up to 100 kHz.

Table 4.10: GE comparison of lightweight PRNG proposals

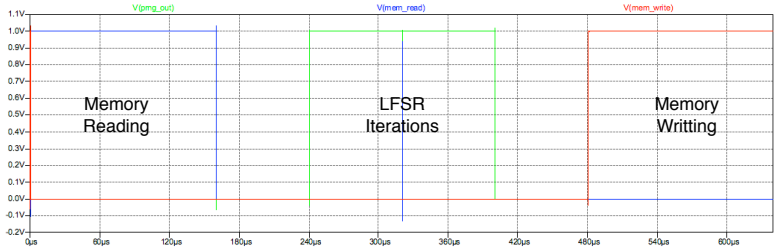
	Trivium	**LAMED**	**Grain**	**Our Proposal**
GE Count	1,857	1,585	1,294	452

At this frequency, the number of clock cycles that are allowed for the generation of a 16-bit sequence from our generator amounts to 220 cycles. This requirement is, indeed, easily achieved by our generator. Notice that the LFSR takes exactly one clock cycle to output each bit. Hence, 16 cycles are needed to generate the whole 16-bit sequence. The proposed hardware implementation (cf. Section 4.4) of our proposal takes 64 clock cycles to complete the full PRNG generation. Depicted in Figure 4.7 (a), the PRNG generation divided in three blocks; memory reading to obtain the previous RN16 (initial state of the LFSR) and the polynomial cycle, LFSR update and PRNG transmission, and memory writing. The proposed implementation takes 16 LFSR cycles to generate the PRNG, but 16 additional cycles are available if the bit discarding option is enabled (cf. Figure 4.7 (b)). Figure 4.8 depicts a full pseudo-random sequence transmission together with the feedback polynomial selection cycle.

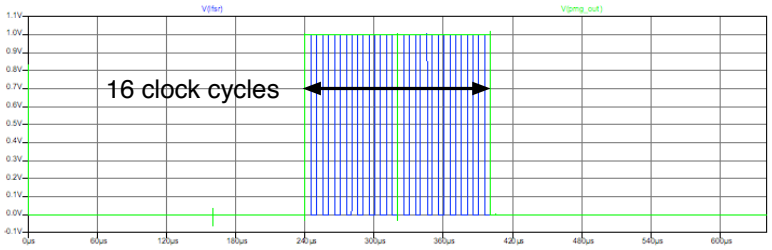
Some tasks of the PRNG like memory reading or writing or bit discarding, can be executed in parallel with the rest of the IC operations since only the 16 cycles of the RN16 transmission are relevant for the set of IC operations. One of the parallelized operations is the *trn* acquisition which as it is specified in [18] (and as it has been simulated for our generator), can decrease its sampling frequency under the 2.2 kHz to provide a different *trn* bit for each PRNG execution and decrease the power consumption which is closely related to the operating frequency [9].

4.5 Chapter Summary

Designing suitable pseudo-random number generators (PRNGs) for security applications its a challenging task. A new pseudo-random



(a)



(b)

Figure 4.7: PRNG time slots scheme. In (a) the 64 clock cycles are divided in reading and writing to the memory, and PRNG generation. In (b) the LFSR shifting clock, and PRNG transmission are detailed.

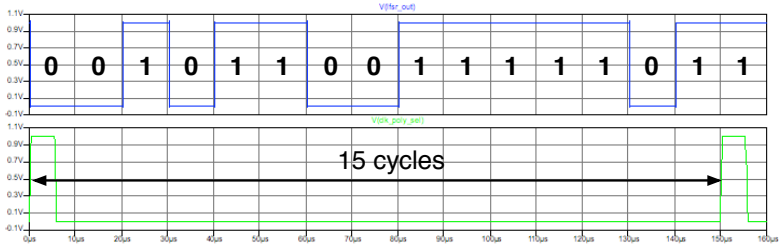


Figure 4.8: Pseudo-random sequence transmission example with the feedback polynomial selection cycle

number generator design for EPC Gen2 devices has been presented. This generator is based on a 16-bit linear feedback shift register (LFSR) designed as a multiple-polynomial LFSR architecture. This leads to having different feedback primitive polynomials fed by a physical source of randomness for handling the inherent linearity of the LFSR module.

Aside from the logical description of the proposed PRNG, a hardware implementation based on CMOS technology is also provided (except for the non-volatile memory hardware).

Our main contributions include:

- The definition of a new PRNG based on a LFSR for security applications, compliant with the EPC Gen2 technology.
- The utilization of a multiple-polynomial architecture to break the inherent linearity of LFSRs.
- An evaluation of primitive polynomials of degree 16 with suitable characteristics for its hardware implementation with up to 2^{72} possible combinations
- The proposed PRNG has 452 GE being smaller than other lightweight proposals in the literature.

5

Evaluation of our PRNG Proposal

A new pseudo-random number generator (PRNG) design for Electronic Product Code Class 1 Generation 2 (EPC Gen2) tags has been proposed in Chapter 4. The proposed PRNG successfully handles the inherent linearity of linear feedback shift register (LFSR) based PRNGs.

After a high level logical description of the method used to obtain the randomness on the tag (cf. Section 4.1), and a preliminary hardware specification implementing the designed technique (cf. Section 4.4), an evaluation is needed to confirm the suitability of our design to the specifications required for the EPC Gen2 standard for radio-frequency identification (RFID). The following sections provide a description of its statistical properties (cf. Section 5.1), a formal

security analysis (cf. Section 5.2), and a detailed power consumption analysis (cf. Section 5.3) according to the current technology implementing EPC tags.

5.1 Statistical Properties

Since the main property of a PRNG is to ensure the forward unpredictability of its generated sequence, the correctness of a PRNG can be measured with statistical tests applied to the output sequence. Similarly to the statistical tests conducted in Section 3.1 and Section 3.3 to analyze the Che *et al.* PRNG proposal and the PRNGs from commercial EPC Gen2 labels, we base the statistical analysis of our PRNG proposal on the NIST Test Suite [4] for checking the possible randomness deviations of the binary pseudo-random sequences generated from our PRNG proposal.

NIST Test Suite performance is detailed in Section 2.4. Briefly stated, NIST tests produce *P-values* summarizing the strength of the randomness hypothesis. If *P-values* results are over the level of significance, the analyzed sequences are likely to be random from statistical point of view. The proportion of tests over the significance level, must fit in the interval specified by Equation 2.12. To statistically confirm the randomness of the analyzed data, one would expect one in 100 sequences to be rejected (that is, the significance level), being a common value in cryptography [4]. *P-values* passing the significance level give a confidence of 99.9% of the randomness of the evaluated sequence (if 100 sequences are evaluated, results should pass 0.9615 as defined in Equation 2.12).

To analyze our PRNG proposal we generate using a software simulation ten different pseudo-random sequences, denoted hereinafter as

Table 5.1: Results obtained from the execution of the NIST statistical test suite

Runs	100									
Sequence	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
Frequency	1.00	0.97	1.00	0.99	0.99	0.99	0.99	1.00	0.98	1.00
BlockFrequency	1.00	1.00	1.00	0.99	0.99	0.99	1.00	1.00	0.99	0.99
Runs	0.99	1.00	0.97	0.99	0.99	1.00	0.99	0.99	0.99	1.00
LongestRun	0.98	0.98	0.97	0.99	0.99	0.99	0.97	0.97	0.97	0.98
Rank	1.00	0.99	0.99	0.98	0.98	0.99	0.98	0.99	0.99	1.00
OverlappingTemplate	0.97	1.00	0.99	1.00	1.00	0.97	0.99	0.99	1.00	0.99
Universal	0.97	0.99	1.00	0.97	0.97	0.99	0.99	0.99	0.99	0.99
ApproximateEntropy	0.97	1.00	1.00	0.99	0.99	1.00	1.00	0.98	1.00	0.99
LinearComplexity	0.99	0.99	1.00	0.99	0.99	1.00	0.99	0.98	0.99	0.99
CumulativeSums	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$
NonPeriodicTemplate	$\mathbf{\frac{146}{148}}$	$\mathbf{\frac{147}{148}}$	$\frac{148}{148}$	$\mathbf{\frac{147}{148}}$	$\mathbf{\frac{147}{148}}$	$\mathbf{\frac{147}{148}}$	$\frac{148}{148}$	$\frac{148}{148}$	$\mathbf{\frac{147}{148}}$	$\frac{148}{148}$
RandomExcursions	$\mathbf{\frac{7}{8}}$	$\frac{8}{8}$	$\frac{8}{8}$	$\mathbf{\frac{6}{8}}$	$\mathbf{\frac{7}{8}}$	$\frac{8}{8}$	$\frac{8}{8}$	$\mathbf{\frac{7}{8}}$	$\frac{8}{8}$	$\frac{8}{8}$
RandomExcursionsVariant	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$	$\frac{18}{18}$
Serial	$\frac{2}{2}$	$\frac{2}{2}$	$\mathbf{\frac{1}{2}}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\frac{2}{2}$	$\mathbf{\frac{1}{2}}$	$\frac{2}{2}$

$T_{i,(1\leq i\leq 10)}$. These sequences are generated with different *seeds*, and fed by different *trn* bits obtained from the *Random.org* service [37]. Each set of sequences T_i amounts to having a total of 10 million 16-bit pseudo-random sequences (RN16), representing about 200 MB of generated pseudo-random data for the whole set of tests. The results obtained from the execution of the NIST statistical test suite once processed each T_i are summarized in Table 5.1.

Notice that the overall results for the tested sequences is highly satisfactory. Only the *Serial*, *NonPeriodicTemplate* and *RandomExcursions* tests partially fails for some analyzed sequences (marked in bold text in the table). As detailed in the NIST documentation [4], if some results in the same test do not achieve the acceptable interval, more tests should be performed in order to solve the uncertainty. For our proposal, the 90% of *Serial* tests, 92.5% of *RandomExcursionsVariant* tests and the 99.5% of *NonPeriodicTemplate* tests are satisfactory. This results significantly improve the statistical tests performed on the Che *et al.* pseudo-random sequences (cf. Section 3.1) and the EPC Gen2 commercial analyzed PRNGs (cf. Section 3.3) For a description of the tests mentioned in this section the reader can review Section 2.4.

5.1.1 Suitability to the Randomness Requirements of EPC Gen2 Standard

In addition to the NIST statistical test suite, we conduct a second series of tests to confirm the suitability of our proposal for handling the statistical and randomness requirements defined by the EPC Gen2 specification [25]. Detailed in Section 2.1, these requirements can be summarized as follows:

1. The probability that any single 16-bit sequence j drawn from the generator shall be bounded by $\mathrm{P}_{min} = \frac{0.8}{2^{16}} < \mathrm{Prob}(j) < \mathrm{P}_{min} = \frac{1.25}{2^{16}}$.

2. Among a tag population of up to ten thousand tags, the probability that any two tags simultaneously generate the same 16-bit sequence shall be less than 0.1%.

3. The chance of guessing the next 16-bit sequence generated by a tag shall be less than 0.025% even if all previous outputs are known to an adversary.

To confirm the achievement of the first requirement, we analyze the frequency of occurrence of each sequence generated from our generator. To do so, we base the Frequency Test on the ten T_i series of sequences already elaborated for the *NIST* statistical test suite, and the results are shown in Figure 5.1. The results confirm that, after analyzing 30 million RN16 sequences, the probability of occurrence of any given value lies between $P_{min} = \frac{0,8}{2^{16}}$ and $P_{max} = \frac{1,2}{2^{16}}$. Therefore, our proposed generator is compatible with the EPC Gen2 Standard for pseudo-random number generation.

Figure 5.2 depicts the evolution of P_{max} and P_{min} values of our PRNG proposal for ten million analyzed RN16s (green line) compared with the *Random.org* reference data, and the EPC Gen2 analyzed commercial PRNGs. It is worth to mention that none of the analyzed sequences (neither the truly generated sequences from *Random.org*) are inside the probability boundaries specified by the EPC Gen2 standard, because more sequences are necessary to reach the required values, as detailed in Section 3.3. Furthermore we depict the frequency analysis of our proposed PRNG in Figure 5.3, and the frequency distribution in Figure 5.4.

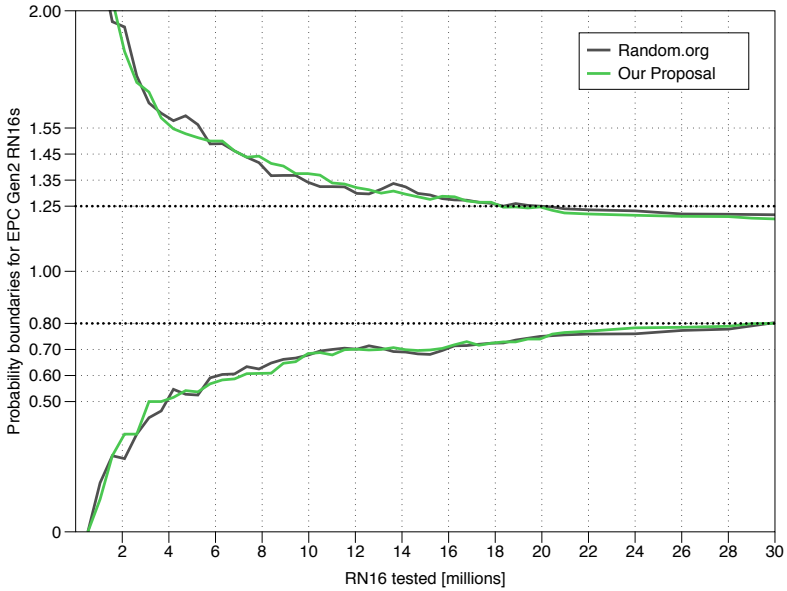


Figure 5.1: Our PRNG proposal is compatible with the 1_{st} EPC Gen2 requirement

The second property for building EPC Gen2 compliant PRNGs requires that two simultaneous identical sequences must not appear with more that 0.1% for a population up to ten thousand tags. To test this property, ten thousand instances of our generator, initialized at random, are used to simulate a real population of 10,000 tags. The obtained results, shown in Table 5.2, verify that, after ten tests of 1,000 iterations each, none of them show a simultaneous identical sequence rate higher than 0.03793%. We can, therefore, confirm that our proposed generator also meets the second requirement of the EPC Gen2 specification.

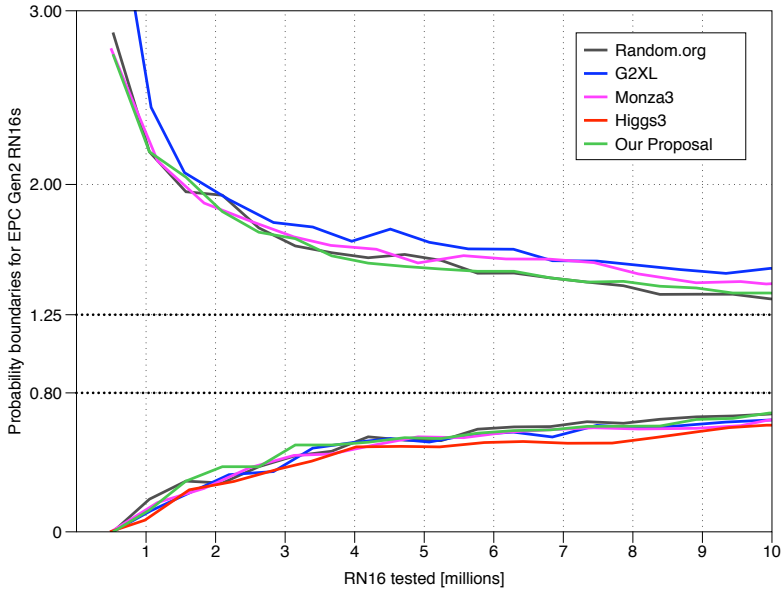


Figure 5.2: Our PRNG proposal shows an improvement of the P_{max} and P_{min} parameters (1_{st} EPC Gen2 requirement), compared with the commercial analyzed PRNGs (cf. Section 3.3)

Finally, to statistically confirm the fulfillment of the third property, we conducted a series of correlation tests based on the T_i sequences. Each test computes the degree of dependence of the ongoing bits regarding their predecessors. Specified in the Standard [4] the pseudo-random sequences cannot be guessed with a chance higher than 0.025%. We can see by looking the results shown in Table 5.3, that all the tests are under the requested values. This confirms the low linearity of our generator. It also proves the fulfillment of the third property, since we can confirm that the generated sequences are not predictable within the requested boundary.

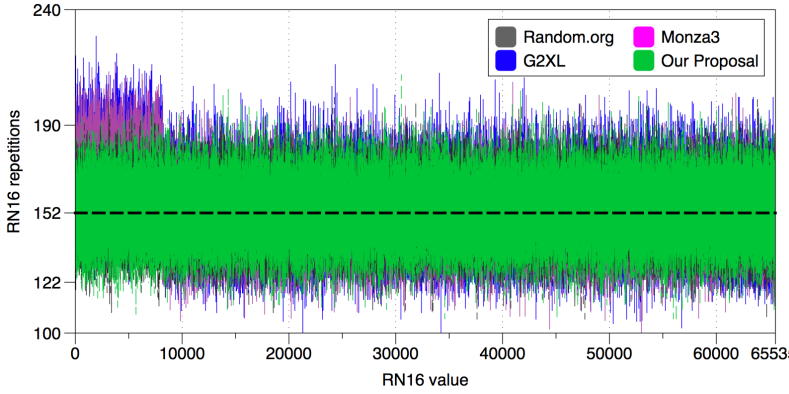


Figure 5.3: The RN16s frequency analysis for our PRNG proposal shows a similar behavior with the *Random.org* reference data.

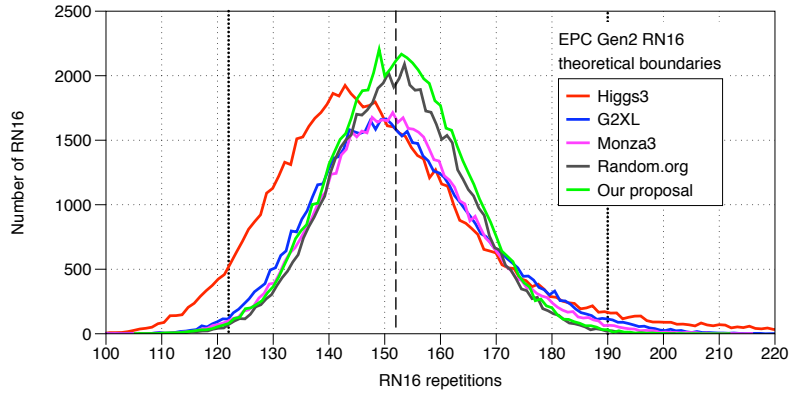


Figure 5.4: The RN16 frequency peak is centered at the theoretical mean for 10 million analyzed pseudo-random sequences

The results provided in this section show a correct statistical performance for a cryptography-purpose PRNG. Although these results augur a good performance for our proposed PRNG, the analysis has been done regardless of the knowledge of the internal PRNG design, that is, without considering the deterministic algorithm which generates the RN16 sequences. The following section deepens on a security analysis based on the internal PRNG structure, in order to evaluate the likelihood of an attack success performed by an adversary knowing the PRNG design.

Table 5.2: EPC Gen2 second randomness property test for our PRNG proposal

Test (% rate)	1_{st}	2_{nd}	3_{rd}	4_{th}	5_{th}
Simultaneous RN16	0.03777	0.03784	0.03793	0.03772	0.03772

Test	6_{th}	7_{th}	8_{th}	9_{th}	10_{th}
Simultaneous RN16	0.03768	0.03757	0.03765	0.03783	0.03749

Table 5.3: EPC Gen2 third randomness property test for our PRNG proposal

Sequence	T_1	T_2	T_3	T_4	T_5
Correlation	0.000001	0.000057	-0.000088	-0.000073	-0.000073

Sequence	T_6	T_7	T_8	T_9	T_{10}
Correlation	-0.000038	0.000134	0.000082	-0.000097	-0.000001

5.2 Security Analysis

To demonstrate the suitability of our PRNG proposal for the RFID tags, a certain level of security requirements shall be provided. The on-board PRNG is the main security tool in the EPC Gen2 tag technology. It is used to avoid identification collisions in multi-tag environments and to encrypt the restricted information in the reader-to-tag communication (cf. Section 2.1).

The final goal to design stream ciphers is based on the randomized approach, where the aim is to assure security by ensuring that breaking the cipher requires an adversary to perform an impractical amount of work [83]. The analysis of EPC Gen2 commercial labels being currently used worldwide, reveal differences in the behavior of its PRNG, leading to significant deviations of its generated data (cf. Section 3.3). Furthermore, security through obscurity (such as current EPC Gen2 systems) has been demonstrated ineffective also in RFID environments [30, 68]. Finally, cryptographic proposals missing an appropriate security analysis have the risk of suffering attacks from adversaries, revealing weaknesses in their system, as we have demonstrated with the attack to the Che *et al.* PRNG (cf. Section 3.2).

The remainder of this section includes a description of the parameters implied in the security characteristics of our PRNG proposal. Attacks against the robustness of our proposal are also presented.

5.2.1 Description of Parameters

In the previous section, we have analyzed the statistical properties of the 200 MB pseudo-random sequences generated by our proposed

PRNG, evaluating the system's randomness like a black-box random data generator. Regarding the security analysis of our system, we should consider the parameters determining the behavior of the PRNG underlying deterministic algorithm. In this new scenario, we consider an adversary knowing the details of the algorithm which enables the PRNG performance.

As defined in the Logical Components description (cf. Section 4.2), the main components of our PRNG are a LFSR, a Feedback Polynomial Selector circuit and a Decoding Logic unit, plus the TRNG module and the Memory Blocks. Each of the above units perform specific actions for the correct generation of pseudo-random bits. Nevertheless, the design rules do not only affect the system implementation, but also the security itself. Since our design is based on a deterministic algorithm (perturbed by a source of true randomness), the size of the components and their design are crucial for the statistical behavior and, therefore, the security of the whole system. Considering that our PRNG proposal is designed to act as a cipher tool on a RFID tag, its aim is to provide a secure enough communication link. That is, a good statistical behavior to avoid unauthorized readers to obtain the encrypted data. Table 4.1 resumes the parameters to consider from the implementation point of view. Moreover, each of the mentioned units have been sized to set up the hardware implementation (cf. Section 4.4) based on similar designs in the literature [60], and following a trade-off between the basic factors for RFID design: cost, power consumption and size (cf. Section 2.1).

Table 5.4 resumes the basic parameters which must be considered from the security point of view. An adversary with the knowledge of the PRNG algorithm can work with the following assumptions: a 16-bit LFSR, 8 different feedback polynomials implemented onboard, no bits are discarded from the LFSR output, and the feedback

polynomial selection is implemented each 15 LFSR shifts. In this section, we also detail the effect of decreasing the polynomial update cycle ($l \leq 15$) and discarding bits of the LFSR output ($d \geq 0$). On the contrary, the adversary does not know the initial state of the LFSR (v_0), the value of the 8 feedback polynomials ($\mathcal{P}_{\text{sel}}$), and the true random bit value r.

The aim of the adversary is, trying to solve the uncertainty of the random bit r, to predict the PRNG output sequences, that is, to solve the initial state and the feedback polynomials implemented on board. Thus, the *trn* bit is the key of the randomness of our PRNG proposal. We can make use of *trn* functionality because of the EPC Gen2 communication model, where the tag sends the keystream in plaintext to the reader, due to the lower signal strength of the tag-to-reader channel (cf. Figure 4.1).

Table 5.4: Definition of parameters for security analysis

Public parameters	
Size of LFSR (bits)	$n = 16$
Number of feedback polynomials on tag	$m = 8$
Implementable polynomials	$\mathcal{P}$
Polynomial Selector update period	$l \leq 15$
Number of LFSR discarded bits	$d \geq 0$
Secret parameters (adversary goals)	
LFSR initial state (seed)	v_0
True random bit (*trn*)	r
Feedback polynomials in tag	$\mathcal{P}_{\text{sel}} \in \mathcal{P}$

Attack Definitions

We denote the set of all analyzed RN16s as bit strings $\mathcal{S} = s_1, s_2, s_3 \ldots$, where each single RN16s is defined as $\mathcal{S}_{1,16} = \{s_1 \ldots s_{16}\}$, $\mathcal{S}_{17,32} = \{s_{17} \ldots s_{32}\}$, etc ..., and the coefficients of a feedback polynomial $p_i(x)$ are defined as $c_1, c_2, \ldots, c_{16}$. The dataset $\mathcal{S}$ is generated through different feedback polynomials ($\mathcal{P}_{\text{sel}} = \{p_i(x)\}$ for $1 \leq i \leq 8$). The selected polynomials on tag are all primitive ($p_i(x) \in \mathcal{P}$). Non primitive polynomials are denoted hereinafter as $q(x) \notin \mathcal{P}$. As described in Section 2.2, to obtain $p_i(x)$ of a 16-bit LFSR, the attack needs at least $2n = 32$ outputted bits generated with the polynomial ($\mathcal{S}_{i+16,i+31} = \text{LFSR}_{p_i(x)}(\mathcal{S}_{i,i+15})$). Equation 5.1, denoted hereinafter as Ω, details the 32 coefficients from $\mathcal{S}$ to solve $p_i(x) = \Omega(\mathcal{S}_{1,16}||\mathcal{S}_{17,32})$.

$$\begin{bmatrix} s_1 & s_2 & \cdots & s_{16} \\ s_2 & s_3 & \cdots & s_{17} \\ \vdots & \vdots & \ddots & \vdots \\ s_{15} & s_{16} & \cdots & s_{30} \\ s_{16} & s_{17} & \cdots & s_{31} \end{bmatrix} \begin{bmatrix} c_{16} \\ c_{15} \\ \vdots \\ c_2 \\ c_1 \end{bmatrix} = \begin{bmatrix} s_{17} \\ s_{18} \\ \vdots \\ s_{31} \\ s_{32} \end{bmatrix} \tag{5.1}$$

The adversary faces different drawbacks while performing the attack. On one hand, (assuming $l \leq 15$) 15 bits or less are generated from each 16_{th} order feedback polynomial, hence, even capturing all outputted bits the adversary is not able to fill all the necessary unknown factors of Equation 5.1 since 2 or more bits are missing. Since each dataset ($\mathcal{S}_{i,i+15}$) has at least 1 uncertain bit, there are as much equation solutions as square the number of uncertain bits. Hence, each attack Ω can return at least 4 different solutions for the system.

Furthermore, the adversary does not know the necessary amount of bits to eavesdrop to obtain pseudo-random sequences generated with

all $m = 8$ feedback polynomials, since this variable depends on the *trn* value. Finally, the PRNG design allows to discard a specific number of bits from the LFSR prior to output them as RN16. This parameter has been previously defined as d, and the effect on the system's security will be evaluated later on this section.

Next, we evaluate the worst possible attack to the PRNG. That is, considering an adversary performing an attack, with unlimited access to the communication channel and with no discarded bits from the LFSR ($d = 0$).

5.2.2 Description of Attacks

We assume an adversary that, by means of a non authorized reader, performs an eavesdropping attack, in a similar manner as performed in Section 3.2 and Section 3.3. We assume an adversary with unlimited access to the *reader-tag* channel, thus the adversary is able to eavesdrop long sequences of information. On the other hand, the adversary's PRNG design knowledge is defined by the public and secret parameters specified on Table 5.4, which are related with the security analysis of the system.

The attack to our PRNG proposal is defined in two steps: the *synchronization* step and the *polynomial detection* step. On one hand, the *synchronization* step analyzes a set of RN16 sequences initializing l simultaneous attacks from the first l bits. This fact gives us an initial probability of $P = \frac{1}{l}$ of matching the first bit generated by a feedback polynomial, but we can discard the attacks returning non primitive polynomials ($q(x) \notin \mathcal{P}$). Hence, the probability is increased at each cycle, and the synchronization is finished when only one of the attacks has returned primitive polynomials ($p(x) \in \mathcal{P}$) in

the subsequent cycles, which corresponds to the *synchronized* cycle. On the other hand, the *polynomial detection* step starts the attack in the position previously synchronized, and its aim is to detect the PRNG initial state (v_0) and the feedback polynomial combination ($\mathcal{P}_{\text{sel}}$) which generate the PRNG stream.

It is worth mention that in any case, the adversary has to face with the lack of information to solve the attack Ω which returns different solutions (since the 16-bit LFSR is updated before finishing the 16 cycles). Furthermore the adversary cannot predict the future feedback polynomial selections, since these selections are done depending on the value r of the *trn* bit. This characteristic adds uncertainty to the design since the adversary knows that two or three feedback polynomials are used for the generation of each RN16 (see Table 4.3 and Table 4.4), but has no information about the specific switching cycles.

Synchronization Step

An adversary eavesdropping pseudo-random sequences from our PRNG is not aware of the feedback polynomial update period. Based on the parametrization presented in Section 4.4, the feedback polynomial is updated at each $l = 15$ cycles, that means bits s_1 to $s_{15} \in \mathcal{S}$ have $P = \frac{1}{15}$ to be the first bit generated with an updated polynomial. Considering that all feedback polynomials used in the design are primitive, a possible approach to synchronize the attack is to discard attacks to $\mathcal{S}$ returning non primitive polynomials. For example, $q(x) = \Omega(\mathcal{S}_{1,16}||\mathcal{S}_{17,32})$ would discard s_1 as synchronization bit.

Figure 5.5 summarizes the *synchronization step* for $l = 15$. 15 attacks are initiated at each of the first 15 bits. Each attack returns 4 possible polynomials as a solution, which can be primitive ($p(x) \in \mathcal{P}$)

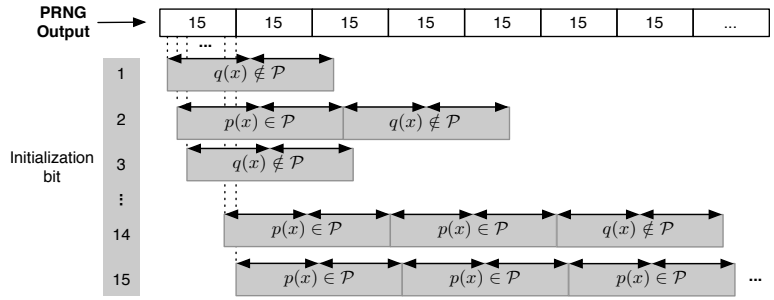


Figure 5.5: For a $l = 15$ polynomial update period, 15 simultaneous attacks (Ω) are initiated in the first 15 consecutive bits. The last attack sequence returning primitive polynomials gives the evidence of attack synchronization.

or either non primitive ($q(x) \notin \mathcal{P}$). Attacks returning four non primitive polynomials can be discarded since that means the analyzed bits cannot be generated through any primitive polynomial. Finally, the last attack returning at least one primitive polynomial gives us the evidence that these 15 bits are synchronized with the polynomial update period.

Figure 5.6 has been empirically obtained after synchronize 1,000 sequences generated with the proposed PRNG using different polynomial update periods ($11 \geq l \geq 15$). The figure depicts the probability of synchronize a pseudo-random sequence generated with the proposed PRNG, regarding the polynomial update period. As detailed in the *Attack Definitions*, decreasing the l value implies increasing the number of unknown bits to solve the attack Ω. Hence, more bits are necessary to obtain the feedback polynomials. For the longest polynomial update period ($l = 15$) about 76 bits are necessary to obtain a probability of 50% to synchronize the attack. For shorter

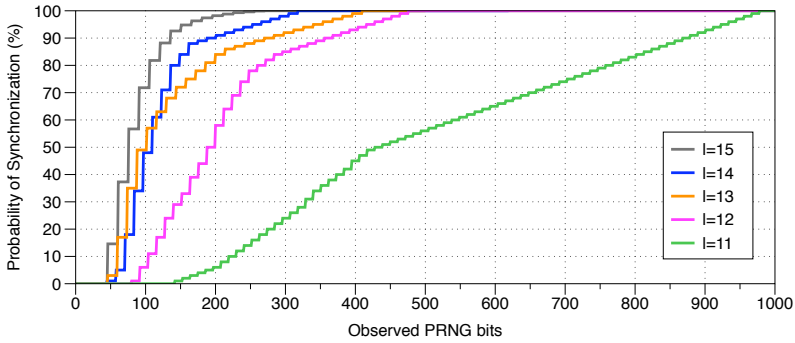


Figure 5.6: Attack synchronization step probability. The shorter the polynomial update period l, the higher the number of necessary bits to synchronize the PRNG sequence

l values the number of necessary bits quickly increases to reach the same probability, being necessary more than 400 bits for an update period of 11 cycles ($l = 11$).

However, the synchronization step is not enough to predict the PRNG output since the adversary may not have collected all m feedback polynomials (due to the unpredictability of the trn value). Hence, the adversary needs to perform the polynomial detection attack in order to obtain the necessary information to predict the PRNG output.

Polynomial Detection Step

In order to predict the PRNG output, the adversary has to find the m feedback polynomials $p_i(x)$ generating the PRNG output. After the synchronization of the PRNG dataset $\mathcal{S}$ the adversary knows that for each polynomial update cycle, $l - 1$ bits out of each polynomial cycle are generated by a specific polynomial. Hence, the polynomial detection method focuses on solving the attack Ω, iterated at each

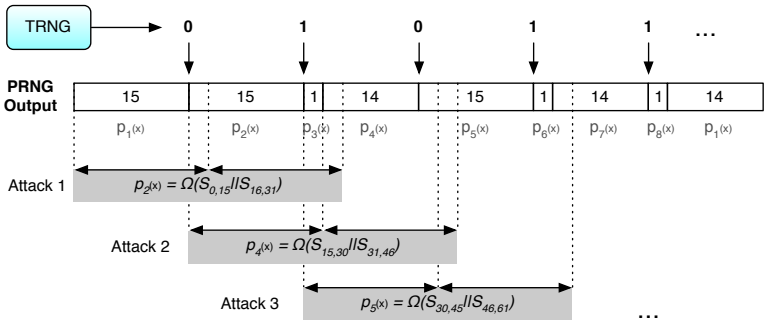


Figure 5.7: The polynomial detection step iterates the Ω attack at each polynomial update cycle, until finding the implemented polynomials in the scheme ($\mathcal{P}_{\text{sel}}$)

polynomial updated cycle, until finding the $\mathcal{P}_{\text{sel}}$ different polynomials. Thus, the longer the number of obtained RN16s, the higher the probability of each feedback polynomial to be solved.

Figure 5.7 illustrates a polynomial detection step example with $l = 15$, (once the attack has been synchronized) which obtains polynomials $p_2(x)$, $p_4(x)$ and $p_5(x)$. In this example, the feedback polynomial is updated one cycle faster than the LFSR, thus, two bits are missing from the necessary 32 bits to solve the attack Ω. Then, the theoretical probability to obtain a valid feedback polynomial $p(x)$ for this specific scheme is bounded by 0.25 and 1 (cf. Equation 5.2), since the two missing bits generates up to four different solutions. That is, the probability to solve the attack depends on the number of primitive polynomials $p(x)$ obtained at each attack iteration.

$$\frac{1}{2^{n-l+1}} \leq P(p_i(x) \in \mathcal{P}_{\text{sel}}) \leq 1 \tag{5.2}$$

Since Equation 5.2 depends on the specific set $\mathcal{S}$ of analyzed sequences, and the polynomial update period, we have empirically checked the probability to obtain $\mathcal{P}_{\text{sel}}$, executing 1,000 tests (initialized at random) over the same sequences previously generated for the synchronization step (generated with different polynomial update periods). Figure 5.8 depicts, on one hand, the probability of $\mathcal{P}_{\text{sel}}$ feedback polynomials being used with respect to the number of analyzed bits, and on the other hand the probability of success $P(p_i(x) \in \mathcal{P}_{\text{sel}})$ of the attack regarding the polynomial update period l used in the PRNG.

We can observe from the Figure 5.8 that after 180 bits there is a probability of 50% that the $\mathcal{P}_{\text{sel}}$ feedback polynomials have been used in the PRNG. On the other hand, after the synchronization step an adversary needs more eavesdropped cycles to obtain the $\mathcal{P}_{\text{sel}}$ feedback polynomials. Specifically, 280 bits are necessary to obtain a probability of about 50% to obtain all the feedback polynomials in the worst case ($l = 15$), and more than 1,100 bits if 11 cycles are used for the polynomial update ($l = 11$).

At this point, an adversary knows the feedback polynomials generating the PRNG output, but ignores in which order they are implemented in the tag, thus, further analysis with larger datasets must be done in order to address the feedback polynomial's order.

Attack for $d \neq 0$

The scenario with d public discarded values assumes that for each n generated bits by the PRNG, d bits are discarded from the LFSR prior to its transmission by the PRNG. To predict the PRNG output the adversary has to find the $m = 8$ feedback polynomials generating the PRNG output, like in the attack for $d = 0$. The adversary

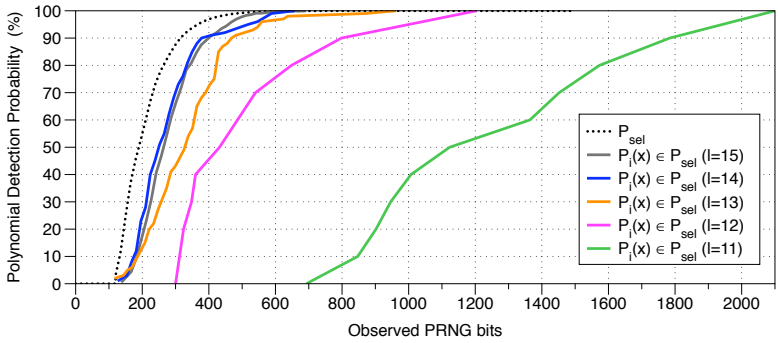


Figure 5.8: Attack detection step probability. The polynomial detection uncertainty can be increased by reducing the polynomial update period l

has now to face two different drawbacks: on one hand, the feedback polynomial switching cycle, and on the other hand the missing bits from the $n + d$ LFSR sequence.

Discarding bits from the LFSR means some bits are feedback shifted to the register but not transmitted, making the output from the PRNG more difficult to predict. The aim of this technique is to increase the security of the PRNG by hiding some sequences of the LFSR. If for the $d = 0$ attack, the adversary has to manage the uncertainty of the missing bits due to the update polynomial period, in this case the adversary must face the additional uncertainty of the discarded bits. Figure 5.9 depicts the PRNG scheme discarding 3 bits, showing that the resulting 16-bit sequences ($\mathcal{S}_i$) are generated by different polynomials. Hence, the effect of discarding bits has a similar effect than the polynomial updating period, but increasing the number of unknown bits.

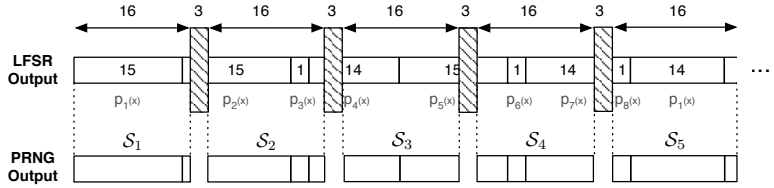


Figure 5.9: The discarding bits method adds uncertainty to the PRNG output

Based on the example of Figure 5.9, the adversary has to face with 2^4 to 2^6 additional unknown bits at each 16-bit sequence. Hence, up to 64 possible values for each attack Ω must be added to the polynomial update uncertainty. Hence both the synchronization and the polynomial detection steps are harder to complete.

5.2.3 Security Summary

The security of our PRNG proposal depends on its different configurable parameters, which are described in Table 5.4. The security analysis shown in this section takes the specific parameters used in the Hardware Specification Section (cf. Section 4.4), presenting different results regarding the polynomial update cycle parameter (l).

The results show that the success probability of the attack depends on uncertainty of the LFSR sequence which is not generated due to the polynomial update period. The shorter the polynomial update period, the smaller the probability to both synchronize or detect the $\mathcal{P}_{\text{sel}}$ on-board polynomials. Hence, appropriate values shall be found depending on the desired level of security.

The attack cost includes the cost of synchronization plus the polynomial detection steps, which implies solving the Ω function at each $2n$

bits, plus the different combinations of possible sequences depending on the polynomial update period. It is worth mention that our PRNG proposal offers better security properties for all l values, compared with the Che *et al.* proposal for the same LFSR size (cf. Section 3.1).

5.3 PRNG Power Consumption

The EPC Class 1 Generation 2 [25] is a low-cost RFID technology. It is designed to balance cost and functionality. Hence, EPC Gen2 tags are equipped with the minimum technology to provide the requested operations. This restriction is due to the cost limitation, but also to reduce the power consumption of the tags. Since the EPC Gen2 tags are designed to work at relatively high distances (about 5 meters), the tags shall consume a small amount of energy to enable its performance. In the remainder of this section the power consumption of our PRNG proposal is evaluated, based on the hardware design presented in Section 4.4.

A RFID tag consists of several components, each of which occupies some physical space and consumes a certain amount of power [27]. A simple representation of a RFID tag is shown in Figure 5.10.

The analog block of a RFID tag adapts the incoming signal from the antenna to power up the tag, and to adapt the information sent from the reader to the digital circuit of the tag. In the same way, the analog block receives the information from the digital circuit to be sent to the reader, which needs to be modulated and adapted to be sent through the antenna as a radio-frequency wave.

The power collected by the antenna is very unstable and unreliable, thus, it cannot be directly used to fed the digital circuit power supply

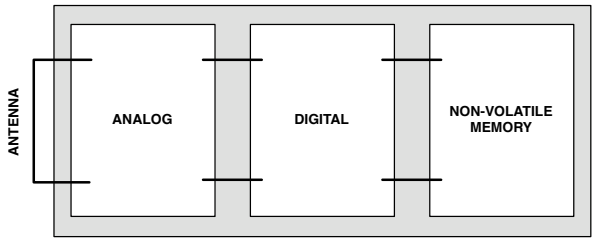


Figure 5.10: Layout of a RFID tag. The antenna supplies energy and data to the analogical block, which adapts the signal to the digital block, which in turn, manages the memory block.

requirements. The analog block includes a rectifier, a charge pump and a voltage regulator to convert the energy from the electric field to a constant voltage source. Hence, regardless of the distance between reader and tag (if inside the communication range), the analog part supplies the same constant voltage to the digital circuit. The internal clock signal can be built from a PLL or a ring oscillator, being the common clocking value around 100 kHz [27, 29, 72] as introduced in the previous sections.

Very-large-scale integration (VLSI) is the process of designing integrated circuits (ICs) by combining thousands of transistors, also known as logical gates (GEs). Even after choosing a technology, VLSI designers need to minimize the power dissipation while meeting other design objectives such as speed, chip, area, device cost, and reliability [50]. Next, the key factors in power consumption are introduced.

5.3.1 Key Factors in Power Consumption for CMOS Technology

The available power budget in a RFID tag is closely related with the hardware configuration. In fact, we can define the functionality of a RFID tag (i.e. the area requirements) regarding the available power at a certain distance. This gives an idea of the available power for activating the essential parts of a RFID tag, thus, also the remaining power for added functionalities, which can be *translated* to area or logical gates (GEs) as shown in the previous section.

The available power at a tag depends on the tag configuration, but also on the distance between reader and tag. The less power a passive RFID tag consumes, the longer the operational range of the devices [29]. The ultra high frequency (UHF) band is regulated by the European Telecommunications Standards Institute (ETSI) [45] and the signal strength cannot exceed two Watts for RFID communications. UHF RFID tags generally require around 100 μW for its general performance [27], thus, the performing distance for this technology is about 5 meters.

Operating Frequency

The tag's analog front-end includes a storage capacitor (C_{stor}) which acts as the battery supplier for the EPC Gen2 tag. It is charged from the radio-frequency signal emitted by the interrogator, and discharged by the operation of the tag IC, more than one thousand million times per second for UHF frequencies. CMOS power consumption is determined mostly by dynamic power consumption which concentrates around clock edges, thus, proportional to the operating frequency. Hence, in order to keep the average power consumption

low, the IC clock signal must be significantly lower than the UHF signal frequency. Most digital RFID circuits are clocked above 100 kHz, thus the capacitor has to provide power for a few nanoseconds of dynamic power consumption every 10 μs [29]. Hence, the power consumption of a LFSR based stream cipher is directly dependent on the frequency operation [83].

Capacitive Load

Capacitive load (C_L) limits the system in terms of frequency signal. Regarding the scarce available power in EPC Gen2 tags, this issue is of major importance. If the output signal of the driving gate is a logic '1' the capacitance of the load gate is charging through the output resistance of the driving gate. However, if the output signal of the driving gate is a logic '0', the capacitance is discharging [29]. Hence, signals in logic gates CMOS components imply charging (and discharging) times associated with the output resistance of the driving gate and the input capacitance of the load gates.

When generating logic signals, CMOS transistors present a predominantly capacitive load to the driving gate. The capacitive load is the equivalent capacitance considering all the logical gates included in the circuit. The more load gate inputs are added to the system, the higher the total capacitance. This increase in capacitance is directly related to the charging and discharging times, thus, also the power dissipation. Reducing the maximum frequency at which the system can be operated reduces the total dissipated power.

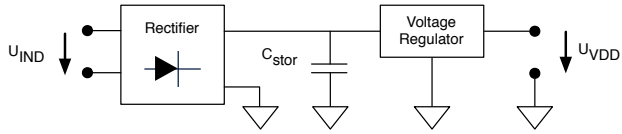


Figure 5.11: Power supply of passive RFID tags [29]

Voltage Source

Since RFID tags are passive devices, the necessary power for their operations is supplied through the radio-frequency waves emitted by the reader. Depending on the distance between reader and tag, the available power can vary in orders of magnitude. Due to this reason tag's analog front end includes some components which are intended to adapt and regulate this unreliable radio-frequency induced power to a regulated power source. Figure 5.11 depicts the schematic of a RFID power supply. The rectifier is used to transform an alternate current to a direct current by blocking the negative part of a waveform, generally using a solid state diode. The capacitive storage has previously been introduced and it is intended to accumulate the energy from the rectified waveforms, to be discharged in the subsequent period. Finally, the voltage regulator manages to adapt the voltage from the capacitor output, to an appropriate reference voltage (U_{VDD}). Najafi *et al.* deepens on the voltage regulator and rectifier for UHF EPC Gen2 tags [66]. The reference voltage in RFID tags is usually 1.5 V for 0.35 μm CMOS technologies, and 1 V for newer technologies (130-180 nm) [29].

The described properties of the circuit providing the supply voltage for the digital part of the circuit lead to the following considerations for the power consumption[29]:

- Keep the average power consumption P_{avg} low
- No clock cycle should consume excessive power

The first consideration is based on the scarce amount of power received at the antenna of the tag. The second consideration regards the prevention of undesired resets of the system. If a clock cycle consumes an excessive amount of power, the storage capacitor drops all the current leaving the subsequent cycles without enough energy.

Power vs. Energy Supply in RFID

The energy used for a (cryptographic) operation depends on the average power (P_{avg}) and the duration t of the computation (cf. Equation 5.3). Hence, energy consumption is one of the most important parameters for battery-powered devices, which depends on the time that the battery is able to supply the requested power. However, for passively powered devices (such as the EPC Gen2 tags) the average power transmitted from the interrogator to the tag is small but in general terms the interrogator can supply the power arbitrarily long. Thus, the energy does not play an important role in the EPC Gen2 tags as long as there is enough time available to do the computation [29].

$$E = P_{\text{avg}} t \tag{5.3}$$

The reference parameter for the EPC Gen2 cryptographic functionality is, together with the number of available cycles for computation, the average power supply (P_{avg}). When implementing cryptographic circuits on RFID tags it is desired that the cryptographic functionality does not limit the operating range, thus, the tags have to cope

with the limited power budget [29]. The remainder of this section deepens on the measure of the power consumption of our proposed PRNG.

5.3.2 Power Consumption Evaluation

Due to the RFID systems characteristics any security mechanism design must be implemented on both interrogator and tag IC, implying an extra power consumption on both sides. On the interrogator side the implementation of security tools does not meet any restriction since the computation capabilities and power supply are not limited. On the tag side, any extra circuit implementation results on more silicon area, thus more power consumption and an increment of the tag's cost.

Standard CMOS transistors is the current choice of most digital circuit designs built for low power consumption and robustness. Hence, it is appropriate to consider power consumption analysis based on an implementation using CMOS technology [85].

An important aspect of security implementations in the design stage is to ensure that the power dissipation of the IC does not exceed the available power budget for its execution. Feldhofer *et al.* have estimated the average power budget for cryptographic operations in 4 μW at five meters to the interrogator [29].

There are different techniques for measuring power consumption in CMOS circuits based on estimation methods [56] and software simulation [50]. The remainder for this section presents two different estimations based on each technique.

Dynamic CMOS Power Estimation

The total power consumption of a CMOS circuit is the sum of static and dynamic power consumption. The static power consumption mainly depends on the size of the circuit and is very small, thus, it can be ignored for our considerations [29]. While the use of direct methods to measure power dissipation may be possible, a simple method for estimating the dynamic power dissipation is based on formulating the power loss during the charging and discharging of capacitances [85]. Equation 5.4 models the average power dissipation of a system composed of a small number of logic gates.

$$P = p_{0\to1} C_L V_{DD}^2 f_{clk} \tag{5.4}$$

C_L is the load capacitance along the critical path and $p_{0\to1}$ represents the logic state transition from low to high (or vice versa) in a single clock cycle. The combination of $p_{0\to1}$ and C_L can also be stated as the average capacitance switched during each clock cycle. f_{clk} represents the clock frequency and V_{DD} is the system supply voltage. Design measures for lowering the power consumption result from minimizing the factors in this equation. It is difficult to apply this formula to ICs of large sizes due to the difficulty to state the number of logic state transitions for each clock cycle. However, it is adequate for estimating the power consumption in small hardware, especially if circuit design tools can be used to evaluate the capacitance estimates [85]. Furthermore, Macci *et al.* present in [56] a worst-case power consumption estimation of CMOS circuits using models based on symbolic neural networks.

Based on measurements presented by Etrog *et al.* [27] the load capacitance (C_L) for each GE is approximately 3 fF. The voltage source and operating frequency have been previously stated in 1 V and 100 kHz for passive low-cost RFID. If we consider that about half GE are switched for each clock cycle, Equation 5.4 returns an estimation of 67.5 nW of average power consumption for our PRNG proposal. The estimation is consistent with similar designs present in the literature [14], and under the available budget of 4 μW of power consumption for cryptographic operations for UHF technologies.

Power Consumption Simulation

After defining the design of the digital core of the PRNG based on GEs (cf. Section 4.4), we conduct an electronic circuit simulation of our proposed construction. The simulation language *SPICE* [24] is used to simulate the circuit, and the *LTSpice IV* software [94] is used to represent the circuit using logical gates. The resulting simulation also allows us to demonstrate the fundamental concepts of our construction and to confirm its validity as a stand-alone device. Figure 5.12 and Figure 5.13 depicts the different modules of the proposed PRNG.

Power dissipation is one of the most important factors in VLSI design and its technology choice. Therefore, accurate simulation of CMOS power dissipation using languages such as SPICE is highly desirable [50]. We evaluate the hardware specification of our PRNG proposal presented in Section 4.4. To precisely evaluate the power consumption of our design it is necessary to provide libraries with parameter models of the specific technology which is simulated. These libraries include a variety of CMOS parameters modeling the transistor's behavior and parasitic circuit elements. Using library models, which

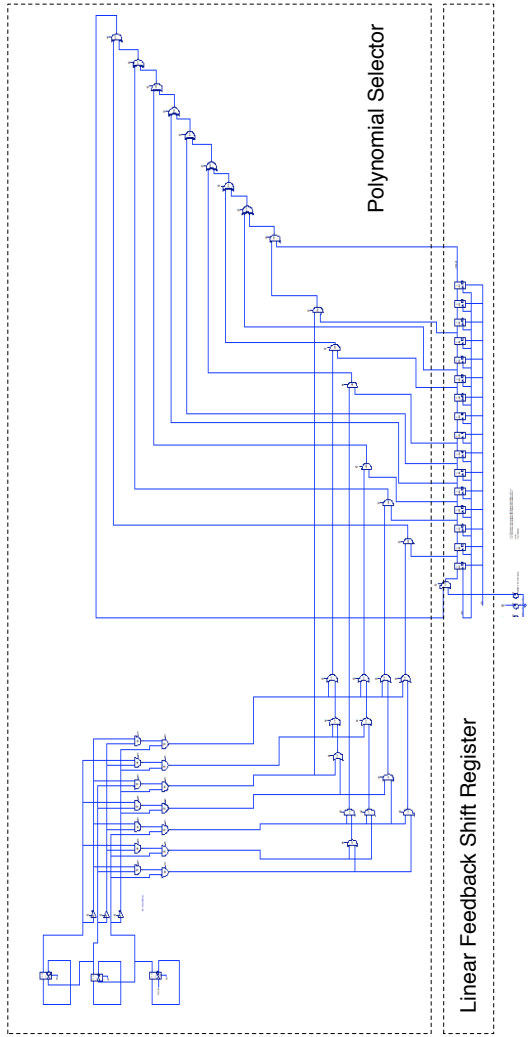


Figure 5.12: Hardware design of the LFSR and the Polynomial Selector of our proposal realized with *LTSpice IV*

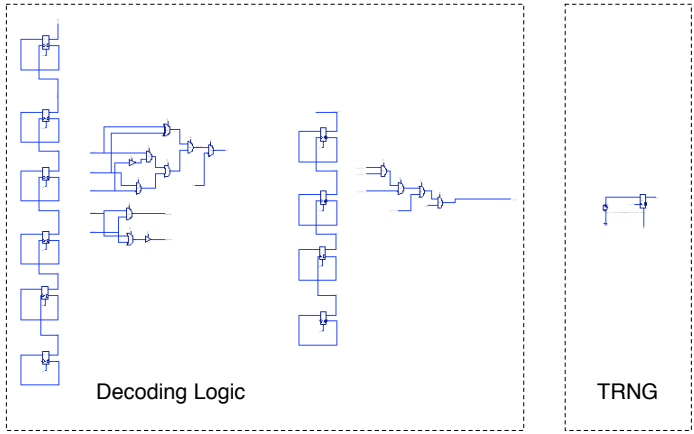


Figure 5.13: Hardware representation of the Decoding Logic and the TRNG modules of our proposal realized with *LTSpice IV*

can be theoretically modeled or hardware measured [24], the precision of the calculations is improved to effectively simulate the circuit like a real fabricated device.

Current UHF RFID products are fabricated on 180 and 130 nm CMOS processes [29]. Since these technologies (specially the 130 nm CMOS processes) are relatively new, it is difficult to find appropriate CMOS models matching the specific technology. For the power consumption simulation we use the *Predictive Technology Model* (PTM) libraries [98] provided by the Nanoscale Integration and Modeling Group from the Arizona State University, which provides CMOS models for 130 nm processes.

The analysis target the average power consumption of our proposed PRNG, in order to evaluate its implementability in a real EPC Gen2 tag. Figure 5.14 depicts the PRNG power consumption during one 16-bit sequence, generated with *LTSpice* (using the PTM libraries).

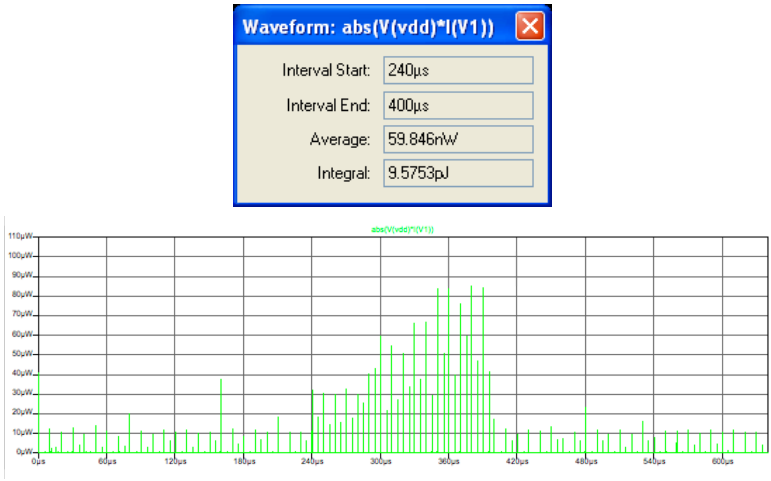


Figure 5.14: *Spice* power consumption simulation. Power dissipation is concentrated around the internal clock cycles.

Power dissipation is concentrated around the clock cycles, specially between 240 μs and 400 μs when the major part of the PRNG operations are performed, thus, the CMOS gates are switched (first and last cycles are reserved for read and write memory operations). The simulated average power consumption for the 16-bit sequence generation is 59.8 nW, which is consistent with the Dynamic CMOS Power Estimation. The simulated power consumption is also under the average power consumption requirements for cryptographic operations in RFID tags proposed by Feldhofer et al. [29].

Non-volatile memory power consumption is not considered in this evaluation. Reading and writing operations are currently expensive in terms of energy. Hence, we reserve a considerable amount of clock cycles for these operations (cf. Section 4.4) in our design. Additional techniques such as multi-voltage power source [100], non-

volatile memory technologies such as *Zuma* [35], plus the gradual improvement of the CMOS processes promise a reduction in the necessary energy to read and write from non-volatile memories.

5.4 Chapter Summary

The evaluation of a cryptographic tool covers different areas related to its design. On one hand, the algorithm or circuit shall ensure suitable statistical and security properties based on its application and the required level of security. On the other hand, its suitability to the specific technology in terms of performance is also of major concern. Aspects such as area implementation (thus, also fabrication cost), power consumption and execution time are basic parameters when evaluating a cryptographic tool.

We have evaluated the statistical properties of the proposed generator following two methodologies. First, we applied the NIST 800-22 statistical test suite for random and pseudo-random number generators for cryptographic applications to 200 MB of pseudo-random data generated with the proposed PRNG. Results show that the sequences satisfy the applied tests, thus, no evidences of statistical non-randomness are detected. Also related to the statistical analysis, we have analyzed the three requirements for pseudo-random number generation specified in the EPC Gen2 standard. These requirements, refer to the probability of appearance of RN16s, probability of simultaneous identical identification, and RN16 prediction probability. The three requirements have been tested using the same pseudo-random sequences used in the NIST analysis demonstrating its suitability to the EPC Gen2 standard.

Since the proposed PRNG combines a LFSR with a non-linear technique (multiple polynomials) which was not used before in security scenarios, no references or previous security analysis can be provided. Hence, a first security analysis based on different adversary scenarios is included in this section. Based on the classification of public and private parameters and the implementation proposed in Chapter 4, a synchronization and detection steps attack is evaluated regarding different values of the polynomial update cycle and the discarding bits value.

Finally, the simulation of our generator as an electronic stand alone circuit has confirmed that the proposal fulfills the power constraints imposed by the UHF technology and the EPC Gen2 standard regarding power consumption. Hardware models reproducing real parameters of the technology processes used to fabricate UHF tags have been used for this analysis. Hence, we can confirm that our PRNG is properly gated to avoid unnecessary state switching on the gates, thus reducing the overall power consumption and time execution.

Our main contributions include:

- A statistical analysis based on the NIST statistical test suite and the randomness requirements of the EPC Gen2 standard
- A security analysis of our PRNG proposal, which is the first analysis based on a LFSR with non-linearity provided the multiple polynomial technique.
- A performance evaluation using the SPICE language for circuit simulation, demonstrating the suitability of our proposal in terms of power consumption.

6

Conclusions

6.1 Concluding Remarks

Electronic Product Code (EPC) Gen2 systems represent one of the most pervasive low-cost technologies in the field of Radio Frequency Identification (RFID). The main feature of EPC Gen2 technology is the tag reduced price (predicted for under 10 US dollar cents) which means a compromise between cost and functionality. Two major issues limit their computation capabilities: economical and technical. On one hand there are billions of RFID tags in the field, thus, the production costs have to be low. On the other hand the available power for operating passive RFID tags is limited because the power for operation has to be supplied over the air interface.

Security and privacy are nowadays issues of concern for low-cost RFID systems. The communication between tags and readers is made

in a potentially insecure channel, hence, any compatible reader can access the communication between tags and readers in its read range. Thus, EPC Gen2 system communications is potentially vulnerable to undesired access to the communication between readers and tags. Particular emphasis is made on the uniqueness of the EPC Gen2 system communications model, which only provides security measures for the content transmitted in the reader-to-tag channel.

Pseudo-random number generators (PRNGs) are the crucial components that guarantee the confidentiality of EPC Gen2 RFID communications. In this dissertation, we have described the problems of using linear feedback shift registers (LFSRs) as underlying mechanisms for the implementation of low-cost PRNGs. Without appropriate measures, the linearity of LFSR-based PRNGs lead to insecure implementations. We have analyzed a cost-effective PRNG proposal for EPC Gen2 devices presented by Che *et al.* The proposal combines thermal noise signal modulation and an underlying LFSR. We have indeed demonstrated that the proposal does not handle properly the inherent linearity of the resulting PRNG. We have described an attack to obtain the feedback polynomial function of the LFSR. This allows us to synchronize and to predict the resulting sequences generated by the Che *et al.* PRNG. Furthermore, we have presented the implementation of a practical attack in a real EPC Gen2 scenario, by means of a compatible Gen2 reader, and a programmable Gen2 tag implementing the Che *et al.* PRNG.

Based on the techniques used to analyze the Che *et al.* PRNG, and the lack of information on secure PRNG for resource-constrained devices in the literature, we also have empirically analyzed the pseudo-random sequences generated by commercial EPC Gen2 tags from the major manufacturers.

With the obtained knowledge, we have focused on the improvement of the security for low-cost passive RFID tags. A new PRNG design for EPC Gen2 devices has been presented. This generator is based on a 16-bit LFSR designed as a multiple-polynomial LFSR architecture. This leads to different feedback primitive polynomials fed by a physical source of randomness for handling the inherent linearity of the LFSR module. We have validated that the resulting generator satisfies the randomness requirements imposed by the EPC Gen2 standard. Furthermore, a security analysis of our proposed PRNG is presented. The results confirm an improvement of the security compared to other PRNG designs like the Che *et al.* proposal.

Finally, the simulation of our generator as an electronic stand alone circuit has confirmed that the proposal fulfills the hardware constraints imposed by the EPC Gen2 standard such as amount of logic gates, time consumption and data transmission rate. Our simulation confirms, moreover, that the hardware complexity of our proposed generator has a much simpler hardware implementation than previous schemes reported in the literature.

6.2 Results of this Dissertation

The main results of this dissertation are stated in the following points:

- A novel method for extracting 16-bit pseudo-random sequences from the reader to tag EPC Gen2 communications using a Demo Tag device. This procedure is based on sniffing standard EPC commands over the wireless communication, and it can be applied to any EPC tag communication to eavesdrop the output of a PRNG [59].

- A linearity vulnerability found in a PRNG proposal by Che *et al.* Such vulnerability has been used to attack the generator using a small amount of generated bits [60].

- An analysis of EPC Gen2 commercial ICs pseudo-random number generators based on the NIST statistical test suite for randomness, finding evidence of non-randomness in the analyzed sequences. Based on theses analysis, we propose a new method for the measurement of the first requirement for random number generation regarding the frequency pseudo-random sequences generation.

- The definition of a new PRNG based on a LFSR for security applications, compliant with the EPC Gen2 technology, handling the LFSRs inherent linearity with a multiple-polynomial architecture. An evaluation of primitive polynomials of degree 16 returns up to 2^{72} possible polynomial combinations based on our PRNG design. The proposed PRNG has 452 GE being smaller than other lightweight proposals in the literature [31].

- Finally, an exhaustive evaluation of our proposed PRNG. First, a statistical analysis based on the NIST statistical test suite and the randomness requirements of the EPC Gen2 standard has been realized. Furthermore, we present a security analysis of our PRNG proposal, assuming an adversary with some knowledge of the internal design of our PRNG. Finally, we present a power consumption evaluation using the SPICE language for electronic circuits simulation, demonstrating the suitability of our proposal for EPC Gen2 technologies.

6.3 Future Research

Following, we review a set of future research directions we are interested in from the research performed in this dissertation:

- **Empirical analysis of PRNGs from commercial EPC Gen2 tags**
  In this dissertation we present a statistical analysis of pseudo-random sequences obtained from three commercial EPC Gen2 tags. We plan to extend the analysis to other models and IC manufacturers, also using other statistical evaluation tools like FIPS suite of tests. Furthermore, we are also interested in the cryptanalysis of these pseudo-random sequences with the aim to obtain a deeper knowledge of the current security state of the art by means of reverse engineering.

- **PRNG proposal**
  The PRNG proposed in this dissertation is intended for its integration in a resource constrained devices. Specifically, we have parameterized its design to be suitable to the EPC Gen2 standard for low-cost RFID. As a future work, we are interested into test the PRNG performance using different design values such as LFSR size or the number of on board polynomials, looking for an optimal trade-off between security and functionality. New PRNG designs are not limited to the EPC Gen2 standards but other RFID or wireless sensors platforms.

  Hardware implementation issues is also inside our future research program. Once we have checked the suitability of our design in terms of area and power consumption, further analysis must be done to optimize our design. Work in the memory storage part which has not been considered in this work, is

one of the possibilities. Also the improvement of integration techniques, like applying sleep logic to the PRNG hardware to improve its efficiency, are topics of our interest.

- **Security and privacy in other resource constrained RFID or wireless sensor platforms**

  Finally, we do not want to limit our research to the EPC Gen2 technology. Despite its wide field of research, the analysis and improvement of the security of other low-cost wireless technologies are also part of our future research objectives. Technologies like Near Field Communication (NFC) which is increasingly used as wireless payment system, High Frequency (HF) technologies for user identification , mixed sensing-RFID platforms such as WISP or wireless sensor networks, offer interesting research perspectives for the forthcoming years.

Bibliography

[1] Atmel Corporation. [Online] Available at http://www.atmel.com/.

[2] CAEN RFID. [Online] Available at http://www.caen.it/rfid/.

[3] IAIK - Graz University of Technology. UHF RFID Demo Tag. [Online] (Last access Feb. 2010), Available at http://jce.iaik.tugraz.at/sic/Products/.

[4] National Institute of Standards and Technology. Random number generation. [Online] (Last access Feb. 2010), Available at http://csrc.nist.gov/groups/ST/toolkit/rng/.

[5] Rowley Crossworks IDE. Crossworks v1.4 and v2.0 for AVR. [Online] Available at http://www.rowley.co.uk/.

[6] ISO/IEC 18000-6: Radio frequency identification for item management - parameters for air interface communications at 860 MHz to 960 MHz. Technical report, International Organization for Standardization (ISO), [Online] Available at http://www.iso.org/, 2006.

[7] M. Aigner, T. Plos, M. Feldhofer, C. Tutsch, C. Ruhanen, Y. Na, S. Coluccini, and M. Tavilampi. BRIDGE - Building Radio frequency IDentification for the Global Environment. Report on first part of the security WP: Tag security (D4.2.1)., Jul. 2007. [Online] Available at `http://www.bridge-project.eu/`.

[8] S.R. Aroor and D.D. Deavours. Evaluation of the state of passive UHF RFID: An experimental approach. *IEEE Systems Journal*, 1(2):168–176, 2007.

[9] R.J. Baker. *CMOS: Circuit design, layout, and simulation.* Wiley-IEEE Press, 2007.

[10] G. Balachandran and R. Barnett. A 440nA true random number generator for passive RFID tags. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 55(11):3723–3732, Dec. 2008.

[11] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede. Public-Key Cryptography for RFID-Tags. In *Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 217–222. IEEE Computer Society, 2007.

[12] E.R. Berlekamp. *Algebraic coding theory.* McGraw-Hill New York, 1968.

[13] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer Berlin / Heidelberg, 2007.

[14] A. Boni and A. Facen. Ultra low-voltage analog circuits for UHF RFID devices in 180 nm CMOS technology. *Analog Integrated Circuits and Signal Processing*, 63(3):359–367, 2010.

[15] M. Buettner and D. Wetherall. An empirical study of UHF RFID performance. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 223–234. ACM, 2008.

[16] L. Buttyan and J.P. Hubaux. *Security and Cooperation in Wireless Networks*. Cambridge University Press, 2007. Available at `http://secowinet.epfl.ch/`.

[17] L. Catarinucci, R. Colella, M. De Blasi, L. Patrono, and L. Tarricone. Improving item-level tracing systems through Ad Hoc UHF RFID tags. In *2010 IEEE Radio and Wireless Symposium (RWS)*, pages 160–163, 2010.

[18] W. Che, H.Deng, X. Tan, and J. Wang. *Networked RFID Systems and Lightweight Cryptography, Chapter 16*, chapter A Random Number Generator for Application in RFID Tags, pages 279–287. Springer, Nov. 2008.

[19] C. L. Chen. Linear dependencies in linear feedback shift registers. *Computers, IEEE Transactions on*, C-35(12):1086–1088, Dec. 1986.

[20] Confidex. Casey inlay. [On-line], 2010. Available at `http://www.confidex.fi/images/stories/pdf/-product_datasheets/Casey_Datasheet.pdf`.

[21] D. Coppersmith, H. Krawczyk, and Y. Mansour. The shrinking generator. In *Advances in Cryptology - Crypto'93*, pages 22–39. Springer, 1994.

[22] C. De Canniere and B. Preneel. Trivium specifications. Technical report, Ecrypt, 2008.

[23] M. Dichtl, B. Meyer, and H. Seuschek. SPICE Simulation of a Provably Secure True Random Number Generator. Cryptology ePrint Archive, Report 2008/403, 2008. Available at `http://eprint.iacr.org/`.

[24] EECS. SPICE Website. UC Berkeley, (Last access) 2010. Available at `http://bwrc.eecs.berkeley.edu/classes/icbook/spice/`.

[25] EPCglobal. EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860-960 MHz. [Online] Available at `http://www.epcglobalinc.org/standards/`.

[26] EPCglobal. The EPCglobal Website. [On-line], (Last access) 2010. Available at `http://www.epcglobalinc.org/`.

[27] J. Etrog, M.J.B. Robshaw, and O. Savry. The possibilities and limitations of cryptography in constrained devices. Technical report, INRIA and Orange Labs, 2009. Deliverable for RFID-AP.

[28] M. Feldhofer and C. Rechberger. A case against currently used hash functions in RFID protocols. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, volume 4277 of *LNCS*, pages 372–381. Springer Berlin / Heidelberg, 2006.

[29] M. Feldhofer and J. Wolkerstorfer. Hardware Implementation of Symmetric Algorithms for RFID Security. In P. Kitsos and

Y. Zhang, editors, *RFID Security*, pages 373–415. Springer US, 2009.

[30] F. Garcia, G. Koning, R. Muijrers, P. van Rossum, R. Verdult, R. Wichers, and B. Jacobs. Dismantling MIFARE Classic. In *Computer Security - ESORICS 2008*, pages 97–114, 2008.

[31] J. Garcia-Alfaro, J. Herrera-Joancomarti, and J. Melia-Segui. A Multiple-Polynomial LFSR based Pseudorandom Number Generator Design for EPC Gen2 Systems. Mitacs Workshop on Network Security & Cryptography, Toronto (Canada), Jun. 2010.

[32] A. Gautham. Practical Evaluation and Analysis Of Passive UHF RFID Tags. Master's thesis, University of Texas at Arlington, 2008.

[33] P.R. Geffe. How to protect data with ciphers that are really hard to break. *Electronics*, 46(1):99–101, 1973.

[34] R. Ghosal, D.C. Ranasinghe, and P.H. Cole. Bit-oriented generators for wireless sensor networks and low cost RFID Transponders. In *Proceedings of the Fourth IASTED International Conference on Advances in Computer Science and Technology*, pages 230–235. ACTA Press, 2008.

[35] R. Glidden, C. Bockorick, S. Cooper, C. Diorio, D. Dressler, V. Gutnik, C. Hagen, D. Hara, T. Hass, and T. Humes. Design of ultra-low-cost UHF RFID tags for supply chain applications. *Communications Magazine, IEEE*, 42(8):140–151, 2004.

[36] S.W. Golomb and L.R. Welch. *Shift register sequences*. Aegean Park Press Laguna Hills, CA, 1982.

[37] M. Haahr. True random number service, 2010. (Last access Feb. 2010).

[38] G. Hancke. Eavesdropping Attacks on High-Frequency RFID Tokens. In *Fourth Conference on RFID Security, Budapest, Hungary*, pages 100–113, 2008.

[39] M. Hell, T. Johansson, and W. Meier. Grain: a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing*, 2(1):86–93, Nov. 2007.

[40] S. Hellebrand, J. Rajskia, S. Tarnick, S. Venkataraman, and B. Courtois. Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers. *Computers, IEEE Transactions on*, 44(2):223–233, Feb. 1995.

[41] T. Herlestam. On functions of linear shift register sequences. In *Advances in Cryptology - EUROCRYPT 95*, volume 219/1986 of *LNCS*, pages 119–129. Springer, Jan. 1995.

[42] D. Holcomb, W. Burleson, and K. Fu. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In *Proceedings of the Conference on RFID Security*, Jul. 2007.

[43] Impinj. Monza 3 ic. [On-line], 2010. Available at `http://www.impinj.com/products/tag-chips.aspx`.

[44] Mathworks Inc. Matlab and Simulink Website. Cryptology ePrint Archive, Report 2008/403, (Last access) 2010. Available at `http://www.mathworks.com/`.

[45] European Telecommunications Standards Institute. The ETSI website. Available at `http://www.etsi.org/`.

[46] RFID Journal. NXP Doubles Memory on Gen2 RFID Chips. [On-line], 2010. Available at `http://www.rfidjournal.com/article/print/6829`.

[47] A. Joux. *Algorithmic Cryptanalysis*. Chapman & Hall/CRC, Taylor & Francis Group, 2009.

[48] A. Juels. RFID security and privacy: A research survey. *IEEE Journal on Selected Areas in Communication*, 24(2):381–394, Feb. 2006.

[49] A. Juels, R.L. Rivest, and M. Szydlo. The blocker tag: selective blocking of RFID tags for consumer privacy. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 103–111. ACM, 2003.

[50] S.M. Kang. Accurate simulation of power dissipation in VLSI circuits. *IEEE Journal of Solid-State Circuits*, 21(5):889–897, 1986.

[51] A. Klimov and A. Shamir. A new class of invertible mappings. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 470–483. Springer, 2003.

[52] K. Koscher, A. Juels, T. Kohno, and V. Brajkovic. EPC RFID Tags in Security Applications: Passport Cards, Enhanced Drivers Licenses, and Beyond. 2009. Withe paper, Available at `http://www.rsa.com/`.

[53] H.R. Lee and D.W. Hong. The tag authentication scheme using self-shrinking generator on RFID system. *International Journal of Applied Science, Engineering and Technology*, 3:33–38, 2007.

[54] D.H. Lehmer. Mathematical methods in large-scale computing units. In *Proceedings of the Second Symposium on Large*

Scale Digital Computing Machinery, Harvard University Press, Cambridge, MA, pages 141–146, 1951.

[55] M. Lehtonen, A. Ruhanen, F. Michahelles, and E. Fleisch. Serialized TID Numbers–A Headache or a Blessing for RFID Crackers? In *the IEEE RFID 2009 Conference, Orlando, Florida*, 2009.

[56] E. Macci and M. Poncino. *Digital Integrated Circuits*, chapter Estimating worst-case power consumption of CMOS circuits modelled as symbolic neural networks. Prentice Hall, second edition, 2002.

[57] J.L. Massey. Shift-register synthesis and BCH decoding. *IEEE Trans. Inf. Theory*, 15:122–127, 1969.

[58] W. Meier and O. Staffelbach. The self-shrinking generator. In *Advances in Cryptology - EUROCRYPT'94*, pages 205–214. Springer, 1995.

[59] J. Melia-Segui, J. Garcia-Alfaro, and J. Herrera-Joancomartí. A Practical Implementation Attack on Weak Pseudorandom Number Generator Designs for EPC Gen2 Tags. *Wireless Personal Communications*, 2010. DOI: 10.1007/s11277-010-0187-1.

[60] J. Melia-Segui, J. Garcia-Alfaro, and J. Herrera-Joancomarti. Analysis and Improvement of a Pseudorandom Number Generator for EPC Gen2 Tags. In R. Sion et al., editor, *Financial Cryptography and Data Security*, volume 6054 of *Lecture Notes in Computer Science*, pages 34–46. Springer, 2010.

[61] A.J. Menezes, P.C. Van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography*. CRC Press, 1997.

[62] R. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsacks. *Information Theory, IEEE Transactions on*, 24(5):525 – 530, 1978.

[63] M. Mihaljević. A faster cryptanalysis of the self-shrinking generator. In *Information Security and Privacy*, pages 182–189. Springer, 1996.

[64] Motorola. RFID technology and EPC in retail. Whithe papers. (last access Feb. 2010).

[65] Motorola. XR Series RFID Readers - Product Guide, 2004. (Last access Feb. 2010).

[66] V. Najafi, S. Mohammadi, V. Roostaie, and A. Fotowat-Ahmady. A dual mode UHF EPC Gen 2 RFID tag in 0.18 μm CMOS. *Microelectronics Journal*, 41(8):458 – 464, 2010.

[67] P.V. Nikitin, K.V.S. Rao, R. Martinez, and S.F. Lam. Sensitivity and impedance measurements of UHF RFID chips. *IEEE transactions on microwave theory and techniques*, 57(5):1297, 2009.

[68] K. Nohl, D. Evans, S. Starbug, and H. Plötz. Reverse-engineering a cryptographic RFID tag. In *Proceedings of the 17th conference on Security symposium*, pages 185–193. USENIX Association, 2008.

[69] NXP. Nxp website. [On-line], 2010. Available at `http://www.nxp.com/`.

[70] NXP. U-code g2xl leaflet. [On-line], 2010. Available at `http://www.nxp.com/documents/leaflet/75016225.pdf`.

[71] ECRYPT Network of Excellence. The eSTREAM project. Available at `http://www.ecrypt.eu.org/stream/`.

[72] C. Paar, A. Poschmann, and M.J.B. Robshaw. New Designs in Lightweight Symmetric Encryption. In P. Kitsos and Y. Zhang, editors, *RFID Security*, pages 349–371. Springer US, 2009.

[73] S.K. Park and K.W. Miller. Random number generators: good ones are hard to find. *Communications of the ACM*, 31(10):1192–1201, 1988.

[74] P. Peris-Lopez. *Lightweight Cryptography in Radio Frequency Identification (RFID) Systems*. PhD thesis, Universidad Carlos III de Madrid, 2008.

[75] P. Peris-Lopez, J. Hernandez-Castro, J. Estevez-Tapiador, and A. Ribagorda. LAMED a PRNG for EPC Class-1 Generation-2 RFID specification. *Computer Standards & Interfaces*, pages 88–97, Dec. 2007.

[76] C.S. Petrie and J.A. Connelly. Modeling and simulation of oscillator-based random number generators. In *Circuits and Systems, IEEE International Symposium on*, volume 4, pages 324–327, May 1996.

[77] J.B. Plumstead. Inferring a sequence generated by a linear congruence. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 153–159. IEEE, 1982.

[78] M. Potdar, E. Chang, and V. Potdar. Applications of RFID in pharmaceutical industry. *Industrial Technology (ICIT), IEEE International Conference on*, pages 2860–2865, Dec. 2006.

[79] D.M. Pozar. *Microwave Engineering*. Wiley, second edition, 1998.

[80] UPM Raflatac. Dogbone inlay. [On-line], 2010. Available at http://www.upmrfid.com/rfid/.

[81] UPM Raflatac. Short dipole inlay. [On-line], 2010. Available at http://www.upmrfid.com/rfid/.

[82] K.M. Ramakrishnan and D.D. Deavours. Performance benchmarks for passive UHF RFID tags. In *Proceedings of the 13th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems*, pages 137–154, 2006.

[83] D. C. Ranasinghe. *Networked RFID Systems and Lightweight Cryptography, Chapter 18*, chapter Lightweight Cryptography for Low Cost RFID, pages 311–344. Springer, Nov. 2007.

[84] D. C. Ranasinghe and P. H. Cole. *Networked RFID Systems and Lightweight Cryptography, Chapter 3*, chapter Networked RFID Systems, pages 45–58. Springer, Nov. 2008.

[85] D. C. Ranasinghe and P. H. Cole. *Networked RFID Systems and Lightweight Cryptography, Chapter 8*, chapter An Evaluation Framework, pages 157–167. Springer, Nov. 2008.

[86] RFID Journal. Wal-Mart Opts for EPC Class 1 V2. Technical report. (last access Feb. 2010) Available at http://www.rfidjournal.com/article/articleprint/641/-1/1/.

[87] M.J.B. Robshaw. Stream ciphers. *RSA Labratories*, 25, 1995.

[88] P. Rosinger, B.M. Al-Hashimi, and N. Nicolici. Dual multiple-polynomial LFSR for low-power mixed-mode BIST. *Computers and Digital Techniques, IEEE Proceedings on*, 150(4):209–217, Jul. 2003.

[89] S.E. Sarma. Toward the 5 cents tag. Technical report, Auto-ID Lab, Nov. 2001. Withe Paper.

[90] B. Schneier. *Applied Cryptography*. John Wiley & Sons Inc., Hoboken, NJ, US, 1996.

[91] O. Semenov, A. Vassighi, M. Sachdev, A. Keshavarzi, and C.F. Hawkins. Effect of CMOS technology scaling on thermal management during burn-in. *IEEE Transactions on Semiconductor Manufacturing*, 16(4), 2003.

[92] C.E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 1948.

[93] J. Strüker, C. Wonnemann, M. Kähmer, and D. Gille. Managing the Deactivation Process of EPC Class-1 Generation-2 Tags in Retail Industry - University of Freiburg, Germany, 2007. (Last access Feb. 2010) Available at `http://www.telematik.uni-freiburg.de`.

[94] Linear Technologies. Ltspice iv. [On-line], 2010. Available at `http://www.linear.com/designtools/software/`.

[95] Alien Technology. Higgs 3 ic. [On-line], 2010. Available at `http://www.alientechnology.com/tags/rfid_ic.php`.

[96] Alien Technology. Squiggle inlay. [On-line], 2010. Available at `http://www.alientechnology.com/tags/index.php`.

[97] Trace Tecnologías. Trace inlay. [On-line], 2010. Available at `http://www.tracetecnologias.com/Imgs/solutions/-inlays_web.pdf`.

[98] Arizona State University. Predictie technology model. Technical report, International Organization for Standardization (ISO), [Online] Available at `http://ptm.asu.edu`, 2010.

[99] D. Wheeler and R. Needham. TEA, a tiny encryption algorithm. In B. Preneel, editor, *Fast Software Encryption*, volume 1008 of *Lecture Notes in Computer Science*, pages 363–366. Springer Berlin / Heidelberg, 1995.

[100] I. Zalbide, J. Vicario, and I. Velez. Power and energy optimization of the digital core of a Gen2 long range full passive RFID sensor tag. In *RFID, 2008 IEEE International Conference on*, pages 125 – 133, 2008.

[101] S.H. Zhou, W. Zhang, and N.J. Wu. An ultra-low power cmos random number generator. *Solid-State Electronics*, 2008.

Publications

Journal Papers

- J. Melià-Seguí, J. Garcia-Alfaro, J. Herrera-Joancomartí. *A Practical Implementation Attack on Weak Pseudorandom Number Generator Designs for EPC Gen2 Tags.* Journal of Wireless Personal Communications (WPC), Springer, ISSN: 0929-6212, DOI: 10.1007/s11277-010-0187-1, JCR, [Online] Nov. 2010.

Lecture Notes in Computer Science

- J. Melià-Seguí, J. Garcia-Alfaro, J. Herrera-Joancomartí. *Analysis and Improvement of a Pseudorandom Number Generator for EPC Gen2 Tags.* Workshop on Lightweight Cryptography for Resource-Constrained Devices (Co-located with Financial Cryptography and Data Security 2010 Conference) Tenerife - Spain, Lecture Notes in Computer Science, Volume 6054, Pages 34-46, Springer, DOI: 10.1007/978-3-642-14992-4_4, January, 2010.

Book Chapters

- J. Melià-Seguí, J. Garcia-Alfaro, J. Herrera-Joancomartí. *RFID EPC-Gen2 for postal applications: A security and privacy survey.* IEEE International Conference on RFID-Technology and Applications (RFID-TA) Guangzhou - China, Pages 118-123, DOI: 10.1109/RFID-TA.2010.5529872, June, 2010.

- J. Melià-Seguí, J. Garcia-Alfaro, J. Herrera-Joancomartí. *Clasificación de las Amenazas a la Seguridad en Sistemas RFID-EPC Gen2.* In XII Reunión Española sobre Criptología y Seguridad de la Información, Tarragona - Spain, September 2010.

- J. Melià-Seguí, J. Garcia-Alfaro, J. Herrera-Joancomartí. *Análisis de Seguridad y Privacidad para Sistemas EPC-RFID en el Sector Postal.* In XI Reunión Española sobre Criptología y Seguridad de la Información, Salamanca - Spain, September 2008.

Conference Papers

- J. Garcia-Alfaro, J. Herrera-Joancomartí, J. Melià-Seguí. *A Multiple - Polynomial LFSR based Pseudorandom Number Generator Design for EPC Gen2 Systems.* Mitacs Workshop on Network Security & Cryptography, Mitacs Focus Period, Toronto - Canada, June 2010.

- J. Melià-Seguí, J. Garcia-Alfaro, J. Herrera-Joancomartí. *EPC-RFID Security and Privacy for Postal Sector.* RFIDSec Asia, Taipei - Taiwan, January, 2009.