

DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

ENHANCING SECURITY IN RFID



by
Ömer AYDIN

July, 2019

İZMİR

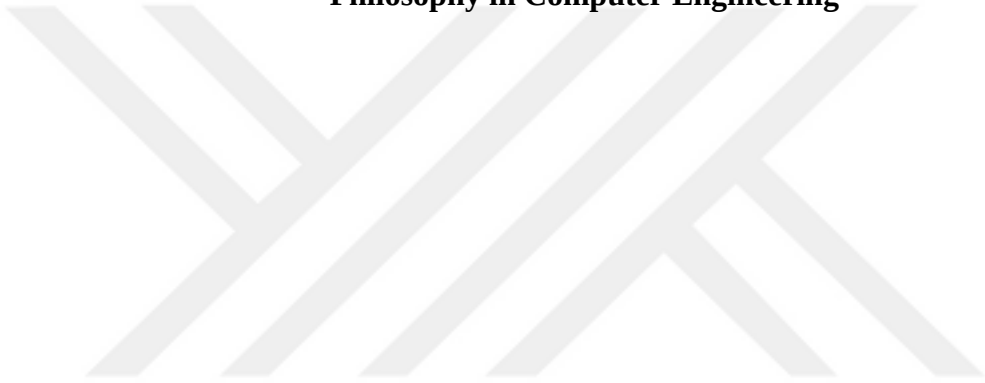
ENHANCING SECURITY IN RFID

A Thesis Submitted to

the Graduate School of Natural and Applied Sciences of Dokuz Eylül University

In Partial Fulfilment of the Requirements for the Degree of Doctor of

Philosophy in Computer Engineering



by

Ömer AYDIN

July, 2019

İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled “**ENHANCING SECURITY IN RFID**” completed by **ÖMER AYDIN** under supervision of **ASSOC. PROF. DR. GÖKHAN DALKILIÇ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.



Assoc. Prof. Dr. Gökhan DALKILIÇ

Supervisor



Prof. Dr. Yalçın ÇEBİ

Thesis Committee Member



Asst. Prof. Dr. Yavuz ŞENOL

Thesis Committee Member



Assoc. Prof. Dr. Yusuf Murat ERTEN

Examining Committee Member



Prof. Dr. Mehmet Ural Türk

Examining Committee Member



Prof. Dr. Kadriye ERTEKİN

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor Assoc. Prof. Dr. Gökhan DALKILIÇ, for his guidance, support and advice at every stage of this thesis. His valuable insights and experiences will guide me in all aspects of my future research.

I extend my thanks to the members of my thesis committee, Prof. Dr. Yalçın ÇEBİ, Asst. Prof. Dr. Yavuz ŞENOL and thesis examination committee members for their useful comments and suggestions.

I would also thank to the all served deans Prof. Dr. Öcal USTA, Prof. Dr. Pınar SÜRAL ÖZER, Prof. Dr. Kâmil TÜĞEN, Prof. Dr. Hüseyin Avni EGELİ and the employees of the Faculty of Economics and Administrative Sciences, for their support during my doctorate process.

In addition, I would like to thank the project support from The Scientific and Technological Research Council of Turkey. The number of the project was 215E225, code was Academic Research Funding Program Directorate (ARDEB) 3001 and name was “Pasif RFID Etiketlerinde İletişim Güvenliğinin Arttırılması”.

I thank to Dokuz Eylül University Computer Engineering Department Chair Prof. Dr. R. Alp KUT and the department’s employees for the support about planning, implementation of my thesis. In addition, I want to thank to Prof. Dr. Yeşim ZORAL, Asst. Prof. Dr. Ahmet ÖZKURT and Department of Electrical and Electronics Engineering for their valuable support to use the RFID reader in Faraday cage located in their labs.

I'm grateful to Turkish Information Technologies Authority, it's employees and Litum Information Systems Company and manager Alp ÜLKÜ for their supports to import FCC Compliant RFID readers to our country.

The production of the WISP passive RFID tag that is used in the thesis was developed at Intel Computer Network Research Laboratories and Washington University labs. I would like to thank Washington University for sending us two WISPs for free, Prof. Joshua SMITH, Dr. Alanson SAMPLE and Dr. Aaron PARKS.

I want to thank to my colleagues Res. Asst. Cem KÖSEMEN and Res. Asst. Umut Can ÇABUK for their support and contribution to my thesis. We published articles, conference papers, completed a TUBITAK project and applied for patents.

Science is a building that rises above previous studies so I would like to thank all the scientists that I have benefited from their studies, research and experiments in my thesis.

Finally, I thank my whole family especially my mother Aytül AYDIN, father Niyazi AYDIN, uncle Nihat AYDIN, wife Emel AYDIN and daughter Zeynep Berre AYDIN. All of your support and love throughout my life have brought me to this achievement.

Ömer AYDIN

ENHANCING SECURITY IN RFID

ABSTRACT

Radio frequency identifying (RFID) is a general term defining systems for transmitting an object or an individual's identification data via radio waves without the use of cables. Radio frequency identifying tags have constraints on hardware and computation due to structural reasons. These constraints occur as a serious risk to the security systems. Considering such problems and system resource constraints, it is of utmost importance to build a structure that is resistant to current security threats.

This thesis contains general information about the basic features of the RFID tags and wireless identification and sensing platform (WISP), which is a special passive RFID tag. According to this information, there were advised and proposed lightweight random number generators which is one of the most important issue for securing the communication between RFID tag and RFID reader. The randomness of the proposed random number generators has been tested by test suites such as ENT, Diehard, NIST and TestU01. With these random number generators which were successful in the tests, random numbers were generated which could be used in authentication protocol and encryption processes. In addition to all these, a new lightweight grouping proof authentication protocol (GPAPXL+) has been developed. GPAPXL+ uses a random number generator which was built in the scope of this thesis and named “xorshiftL+”. The security analysis of the proposed grouping proof authentication protocol has been tested theoretically and also with a security testing software. This new protocol has been proven to be successful based on the analysis and test results and it was implemented on the WISP RFID tag.

Keywords: Radio frequency identification, RFID, RFID attacks, wireless identification and sensing platform, WISP, WISP5, security, random number generator, pseudo-random number generator, true random number generator, RNG, PRNG, TRNG, genetic algorithm, GA, authentication protocol, grouping proof authentication protocol, cryptography, NIST, ENT, DieHard, TestU01, DES, AES, GPAPXL+

RADYO FREKANSI İLE TANIMLAMADA GÜVENLİĞİN ARTTIRILMASI

ÖZ

Radyo frekansı ile tanımlama (RFID), radyo dalgaları vasıtasıyla kablo kullanmadan bir birey veya nesneye ait tanım verisini iletmeyi sağlayan sistemleri tanımlayan genel terime denir. Yapısal nedenlerden dolayı Radyo frekansı ile tanımlama etiketlerinde donanım ve hesaplama kısıtları mevcuttur. Bu kısıtlar güvenlik sistemlerinde ciddi bir risk oluşturmaktadır. Sistem kaynakları konusundaki kısıtlar ile bahsedilen sorunlar dikkate alındığında güncel güvenlik tehditlerine karşı koyabilecek sistemler inşa etmek önem kazanmıştır.

Bu tez, RFID etiketleri ile özel bir pasif RFID etiketi olan kablosuz tanımlama ve algılama platformunun (WISP) temel özellikleri hakkında genel bilgiler içermektedir. Bu bilgiler ışığında RFID etiketi ve RFID okuyucu arasındaki iletişim güvenliğini sağlamak için gerekli en önemli konulardan biri olan düşük kaynak tüketen rastgele sayı üreticileri geliştirilmiştir. Üretilen rastgele sayı üreticilerinin rastsallığı ENT, Diehard, NIST ve TestU01 gibi test araçları ile sınanmıştır. Testlerde başarılı olan bu rastgele sayı üreticileri ile doğrulama protokolü ve şifreleme işlemlerinde kullanılabilecek güvenli rastgele sayılar üretilmiştir. Tüm bunlara ek olarak düşük kaynak kullanımına sahip yeni bir grup doğrulama protokolü (GPAPXL+) geliştirilmiştir. GPAPXL+ bu tez kapsamında geliştirilen ve “xorshiftL+” şeklinde adlandırılan bir rastgele sayı üreticini kullanmaktadır. Önerilen grup kimlik doğrulama protokolünün güvenlik analizi teorik olarak ve ayrıca bir güvenlik test yazılımı ile test edilmiştir. Bu yeni protokolün analiz ve test sonuçlarına göre başarılı olduğu kanıtlanmış ve protokol, WISP RFID etiketi üzerinde uygulanmıştır.

Anahtar Kelimeler: Radyo frekansı ile tanımlama, RFID, RFID saldırıları, kablosuz tanımlama ve algılama platformu, WISP, WISP5, güvenlik, rastgele sayı üretici, sözde rastgele sayı üretici, gerçek rastgele sayı üretici, RSÜ, SRSÜ, GRSÜ, genetik algoritma, GA, doğrulama protokolü, gruplama kanıtı doğrulama protokolü, şifreleme, NIST, ENT, DieHard, TestU01, DES, AES, GPAPXL+

CONTENTS

	Page
Ph.D. THESIS EXAMINATION RESULT FORM	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT.....	v
ÖZ	vi
LIST OF FIGURES	xiii
LIST OF TABLES	xv
 CHAPTER ONE - INTRODUCTION	 1
1.1 Overview	1
1.2 Motivation	5
1.3 Aim of Thesis	5
1.4 Contribution.....	6
1.5 Thesis Organization.....	7
 CHAPTER TWO - PRELIMINARIES	 9
2.1 History of Cryptology	9
2.2 Data Encryption Standard	13
2.3 Advanced Encryption Standard.....	13
2.4 ENT Test Tool.....	14
2.5 Diehard Test Suite	14
2.6 NIST Test Suite	14

2.7 TestU01 Test Suite	15
2.8 Radio Frequency Identification	15
2.8.1 The Basics of RFID Technology	16
2.8.2 History of the RFID	18
2.8.3 RFID Tag	19
2.8.3.1 RFID Tag Types by Energy Harvesting Method	22
2.8.3.2 RFID Tag Types by Memory Access Types	23
2.8.3.3 RFID Tag Types by Frequencies	24
2.8.4 RFID Antennas	26
2.8.5 RFID Readers	27
2.8.6 Access Controllers	27
2.8.7 Middleware Software	28
2.8.8 Reader Interface	28
2.8.9 EPC and EPCglobal	28
2.9 Wireless Identification and Sensing Platform	30
2.10 Random Number Generators	32
2.10.1 Area of Use and Importance of Random Numbers	33
2.10.2 True Random Number Generators	34
2.10.3 Pseudorandom Number Generators	34
2.10.4 Hybrid/Mixed Random Number Generators	34
2.11 Genetic Algorithm	35
2.12 Authentication Protocols	35

CHAPTER THREE - LITERATURE REVIEW..... 36

3.1 Random Number Generation Methods.....	36
3.2 Authentication Methods	47
3.3 Other Security and Privacy Enhancing Methods in RFID	53

**CHAPTER FOUR - PROPOSED SECURITY ENHANCING METHODS FOR
RFID 56**

4.1 Proposed True Random Number Generators	56
4.1.1 True Random Number Generation by Using Analog to Digital Converter	56
4.1.2 True Random Number Generation by Using Temperature Sensor	57
4.1.3 True Random Number Generation by Using Difference Between Clock Pulses.....	57
4.1.4 True Random Number Generation by Using Accelerometer	58
4.2 Merged Linear Feedback Shift Register.....	60
4.3 Modified Xorshift+ Algorithm: XorshiftR+	62
4.4 Lightweight Version of the Xorshift Algorithm: XorshiftL+	65
4.5 Genetic Algorithm and Random Number Generation.....	66
4.6 Grouping Proof Authentication Protocol Using XorshiftL+	70
4.6.1 Registration Phase	71
4.6.2 Authentication Phase	72
4.6.3 Verification Phase.....	74

CHAPTER FIVE - TESTS AND EXPERIMENTAL RESULTS 75

5.1 Proposed Random Number Generators	75
5.1.1 ENT and Diehard Test Results of the Random Numbers Generated from Four Different Sources	75
5.1.2 Test Results for Random Number Generation by Using Temperature Sensor	76
5.1.3 Test Results for Random Number Generation by Using Analog to Digital Converter	77
5.1.4 Test Results for Random Number Generation by Using Difference Between Clock Pulses	78
5.1.5 Test Results for Random Number Generation by Using Accelerometer	78
5.1.6 Test Results for Merged Linear Feedback Shift Register.....	79
5.1.7 Test Results for XorshiftR+	81
5.1.7.1 Empirical Results	83
5.1.7.2 Testbeds	84
5.1.7.3 Xorshift+	84
5.1.7.4 Xorshift*	84
5.1.7.5 Reduced Xorshift+ var1 ($s[1] = x$, return $x + y$)	85
5.1.7.6 Reduced Xorshift+ var2 ($s[1] = x + y$, return $x + y$).....	85
5.1.7.7 Reduced Xorshift+ var3 ($s[1] = x + y$, return x).....	85
5.1.7.8 Simple Linear Congruential Generator	87
5.1.7.9 Linear Feedback Shift Register: LFSR113	88
5.1.7.10 Xoroshiro+	89
5.1.7.11 Further Notes	89
5.1.8 Test Results for XorshiftL+	90

5.1.9 Test Results for Random Number Generation Using Genetic Algorithm	93
5.2 Results and Analysis for the Grouping Proof Authentication Protocol Using XorshiftL+	94
5.2.1 Traceability Attack Analysis	95
5.2.2 Replay Attack Analysis	95
5.2.3 Physical Disclosure Attack Analysis	95
5.2.4 Impersonation Attack Analysis.....	96
5.2.5 Remove Unwanted Tags Analysis.....	96
5.2.6 Partial-Proof Analysis.....	96
5.2.7 Desynchronization Attack Analysis	97
5.2.8 Denial of Service Attack Analysis	97
5.2.9 Man in The Middle Attack Analysis	97
5.2.10 Cloning Attack Analysis.....	97
5.2.11 Eavesdropping Analysis	97
5.2.12 Security Analysis with Scyther.....	98
5.2.13 Security Service Analysis	99
5.2.14 Examination and Comparisons of Authentication Protocols.....	100
 CHAPTER SIX - CONCLUSION	105
 REFERENCES.....	109
 APPENDICES	141
 APPENDIX 1: C code to sample voltage via ADC on WISP.....	141

APPENDIX 2: C code to generate random number sampling from temperature sensor on WISP	142
APPENDIX 3: C code to generate true random number by using difference between clock pulses	143
APPENDIX 4: C code to generate true random number by using accelerometer (1 st method).....	144
APPENDIX 5: C code to generate true random number by using accelerometer (2 nd method).....	146
APPENDIX 6: Scyther code for testing our proposed authentication protocol ..	148



LIST OF FIGURES

	Page
Figure 2.1 A block diagram of a typical RFID system	16
Figure 2.2 Working fields of Auto-ID group	17
Figure 2.3 A tags microstructure.....	20
Figure 2.4 Tag construction	20
Figure 2.5 General architecture of a passive RFID tag.....	21
Figure 2.6 Tag examples	21
Figure 2.7 RFID tag types.....	22
Figure 2.8 RFID antennas	27
Figure 2.9 RFID readers.....	27
Figure 2.10 Context of a 96-bit EPC.....	29
Figure 2.11 ADXL362 accelerometer sensor.....	30
Figure 2.12 AD22100 temperature sensor	31
Figure 2.13 WISP system block diagram.....	31
Figure 2.14 MSP430 clock system.....	32
Figure 3.1 Development lapse of (some) xorshift based RNGs	41
Figure 3.2 A typical UHF RFID tag reading scenario	50
Figure 4.1 Flow chart for true random number generation by using temperature sensor, voltage sampling and difference between clock pulses.....	57
Figure 4.2 Flow chart for true random number generation by using accelerometer (1 st method).....	58
Figure 4.3 Flow chart for true random number generation by using accelerometer (2 nd method).....	60
Figure 4.4 LFSR general structure	60
Figure 4.5 MLFSR input and outputs	61

Figure 4.6 MLFSR flow chart.....	61
Figure 4.7 Merge function C code for LFSRs.....	62
Figure 4.8 True random number generator flow chart.....	65
Figure 4.9 HRNG C code implementation.....	65
Figure 4.10 Flow diagram of genetic algorithm	67
Figure 4.11 C code for generated PRNG by GA	69
Figure 4.12 RFID system components with group authentication protocol applied..	70
Figure 4.13 Group authentication protocol steps	71
Figure 5.1 Bitmap of xorshiftR+; 1000 points plotted using scatter in TestU01.....	88
Figure 5.2 Bitmap of simple LCG; 1000 points plotted using scatter in TestU01.....	90
Figure 5.3 Scyther security analysis tool results	98

LIST OF TABLES

	Page
Table 2.1 The general characteristics of the tags according to their energy acquiring method	23
Table 3.1 Some popular and known Pseudorandom Number Generators	37
Table 4.1 Notations	70
Table 5.1 ENT test results of the true random numbers generated from four different sources	76
Table 5.2 Diehard test results of the true random numbers generated from four different sources	76
Table 5.3 NIST test results of the true random numbers generated using temperature sensor	77
Table 5.4 True random number generator which generates numbers by sampling voltage on analog to digital converter	77
Table 5.5 NIST test results for true random number generators by using difference between DCO & VLO clock pulses	78
Table 5.6 NIST test results for true random number generator using accelerometer (1 st method).....	79
Table 5.7 NIST test results for true random number generator using accelerometer (2 nd method).....	79
Table 5.8 NIST test result for 16-bit and 32-bit random numbers generated using C rand() function	80
Table 5.9 NIST test result for 32-bit random numbers generated by merged linear feedback shift register.....	80
Table 5.10 Merged linear feedback shift register working times on WISP	81
Table 5.11 Comparative data of empirical test results from Big Crush for all subject RNGs	83
Table 5.12 NIST STS results for reduced xorshift+ var3	86

Table 5.13 ENT test result for original xorshift+.....	87
Table 5.14 ENT test result for xorshiftR+	87
Table 5.15 NIST STS results for xorshiftL+.....	91
Table 5.16 Time comparison with the known pseudorandom number generator algorithms	92
Table 5.17 ENT test results for the proposed hybrid random number generator.....	92
Table 5.18 NIST test results for the numbers generated by GA (1 st Method)	93
Table 5.19 NIST test results for the output of the PRNG generated by GA (2 nd Method)	94
Table 5.20 Examined protocols' reference numbers and names.....	101
Table 5.21 Techniques, verifications and classes of authentication protocols	102
Table 5.22 Comparison of APs in terms of security threats and security services ..	104

CHAPTER ONE

INTRODUCTION

In Section 1.1, a quick overview of the radio frequency identification (RFID) tags, electronic product code (EPC), wireless identification and sensing platform (WISP), random number generators (RNG) and authentication protocols (AP) have been given. In Section 1.2, it is talked about the motivation reasons of the thesis. In Section 1.3 the aim of the thesis is explained. In Section 1.4, contributions have been listed. Finally, Section 1.5 was given the thesis organization.

1.1 Overview

In recent years, emerging technologies, such as smart cards, radio frequency identification tags (RFID), wireless sensor nodes and the concept of Internet of things (IoT) brought not only new solutions but also challenges in their scope of application. The proliferation of devices manipulating or transmitting sensitive and critical information require more attention to security issues, because classical security algorithms cannot offer effective and feasible security solutions for these groups of devices. Thus, many lightweight cryptographic algorithms have been suggested in literature, including block ciphers titled “The LED block cipher” (Guo, Peyrin, Poschmann & Robshaw, 2011), “PRINTcipher: A Block Cipher for IC-Printing” (Knudsen, Leander, Poschmann & Robshaw, 2010), “KLEIN: a new family of lightweight block ciphers” (Gong, Nikova & Law, 2011), “KATAN and KTANTAN—a family of small and efficient hardware-oriented block ciphers” (De Canniere, Dunkelman & Knežević, 2009), “LBlock: a lightweight block cipher” (Wu & Zhang, 2011), “Piccolo: an ultra-lightweight blockcipher” (Shibutani, Isobe, Hiwatari, Mitsuda, Akishita & Shirai, 2011), “The SIMON and SPECK families of lightweight block ciphers”(Beaulieu, Shors, Smith, Treatman-Clark, Weeks & Wingers, 2013), and hash functions hash functions titled “Quark: A lightweight hash”(Aumasson, Henzen, Meier & Naya-Plasencia, 2013), “The PHOTON family of lightweight hash functions” (Guo, Peyrin & Poschmann, 2011), “SPONGENT: A lightweight hash function” (Bogdanov, Knežević, Leander, Toz, Varıcı &

Verbauwhede, 2011). The aim of lightweight security algorithms is to find a balanced solution for performance, speed and security needs taking into account limitations such as storage, and processing power.

EPC tags (GS1, 2019) which is called RFID become more common nowadays. RFID is a new technology which is planned to replace classical barcode technology. The reason for this replacement is the advantages of RFID over barcodes. The advantages of RFID over barcodes are wireless data transmission, communication type and ability of computing operations. Barcodes can be read by optical method in which devices communicate from a distance using light to carry information. The disadvantages of this communication are the limitation about distance and direct light vision requirement. RFID Tags are read/write, durable and reusable, capable of storing more data, can be printed with a barcode, don't need line of sight and also transmitted data can be encrypted. These major benefits enable applications in a very large area such as library management, supply chain management, smart home appliances, anti-counterfeiting banknotes and etc. RFID technology has several privacy and security risks over its advantages. The security and privacy problems come from the RFID tag's nature because when it is read using a query that is sent by the reader, tag wirelessly releases a unique id (EPC) which is a static bit string. Everybody who has compatible RFID reader can track this communication and get all information. To solve this problem, encryption algorithms and authentication protocols are used. Random number generation is the one of the most important requirement for both authentication protocols and encryption. In some RFID tags, there is built-in random number generators. Especially for passive RFID tags, built-in random number generators do not provide standards for randomness due to the low processing capabilities.

Random number generation has an important role in cryptography and authentication, and therefore, in security. These numbers can be used as the key or the seed for key generation in cryptographic algorithms and nonce for authentication protocols. Because of this reason, it is so important to create a new, secure, and lightweight random number generator for the RFID tags. For example, random numbers can be an input as the secret key for symmetric encryption algorithms such as data encryption standard (DES) (Federal Information Processing Standard [FIPS],

1999) and advanced encryption standard (AES) (FIPS, 2001). Encryption algorithms and authentication protocols need number sequences generated with high randomness otherwise, attackers can easily infiltrate to the system. There should be a method to generate these number sequences, which is called random number generator. Random number generators can be separated in two groups according to their sources. True random number generators (TRNGs) use hardware sources. On the other hand, deterministic or in other words pseudorandom number generators (PRNGs) which use algorithms for generating random numbers or bit sequences. PRNGs mostly create random numbers quickly; therefore, they are suitable for stream ciphers. Moreover, well-designed PRNG algorithms are easy to implement in lightweight devices. However, the generated number sequence can be predicted if the algorithm is known (Fischer & Bernard, 2011). For this reason, this type of generators must be computationally secure and seeded by an unpredictable source. These needs make it necessary to use the two type of random number generator in a mixed way. TRNG is used as the initial seed of PRNG. This type of random number generators can be called as hybrid random number generators (HRNGs). HRNGs make a combination of a PRNG (fast generation and high quality random numbers) seeded repeatedly by a TRNG (high unpredictability but slow generation). HRNG must be designed to overcome the challenge of the balance between speed and predictability (Fischer & Bernard, 2011).

Communication between the RFID reader and the tag occurs wirelessly. Therefore, this communication can be corrupted or accessed by the third parties. It is very important that the communication between RFID and the reader is established in a secure structure. To establish a secure communication between two entities, first they have to authenticate the identities of each other. So the second important issue to build a secure RFID system is an authentication protocol. This protocols are a type of computer communication model. It is designed to prove the identity by transferring required information between the entities. This is the most important protection layer for secure communication within computer and IoT networks. The easy of identity theft in the virtual world has led to a search for specific identity authentication methods. As the amount of valuable data transmitted over the network increases day by day, it is getting more important to manage the authentication of the users. Since it

is easy to steal someone's identity in the computer world, it has been tried to find specific authentication methods in order to determine whether the person who requested the information has identified himself or herself to the other party.

In RFID systems, the grouping proof authentication schemes provides a simultaneous solution that allows for the authentication of multiple tagged objects. In the real world, the grouping proof based authentication scheme has become important because tags are used on an increasingly wide range of objects. Group authentication protocols are used in many applications such as hospital inpatient system applications, retail sales applications and Internet of Things. (Atzori, Iera & Morabito, 2010; de Fuentes, Peris-Lopez, Tapiador & Pastrana, 2015; Kumar, Gopal & Aggarwal, 2014). The cost and scalability of the tag are the two main topics can be analysed for a grouping proof based authentication. Too many various and types of tags are used in the grouping proof authentication schemes and protocols. The overall cost of the system depends on the cost of each tag. Considering the overall cost, the system's designers prefer to use low-cost tags. Low-cost tags have low or limited hardware sources and processing power. This type of tags has the maximum of 3000 gate that can be reserved for security systems (Sundaresan, Doss, Piramuthu & Zhou, 2014). Using lightweight tags, it is difficult to establish sophisticated encryption and security systems which need too much resources. Design of a secure protocol on lightweight and low-cost tags is essential. Therefore, a balance between cost and security should be established. However, scalability is another important issue in that systems. This issue is based on the responses dependency, protocol's execution speed and run-time. Scalability should not depend on the current tag population or number of tags participating the system in which the authentication protocol is running.

Considering the information mentioned above, two main topics were considered in this thesis. First topic is developing secure, stable and lightweight random number generator(s) for the RFID tags and other IoT devices. Second topic is designing and implementing a new scalable, lightweight grouping proof based authentication scheme which is secure against known attacks in the literature.

1.2 Motivation

Today, RFID tags which are currently being used in many areas, come across with many various types. Passive tags have no power supplies; they harvest their energy from the magnetic field that is emitted by the RFID reader. Harvested energy used to power its sensors and communication circuits up. RFID reader can send and receive data. Additionally, the RFID can send and receive data. This mutual communication should not be vulnerable to external threats. Similarly, this data must not be processed after accessed by the third party. For these reasons, some specific security infrastructures such as random number generators are located in that devices. Because of the hardware and computation constraints of the RFID tags, mostly the generated random numbers are not statistically and cryptographically secure. This causes security risks and bugs in the authentication protocols and encryption of the transmitted data. On the other hand, so many solutions were suggested to build a secure authentication, however, an authentication protocol accepted by all could not be proposed. This is because many of these authentication protocols have security problems or don't run under limited hardware and computation abilities. These are all our motivation topics and also this thesis addresses the following research questions:

- How to construct a random number generator using WISP passive RFID's different hardware sources and can they pass the known statistical randomness test?
- How to construct a pseudorandom/hybrid number generators that passes randomness tests of the well-known test suites to enhance security for lightweight devices such as RFID tags?
- How to construct a lightweight, scalable and secure grouping proof authentication scheme for RFID tags and other lightweight devices?

1.3 Aim of Thesis

The aim of this thesis is enhancing security in RFID systems. There are many things to enhance the security challenges in lightweight devices such as RFID tags. As evidenced by the number of articles published in recent years, enhancing security in lightweight devices is a popular and important issue. Many articles are mostly focused on attacks. First of all, it is necessary to determine the basics of security risks. RFID

systems has so many limitations such as energy harvesting, low capacity hardwares and etc. so the wireless communication between the parties has security risks. Privacy and tracking are two of these risks. There so many other security risks for wireless communication, applications and structural design in RFID systems. Random numbers are so important for wireless communication in that systems. These numbers can be used for nonce value or encryption key in communication stages. In both of these usage, used number series have to be random so the adversary can't guess or calculate the next number in that series. On the other hand, there should be an authentication protocol to authenticate the tags' identity when they try to join the group and send or receive data. Most of the authentication protocols are unsuccessful to overcome privacy and security challenges. It is necessary to develop a new grouping prrof authentication protocol because RFID tags are used as a group and so many ecurity and privacy risks exist. On the other hand, so many proposed protocols are theoretically or simulated, there are not so much protocols applied on real hardware.

1.4 Contribution

The main contributions of this thesis can be described in three main points;

Firstly, the newest version WISP passive RFID tag that is WISP 5.0 was obtained from the Washington University research laboratory. The firmware was modified and four TRNGs were created. These four true random generators use different hardwares, sensors or hardware specifications to create random numbers. First TRNG uses WISP's built-in accelerometer. Second one is uses built-in temperature sensor. Third one is uses Analogue Digital Converter's (ADC) output to measure capacitor's energy level. Last one uses the frequency difference of the very low power low frequency oscillator (VLO), digitally controlled oscillator (DCO) which are two built-in clocks of the WISP. The randomness of the number series generated by these TRNGs were tested by using test suites. That TRNGs are used in some of our publications and a project. The project is supported by TUBITAK with the project number 215E225. The publication titled "A hybrid random number generator for lightweight cryptosystems: xorshiftLplus" was presented in the 3rd International Conference on Engineering Technology and Applied Sciences (ICETAS) in 2018.

Secondly, different PRNGs were created by using different methods. For example, Linear Feedback Shift Register structure, Marsaigle's xorshift structure and genetic programming methods were used. One HRNG which uses a PRNG initialised by one of the TRNG was created. Moreover, genetic algorithm was used to create PRNGs. All these PRNGs are lightweight and proposed and tested on WISP passive RFID tag. All number series created by these PRNGs were tested by the randomness test suites. These PRNGs were also published in conference papers and a journal indexed in Science Citation Index Expanded (SCI-Exp). "Pseudo Random Number Generator for Internet of Things: Merged Linear Feedback Shift Register" titled conference paper was presented in 3rd International Management Information Systems Conference, 2016 (Aydın & Dalkılıç, 2016). "The Pseudorandom Number Generator Generation Method with Genetic Programming for Lightweight Devices" titled paper was published in International Conference on Computer Science and Engineering, 2018 (Kösemen, Aydın & Dalkılıç, 2018). "Genetic programming based pseudorandom number generator for wireless identification and sensing platform" titled article was published in Turkish Journal of Electrical Engineering & Computer Sciences, 2018 (Kösemen, Dalkılıç & Aydın, 2018).

Finally, a new scalable, lightweight and secure grouping proof authentication protocol was implemented. This protocol is proposed for WISP passive RFID tag and tested structurally, theoretically against the well-known attacks. Moreover, Syther tool was used to analyze the protocol and tested for the attacks. This protocol was implemented in the project supported by TUBITAK with the project number 215E225.

1.5 Thesis Organization

This thesis is composed of six main chapters. The introduction is given in the first chapter. In this chapter, the motivation, aim of the thesis and contribution are discussed.

The chapters of this thesis are explained briefly below;

In chapter two, it is tried to refresh the memories by giving basic definitions and detailed information about the DES, AES, ENT, Diehard, National Institute of

Standards and Technology (NIST) test suite, TestU01, RFID, WISP, random number generators, genetic algorithm (GA) and authentication protocols.

In the third chapter related works within the thesis's research area are given. Especially, literature about enhancing security technics and methods is tried to be reviewed.

In chapter four, proposed random number generators and the authentication protocol are defined and summarized.

In chapter five, test methods to check randomness or security and privacy and the results are given.

At the end, in the sixth chapter, the conclusions of the thesis are presented. Then, the future works are discussed.

CHAPTER TWO

PRELIMINARIES

As it is given in our motivation topics in the previous section, there is a need for scalable security systems for lightweight hardwares such as RFID and IoT devices. Usage of these hardwares at many critical areas such as health, stock, logistics etc. has been increasing day by day. Security and privacy have become so important. It is important to know some topics to build and understand secure system architectures for RFID and other lightweight devices. In this chapter, it is tried to briefly summarize some important topics used in the thesis to refresh the memory of the reader.

2.1 History of Cryptology

One of the most important issues of the security is the science of password and encryption. Password and encryption are used everywhere and every date that requires security in our life. Cryptology has been defined as the science of password by society. Its history is one of the oldest technical works dating back 4000 years (Cryptography timeline, 2004).

Cryptology has evolved over the years as follows:

- Egyptians used non-standard hieroglyphs which were the first written cryptographic documents around 1900 BC.
- In Mesopotamia, a modern approach to cryptography was approached in 1500 BC. The first use of cryptography was in that year and it was encrypting a secret formula.
- In the Hebrew scripts in Jeremiah's book, the simple substitution code, also known as Atbash, was used in reverse alphabets. Atbash is one of the Hebrew encodings used in 590 BC.

- The Greek Spartans found the first form of military cryptography in 487 BC. The soldiers used a wooden stick called Skytale, which could be wrapped around a strip of leather or parchment.
- Artha-sastra is a book written by Kautilya in 300 BC. This book recommends various cryptanalysis methods, which mean breaking the code that provides access to intelligence information.
- Julia Ceasar used the encryption method, which is known as “Caesar Encryption” between 100 and 44 BC. Each letter in the text was shifted by 3 letters from its alphabet, in this method.
- Abu Abd al-Rahman, who was the creator of the first Arabic dictionary, has written a book on how he solved an encrypted text written in Greek for the Byzantine emperor between 725 and 790.
- Some encrypted texts have been found in the documents surviving today from the Ghaznavids between 1000 and 1200.
- A book about encryption was written by Blaise de Vigenere in 1586. In this book, for the first time, automatic encryption keying was mentioned for cipher and encrypted text. DES, cipher block chaining (CBC) and the cipher feedback (CFB) modes still use this method.
- Thomas Jefferson, a mathematician at the University of Pennsylvania, invented the Wheel Cipher in 1790. This led to the development of the M-138-A Strip Cipher machine. In the 2nd World War, the US Navy used the M-138-A Strip Cipher.
- Charles Wheatstone found the playfair password named Lyon Playfair who is one of his friend in 1854.
- Friedrich W. Kasiski publishes a book that gives the first general solution of the multi-alphabet code that uses repetitive letters in 1861. Thus, the multi-alphabet password, which has been accepted as strong for many years, is now considered weak.

- A strong military cryptography was used in 1914 during the World War I. During the communication, the British made their messages using cryptographic tools. The French used 4-digit numbers that expressed the words.
- Joseph Mauborgne and Gilbert Vernam found the perfect cryptographic system named “one-time pad” in 1919.
- William Frederick Friedman founded Riverbank Laboratories, made cryptanalysis for the US between 1920 and 1930. He solved the Japanese Purple Cipher Machine encryption system in World War II.
- Arthur Scherbius invented the Enigma machine in 1923. The German government took over and developed this machine and used it to encrypt military communications in World War II. Alan Turing and his team unravel the mystery of this machine.
- The Japanese invented the Purple cipher machine to encrypt the messages in 1933. This machine uses rotors to displace letters in the alphabet.
- Officially the National Security Organization (NSA) was established in the United States in 1952.
- Lucifer algorithm was developed by Horst Feistel (IBM) in 1970. It forms the basis of DES.
- Stephen Wiesner and Gilles Brassard started the history of the quantum cryptography with their work at Columbia University. They started to work on quantum cryptography in the early 1970s (Brassard & Crépeau, 1996).
- A special method used to exchange cryptographic keys, was first published in 1976 by Whitfield Diffie and Martin Hellman in their article "New Directions in Cryptography".
- DES was announced by the US as federal information processing standard (FIPS) 46 in 1976.

- Ronald L. Rivest, Adi Shamir and Leonard M. Adleman worked on a new algorithm and found the RSA algorithm in 1978. RSA was generated by using the initial letters of their surnames.
- Victor S. Miller and Neal Koblitz described a cryptographic system called elliptic curve cryptography (ECC) in their separate studies in 1985.
- Xuejia Lai and James Massey: They found the International Data Encryption algorithm (IDEA) in 1990.
- The pretty good privacy (PGP) system was developed and released by Phil Zimmerman in 1991.
- NIST published secure hash algorithm (SHA-1) as a standard in 1995.
- NIST opened a competition for a symmetric algorithm to replace DES in 1997.
- Rijndael algorithm designed by Nijar's Joan Daemen and Vincent Rijmen was standardized as AES in 2001. (Cryptography timeline, 2004).
- NIST has released its secure hash algorithm 3 (SHA-3). It became standard in 2015 (Fips 202, 2015).
- The NSA announced to replace ECC by moving to a quantum-resistant algorithms suite, in 2015 (Wohlwend, 2016).

This historical development proves that encryption is used in a large part of the known period of human history. Important information is encrypted using very different types of cryptography to prevent from being seen or stolen by others. Here, 3 basic points stand out. It is expected that the encrypted text will not be understood by others. Second, an encrypted text should be able to be decrypted properly. Thirdly, this situation should be understood if there is a distortion in the encrypted text.

There are two types of password science, the names of which are referred to as Cryptography and Cryptanalysis. When the word analysis of the cryptology in the is

made, it is seen that the word “crypto” means “secret” and “graphy” means “text”. These words are used in ancient Greek and cryptology is the result of the merger of these two words. The internet is a factor that increases the field of cryptology science. Internet has emerged with security deficits.

In the following sections, we will briefly talk about some encryption algorithms, randomness test suites that are currently in use.

2.2 Data Encryption Standard

At the end of the 1960s, a group that is directed by "Horst Feistel" a researcher at IBM, developed a cryptographic system called LUCIFER. In 1973, the US standards institute National Institute of Standards and Technology (NIST) invited firms to establish a standard for civil use. As a result of the investigations, the closest solution was found to be LUCIFER. US National Security Agency (NSA) experts made some arrangements on LUCIFER which has a 128-bit encryption key so they reduced the key length to 56 bits. This new algorithm was published as DES in 1977 and was put into use as standard in many areas, especially in the finance industry (Standard, 1977; Diffie & Hellman, 1979).

The DES algorithm is a symmetric encryption algorithm that uses secret key management. Both encryption and decryption process uses the same key. DES uses the Feistel structure for encryption, such as the many symmetric encryption algorithms do today. By the year 2000, DES had been largely frangible. For this reason, TripleDES was recommended. TripleDES is a cryptographic technique created by the successive execution of the Standard DES with two or three keys of 112 or 168 bits.

2.3 Advanced Encryption Standard

AES (Rijndael) algorithm encrypts 128-bit data blocks by using 128, 192 or 256-bit keys. It is also a type of Substitution–Permutation Network (SPN) algorithm (Heys & Tavares, 1996; Heys, 2002; Keliher, 2003; Stinson, 2005). Number of rounds is changing according to the key length in AES. If the 128-bit key is used, encryption is done in 10 rounds. 12 and 14 rounds are required for 192-bit and 256-bit keys, respectively. In AES algorithm each round consists four layers. First of all, 128-bit

data transform to 4x4 byte matrix. Then, in every cycle, bytes are replaced, columns are shifted and exclusive-or (XOR) operations are performed with the key determined for that loop from the key planning. Within the process, transforms the inputs to a new value as an output each state into a new value using an S-box table. The columns are shifted in the 4x4 byte matrix, and the values in that column are mixed for any column. In the last layer of the loop, XOR is done with the key belonging to that round.

2.4 ENT Test Tool

ENT is a program that performs various tests on the byte series stored in a file and shows the results of these tests (Walker, 2008). This program is a useful tool for evaluating cryptographic and statistical sampling applications, compression algorithms and pseudorandom number generators. ENT test suite includes entropy, arithmetic mean, chi-square, serial correlation coefficient and monte carlo value tests.

2.5 Diehard Test Suite

Diehard (Brown, Eddelbuettel & Bauer, 2013) is one of the test suite that test the random number generators. Diehard includes birthday spacings, the squeeze, monkey tests, count the 1s, minimum distance, parking lot test, ranks of matrices, random spheres, overlapping permutations, runs, overlapping sums and craps tests.

2.6 NIST Test Suite

In the basic article (Bassham, Rukhin, Soto, Nechvatal, Smid, Barker, ... & Dray, 2010), in which NIST is explained, some methods are described to select and test the pseudorandom number generators, statistically. The output of that generators are used in so many cryptographic operations such as generating encryption and decryption keys. These test results can be useful to get an idea about the suitability of the generators for the cryptographic applications. The statistical randomness tests are listed as follows.

NIST statistical test suite includes frequency, block frequency, non-overlapping template matchings, random excursions, longest runs of ones, rank, Spectral, cumulative sums, approximate entropy, overlapping template matchings, runs,

universal statistical, random excursions variant, linear complexity and serial tests. The results of these test is evaluated by their probability values. For example, if the result of the Frequency test is less than 0.01 so this means that the test is failed otherwise it is successful. Moreover, if the calculated p-value of a sequence is greater than 0.0001 so it can be called to be uniformly distributed (Bassham et al., 2010).

2.7 TestU01 Test Suite

TestU01 is a software library coded with American National Standards Institute (ANSI) C language and contains statistical tests for uniform random number generators. There are many predefined random number generators in generic form within this library. Statistical tests can be applied to related generators. In addition, user-defined random number generators can be tested by the statistical tests included in the package. There are specific tests for bit sequences or sequences of uniform random numbers in $[0,1]$. In the package, there are tools to draw the vectors of the generated points (L'Ecuyer & Simard, 2007).

2.8 Radio Frequency Identification

Wireless technologies have a big role in our daily lives. Most basically technologies like Wireless Fidelity (Wi-Fi), Bluetooth (Xydis & Blake-Wilson, 2002), Near Field Communication (NFC), IrDA and ZigBee (Zhou, Jiang, Wang & Zhang, 2014) are used in many fields of wireless communication. As a result of the increasing usage of the smartphones day by day, these new technologies reaches to so many people. File transfers, video chats, device sharing and many more applications are possible with these technologies. These advancements in wireless systems found a big place in military technologies (Bowden, 1985), applications requiring information confidentiality (Şahin, 2010) and production activities. Today, RFID systems have become a preferred system in both production areas and supply chain processes because of the important criterias such as time constraints and financial costs in that areas. In comparison with barcode technology RFID is a fairly new technology that can be used in many fields. RFID technologies are improvable, flexible, versatile, trustworthy and cheap.

2.8.1 The Basics of RFID Technology

RFID is called radio frequency identifying. It means that components used by RFID are communicating with radio frequencies. To summarize RFID, it is an automatic monitoring and identification system that uses radio frequencies to provide data exchange. It has a microchip containing information about the object, an antenna and an integrated tag to an object. There are 3 components that has a role in this information exchange. Reader, tag, antenna. The RFID reader emits electromagnetic waves and this waves reach to the antenna of the tag. The RFID readers use radio frequencies to read the numerical item code and other stored data from the RFID tag. RFID tags are the components that hold information about the attached objects. They activate the circuits in microchip of the tag and then microchip modulates the waves to send them back to reader with its antenna. Information provided to the reader will be transformed into the digital data. There are access controllers that play a role in storing the information in the computer and communicating with the RFID middleware software. Based on this information hardware unit that function in this system can be listed as reader, antenna, tag and software units. Software units include access controllers, RFID middleware software and EPC Code.

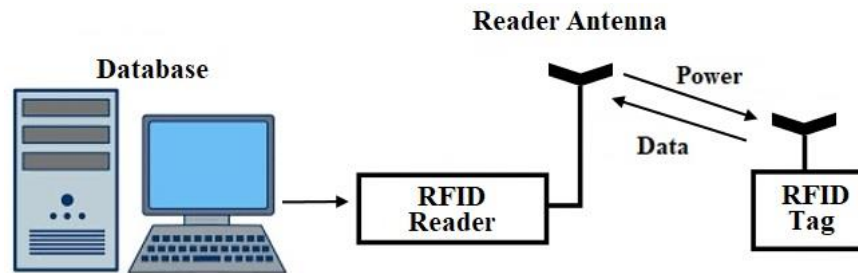


Figure 2.1 A block diagram of a typical RFID system (adapted from Boularess, Rmili, Aguilu & Tedjini, 2015)

Communication between the tags and the readers are provided by the antenna and it is called coupling. It is possible to extend the range of reading by adding more antennas or structure and power of the antennas. RFID system working progress has been shown in Figure 2.1.

Coupling can be done in two ways in RFID systems. Electromagnetic or magnetic. (Bhuptani & Moradpour, 2005) The method to be used is selected by considering the application requirements and criterias such as tag cost, reading speed, distance and tag size. Magnetic coupling is generally used in short distance RFID systems for access control applications (Pala, 2007). Electromagnetic coupling is a type connection most used in data communication between the tag and the reader. In this method, electromagnetic waves emitted by the reader reach to the antenna of the tag and power the circuits inside the tag. The tag's built-in capacitor harvests energy from the radio waves emitted from the RFID reader so the circuits are powered. This energy is used to send the signal back to the reader and then the reader converts the received signal into the digital data and the scanning process completes.

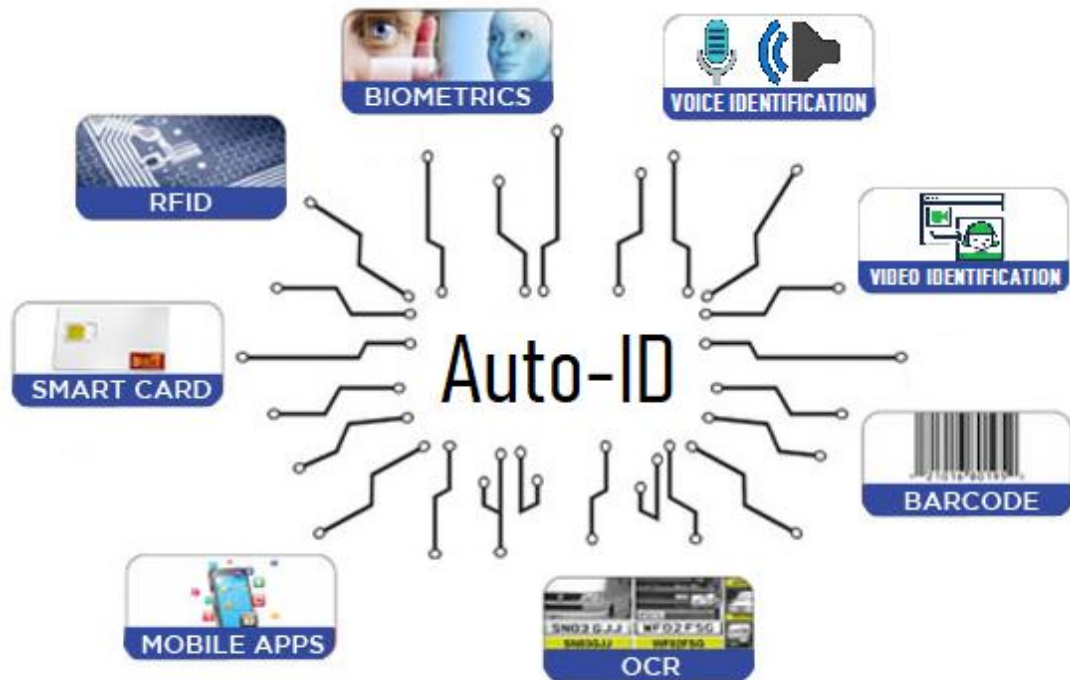


Figure 2.2 Working fields of Auto-ID group (adapted from Patel, Patel & Patel, 2009)

RFID is a part of Auto-ID group (Automatic Identification). Auto-ID often gets recognition with automated data collection. In another word, it means expressing assets, collecting data about them and evaluating the information rapidly without manually counting the collected data by using that technology. (Demirel, 2007). Auto-ID Group is used in bar codes, voice identification, optical character identification,

biometric recognition, smart cards and RFID as shown in Figure 2.2 (Finkenzeller, 2003).

2.8.2 History of the RFID

The first work to be considered a milestone with RFID was published by Harry Stockman in October 1948 under the name of Communication by Means of Reflected Power (Stockman, 1948). Stockman's article became more valuable and important after 30 years. Moreover, developments such as integrated circuits, communication network tools, transistors and microprocessors were also required. After the developments about radio and radar technologies in the 1930s and 1940s, the 1950s was a breakthrough for RFID. The use of RFID in aviation caused the exploration of the many new technologies such as the separation of the identification friend or foe (IFF) aircraft (Kocamaz, 2008).

In the early 1970s, Checkpoint and Sensormatic companies started their first commercial applications, electronic article surveillance (EAS). This application was specifically developed against theft (Dilonardo, 2014). In EAS systems, products and materials are monitored electronically. In the system, the system controls the tag that is placed on the objects and yields warning signal and the tags are 1-bit. EAS systems, which include anti-theft, battery-free, inexpensive and simple passive tags especially used in libraries and clothing stores.

RFID applications in the 1980s spread over many areas. Animal tracking systems in Europe have become very popular, and are also equipped with RFID for the toll roads in countries such as Norway, Portugal, France and Italy (Bazaatı, 2012). The 1990s are a remarkable period for RFID. The toll system in Oklahoma City which opened in 1991, allows cars to pass quickly without discontinuation (Bazaatı, 2012). Developments continued through the development of integrated circuits in the 1990s until RFID tags were reduced to a single integrated circuit size. Moreover, intergovernmental standards have been determined in frequency spectrum allocation.

In the 2000s, the production of small sized RFID has been appeared. Small sized RFIDs were integrated into many systems. The prices have been reduced. Special

switching and coding systems have been developed. US government agencies such as large retail chains and department of defense (DoD) have started to take sanction decisions for their suppliers to use RFID systems in these years. In the same years, studies gained speed to get a better standard of EPC with the initiative of Massachusetts Institute of Technology (MIT)'s Auto-ID Labs. Global Standards One (GS1), which operates in the determination of international standards in the supply chain, has set up the EPCGlobal Organization in cooperation with MIT, Cambridge, and the University of St. Gallen. This non-profit organization operates to improve RFID standards. Each product can be tracked with a RFID tag's unique id in EPC system. EPC information service (EPCIS) makes the product information can be accessed from anywhere in the world with object naming service (ONS) (Üstündağ & Tanyaş, 2009).

Tags are manufactured in a form which they can stick to glass or objects. They changed to smaller microwave tags consisting of a single complementary metal oxide semiconductor (CMOS) integrated antenna and a circuit. RFID based electronic payment transactions have become more popular in the United States and 5.9 GHz wide bandwidth was allocated for RFID by the Federal Communications Committee (FCC). The transportation has become a field of RFID applications with the rapid development process in the late 1990s. In MIT; an Auto-ID center was opened about RFID. EPC Global has become the leader of the standard development work in RFID applications so many standards were introduced by the international organization for standardization (ISO) (Chiesa, Genz, Heubler, Mingo, Noessel, Sopieva, Slocombe & Tester, 2002). As a result, the real adventure of RFID, which started with the World War II as a radar, has changed and still continues to improve and change today.

2.8.3 RFID Tag

The tag is a chipset (chip) in which the objects information is stored and components such as an antenna to communicate with the reader. Layer structure of a tag is shown in Figure 2.3. It uses Radio Frequency to communicate with the reader.

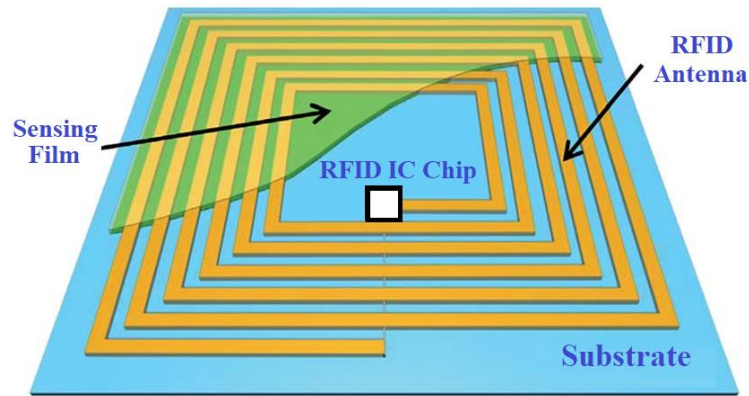


Figure 2.3 A tags microstructure (adapted from Singh, Singh & Nalwa, 2017)

Tag surfaces can be covered with various materials. Every tag has its own unique id. Memory capacities of tags can be in a range between 64 bits and 8 megabytes. RFID tags can communicate with the readers without having to be in physical contact. Things to consider for choosing an RFID tag are as followings: Surface that will be tagged (metal, plastic, wood etc..), reading range that is wanted, size of tag, extremity of environmental conditions (extreme hot, cold or moisture), method used to add the tag (clinching, fixating, cable connections) (Atlas RFID Solutions, 2019).

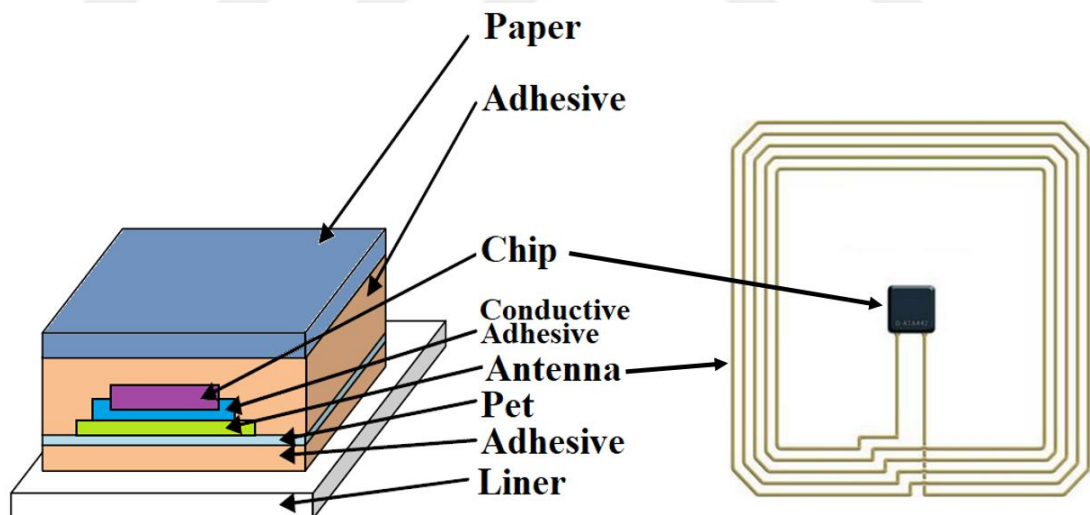


Figure 2.4 Tag construction (adapted from Repanovici & Cristea, 2011)

Tag construction and all the layers can be seen in Figure 2.4. Polypropylene or paper are usually used as a first layer to protect the other layers. There is an adhesive layer under this layer. It can be pressure sensitive or hot melt. There is conductive adhesive linking the antenna and the integrated circuit (IC) or chip. It can be a tape, an epoxy or a paste. The antenna is attached to a substrate of plastic which is usually PET.

The antenna is made of copper or aluminium. Liner is the last layer which is attached to the others by an adhesive layer. Liner is a silicone-coated paper.

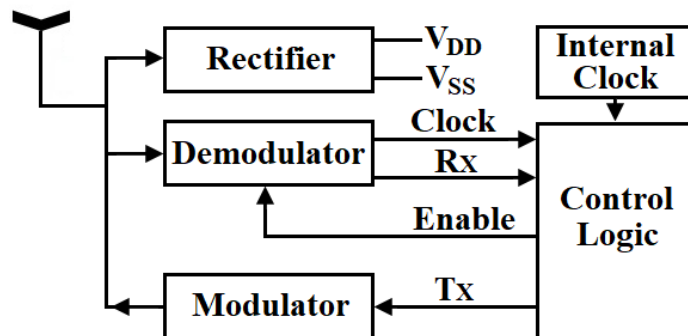


Figure 2.5 General architecture of a passive RFID tag (redrawn from Vaddi, Agarwal, Dasgupta & Kim, 2011)

General architecture of a passive RFID tag is shown in Figure 2.5. Generally, microstructures of RFID tags contain memory, antenna power unit (in active tags), coding/decoding and communication control units. Some of the currently used tag examples can be seen in Figure 2.6.



Figure 2.6 Tag examples (adapted from RFID-Internet Archive, 2019)

RFID tags can be grouped according their energy harvesting method, memory type and working frequencies. Here we will give some brief information about that groups. Groups can be seen in Figure 2.7.

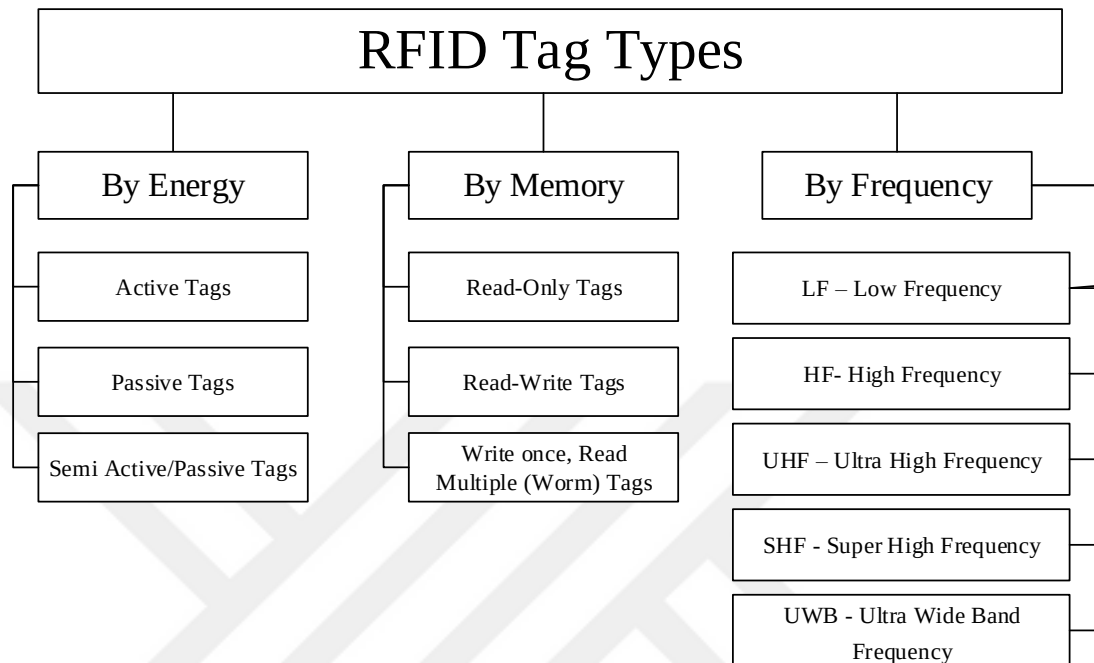


Figure 2.7 RFID tag types

2.8.3.1 RFID Tag Types by Energy Harvesting Method

Tags require energy to communicate with readers. They either take that energy from the reader or they themselves carry a power supply. The way of harvesting that energy is important in choosing a tag because reading range, lifetime, cost may differ in these tags. There are 3 ways that an RFID Tag can acquire energy. The detailed comparisons are given in Table 2.1.

- Passive RFID Tags

These tags don't have a built-in power supply. They're passive about first contacting the reader. That is why they require the reader to activate their chip and to communicate with the reader. Required energy comes from the reader via radio waves. This energy triggers the tag's chip by creating a voltage induction in the tag antenna. The chip sends tag's information to the reader via its antenna. Passive tags are used in many areas because they are cheap and sufficient in many fields. These type of tags

are affected by environmental effects more than other types so this is a big disadvantage in comparison.

Table 2.1 The general characteristics of the tags according to their energy acquiring method (Weis, 2007; DLR-TL001, 2019; Cisco Systems, 2008; Zavvari & Patel, 2012)

Tag Types	Passive Tags	Semi-Passive Tags	Active Tags
Power Source	Harvesting from the reader	Battery&Harvesting from Reader	Battery
Communication	Only Send Responses	Only Send Responses	Sends Responses or can start communication
Distance	LF, HF : 0,2m UHF, SHF : 3m UWB:10m	More than 100m	More than 100m
Cost	Cheap	Expensive Than Passive Tags	Most Expensive Tags
Applications	EPC, Short distance cards	Electronic transition, pallet tracking	Animal tracking, large scale goods tracking
Samples			

- Semi-Passive RFID Tags

Semi-passive RFID tags carry their own power supply on their chips but they require the reader energy to harvest energy for communication. These type of tags use the built-in battery to power the chip circuits or sensors and don't use it to create the response to the reader. They don't get activated without being read by the reader. They have a greater range and cost more than passive tags.

- Active RFID Tags

They have their own battery for both their chip and communication with the reader. In short, without waiting the reader, active tags can send signals to the reader. Their reading range is wide and they are the costliest tags.

2.8.3.2 RFID Tag Types by Memory Access Types

As we group the RFID tags by energy harvesting methods, we can group them by their memory access types. RFID tags have their own memory. The capacity of the memory mostly so small comparing the today's memory capacities. These memories

can be grouped by access types. These groups were explained in the following paragraphs.

- Read Only Memory Tags

These tags get programmed in production and then cannot be changed after manufacturing. They have a limited memory capacity and they're cheap. Most of them are passive tags (Yılmaz, 2013).

- Readable and Writable Tags

With these type of tags, it is possible to both read and write. They usually have a memory capacity of 32 to 128 kilobytes. Their cost is higher than the read-only memory tags. That is why they don't get used in the systems that the low cost is primary requirement. They are available in both passive and active tags and they use EEPROM, FRAM and SRAM memory types.

- One Write-Multiple (Worm) Read Tags

Under normal conditions they're only identical to read-only tags. Main difference is, while read-only tags are manufactured with random IDs, worm tag IDs can be updated with the request of the user. Again it is not possible to change the information after manufacturing. They usually have a 128 bit - 1 KB memory capacity range.

2.8.3.3 RFID Tag Types by Frequencies

RFID tags also can group by their working frequencies. They can operate at different frequencies according to their reading distances, power requirements, and data send capabilities. High frequency provides faster and longer distance data transmission than low frequencies. Low frequency tags work better than high frequency ones in the areas contain liquid and metal. If the distance between the reader and the tag is short, effects of the environment will decrease (Ergen, 2008).

- Low Frequency (LF) Tags

These tags generally work in 120-140 kHz frequency range. Generally passive tags work at low frequencies. Reading range is usually in between 1 centimetre and 2 meters. They can operate at close ranges in harsh environments like metal and liquid. For this reason, it is used in animal tracking by being implanted to those animals. LF tags are used in car immobilizer systems, too. The tag inside the key is activated when it gets closer to ignition switch. Data read rates of the LF tags are low (Weis, 2007).

- High Frequency (HF) Tags

These tags work at 13.56 MHz frequency. They can be read between 20 cm and 1 meter range. This type of tags is used in smart cards (MIFARE, ISO / IEC 1443), credit cards, object tagging, ID cards, baggage monitoring at airports, building entrance controls. Compared to LF tags their data read range is higher but they cannot be read as secure as LF tags in environments such as liquid and metal. Since they operate in a narrow frequency band, HF tags cannot be used in applications that require high security.

- Ultra High Frequency (UHF) Tags

These tags work in a range of 860-928 MHz frequencies. While tags in Europe are working in 868-870 MHz range, in America and Canada they work at 902-928 MHz. They work at 865.6-867.6 MHz range, in Turkey. They can be read in a range of 60 centimetres and 12 meters. They are mainly used in object tracking, product supplying and managing and pallet tagging. Comparing to LF and HF Tags their ability to be read in liquid and metal environments is very low. There are some materials have been developed to try to prevent this, but it was not very successful (Weis, 2007).

- Super High Frequency (SHF) Tags

These tags can work in a range of 2.45 GHz and 5.8 GHz frequencies and they are known as microwave frequency tags. They can be read in a range up to 30 meters. This technology has been developing rapid recently. While being heavily used as semi-passive and active tags, they can also be used as passive tags too. Semi-passive

microwave tags are often used in fleet identifying and toll systems. Although reading rates are higher than UHF tags, their costs are expensive. Also they use more energy compared to LF tags. Another important problem is that it interferes with the 802.11 b/g Wi-Fi network.

- Ultra Wide Band Frequency (UWB) Tags

These tags work in a range of 3.1 GHz and 10.6 GHz frequencies. It's a fairly new technology. Rather than sending a very strong signal from a certain frequency, with the UWB we can send a low power signal over a wide band. It is a better way to transmit signals. It can be read in a range up to 200 meters. The signals are not affected by the liquid or metal environment. Therefore, they are preferred to be used in health sector and hospital applications (Weis, 2007).

2.8.4 RFID Antennas

RFID antennas are components that used to collect or send electromagnetic waves. They provide communication between “reader and reader” or “reader and tag”. Since the tag reading range is limited in many cases antenna usage is very important. The antennas must achieve the best signal reception at low power and adapt to specific conditions. Antennas should be designed in different length, shape and frequency ranges to ensure the best performance, depending on the environment in which the applications operate and the distance required by the application (Ranasinghe, Leong, Ng & Cole, 2006; Krauss & Marhefka, 2002).

There are two types of antennas and these are planar transmission and the circular transmission antennas. Planar transmission antenna concentrates on a single axis for maximum gain at the longest possible reading distance. On the other hand, circular transmission antenna distributes the generated UHF energy evenly over longer distances. Thus, it is ensured that all the tags in the environment are read by the circular directional propagation. (Hossain & Karmakar, 2006; Ahson & Ilyas, 2008). These specific features of antennas are important for antenna selection that is why there are different type of designs available (Uçar, Sondaş & Erdemli, 2010). Usually RFID applications use antenna elements with circular polarization. The basic reason for that

choice is to ensure the reliability of the communication between the reader and the tag that placed in any direction (Uçar et al., 2010). Also one reader can be connected with more than one antenna. In Figure 2.8, some antenna examples can be seen.



Figure 2.8 RFID antennas (adapted from Boer Barkod&RFID, 2019)

2.8.5 *RFID Readers*

The RFID Reader has a microcontroller and components such as antenna, top detector (for tag detection) and comparators. They can emit signals, share energy and make data transfer by using their antennas. Some tags take their energy from the readers' signals sent by their antenna. The tag uses that energy to activate its own microchip. With this energy, the tag can send data to the reader by increasing or decreasing the resistance of its antenna. Readers now operate in the frequency range of 125 kHz to 2.4 GHz in the LF, HF, UHF and SHF and UWB frequency bands. Figure 2.9 shows the examples of fixed and mobile RFID readers commonly used.



Figure 2.9 RFID readers (Swedberg, 2013; Alien Technology LLC., 2019)

2.8.6 *Access Controllers*

Access controllers are computer or servers running on database software or application software. Access controllers can be seen as the brain of RFID systems and they control the RFID middleware software. They are used to connect multiple readers

in a network environment, to process information centrally. The controller uses data collected by readers inside an area.

2.8.7 Middleware Software

A tag can be read multiple times by the same or different readers at different points. As a result, a large amount of data is generated on the RFID system network by reading the tags. To hold and manage the generated data effectively, they should be listed, filtered and processed. At this point, middleware software steps in. The middleware software is a technology which is responsible from the RFID system, services, transmitted data and managing communication among RFID systems. Middleware software enables data to be used efficiently, transferred within a selected network and compressed (Yüksel & Zaim, 2009). There are middleware software programs such as Savant, WinRFID, WebSphere, Sun Java and FlexRFID.

2.8.8 Reader Interface

It provides interactions with RFID hardware. It is the bottom layer of the middleware software. It contains drivers for all system supported devices. It is responsible for managing all hardware-related parameters such as wireless broadcasting, host-side communication and reader protocols.

2.8.9 EPC and EPCglobal

Electronic product code is a number identifies objects. It is a unique and universal identifier number. A global mercantile product number "EPC" has been seized to transfer built-in standards from the barcode to RFID tags. EPC tags can be used in many different places for the end users. These tags can be used to inform the store owner, if a product is stolen or got out of the store without make a payment for it (Castelluccia & Soos, 2007).

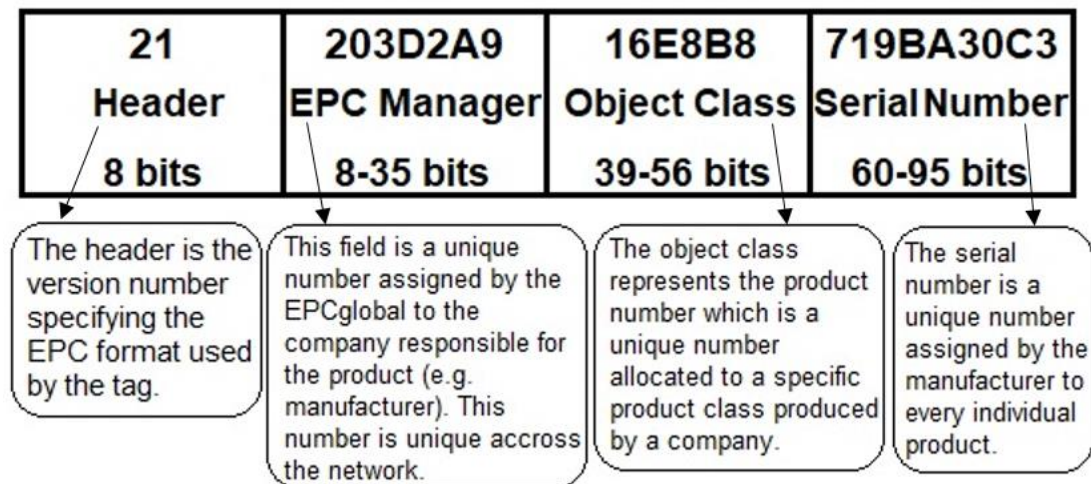


Figure 2.10 Context of a 96-bit EPC (redrawn from Castro & Wamba, 2007; Leong, Ng & Engels, 2005; Harrison, McFarlane, Parlikad & Wong, 2004)

EPC code is a standard code with header and the code has different sizes, such as 64-bit, 96-bit, 128-bit, and 256-bit, consisting of 3 array data, EPC Manager Number, Object Class and Serial Number. The header specifies the version number of the EPC, the second part of the number is EPC Manager, and the third part called the Object Class specifies the type of object. Unique serial number is given by manufacturer to each product. Today, the most widely used electronic product code in supply chain applications is the 96-bit EPC (Finkenzeller, 2005; Glover & Bhatt, 2006). An EPC code is shown in Figure 2.10.

EPCglobal has focused on developing a global network system by combining both RFID technology and EPC. EPCglobal is a type of membership based on organization of industry leaders and academic organizations to ensure the standardization and worldwide adoption of EPC technology. Intention of EPCglobal is to increase object visibility, effectiveness, manageability throughout the supply chain and to provide high quality information flow between companies and trade partners. (Lahiri, 2005; Brown, 2006). EPCglobal's Turkey representative is GS1 Turkey who is within the Union of Chambers and Commodity Exchanges of Turkey (TOBB) (Türkiye Odalar ve Borsalar Birliği [TOBB], 2019).

2.9 Wireless Identification and Sensing Platform

WISP is a wireless identification and sensing platform. It is a battery-free RFID tag to be supported by the Ultra High Frequency (UHF) RFID readers. The outstanding features of the device are a fully programmable ultra-low-power 16-bit flash microcontroller with wireless power supply, UHF backscatter communication, and an analog-to-digital converter. (Yeager, Sample & Smith, 2008) WISP is a passive RFID has built-in sensors. WISP tag has a built-in temperature, accelerometer sensor and three clocks named Digitally Controlled Oscillator (DCO), Sub-Main clock (SMCLK) and Auxiliary clock (ACLK) (Elbal, 2015). WISP 5.1 also has Texas Instruments MSP430FR5969 microcontroller (Yıldırım, Aantjes, Majid & Pawełczak, 2016). WISP also can include camera etc. devices integrated on it (Naderiparizi, Parks, Kapetanovic, Ransford & Smith, 2015). Figure 2.11 and Figure 2.12 show the technical structure of the accelerometers and temperature sensors on WISP 5, respectively.

The accelerometer ADXL362 attached on WISP is a very low-energy accelerometer with a 3-axis. It uses less than 2 μA energy for 100 Hz data output. ADXL362 does not identify input signals by insufficient sampling unlike accelerometers that use a power service cycle to achieve low power consumption. The sensor performs sampling at all speeds and all bandwidth. The ADXL362 always has a 12-bit output. It has $\pm 2\text{ g}$, $\pm 4\text{ g}$, and $\pm 8\text{ g}$ measurement ranges.

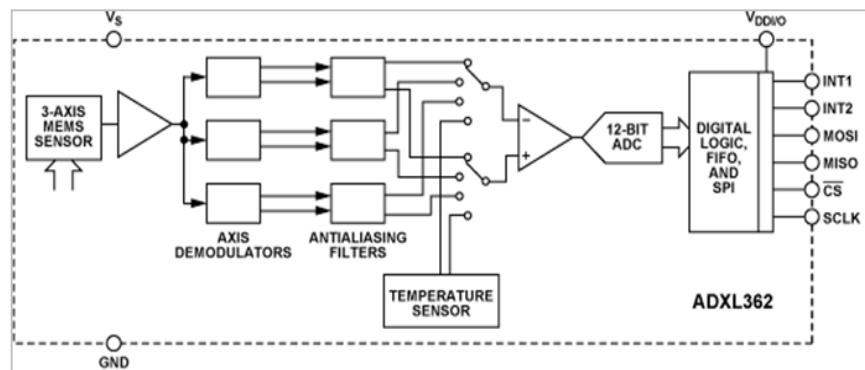


Figure 2.11 ADXL362 accelerometer sensor (ADXL362, 2019)

The AD22100 temperature sensor is a ratiometric sensor proportional to the output voltage supply voltage. The sensor mainly operates with a special temperature-related resistance.

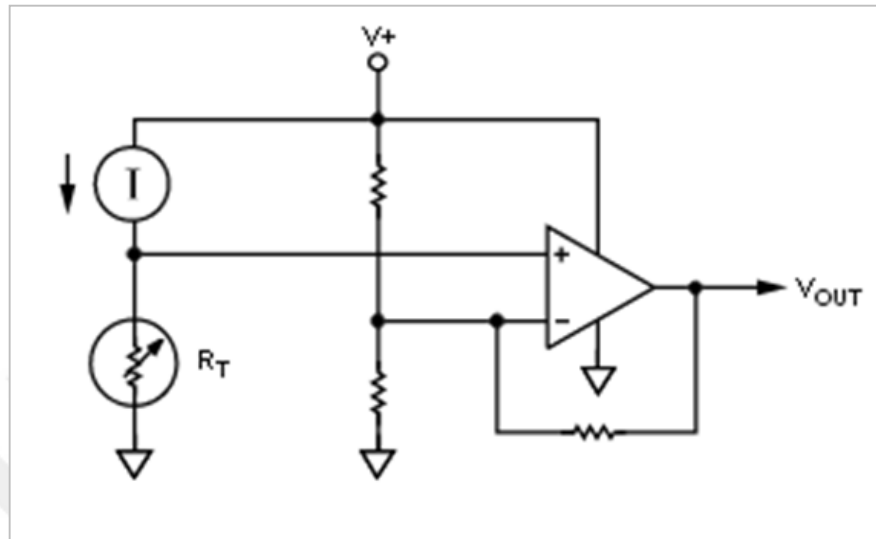


Figure 2.12 AD22100 temperature sensor (AD22100, 2019)

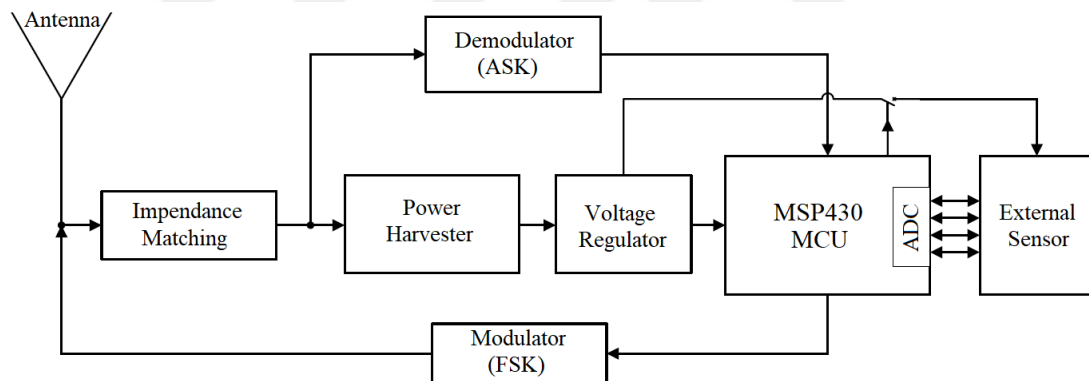


Figure 2.13 WISP system block diagram (redrawn from Sample, Yeager, Powledge & Smith, 2007)

WISP's block diagram can be seen in Figure 2.13. WISP's MSP430 microcontroller has two clocks DCO and VLO that are working at different frequencies. These two different clocks are used to generate signals. The SMCLK clock can be reset to 1, 8, 12 and 16 Mhz. Similarly, the ACLK clock running at 12 kHz can also be reset. The registers required to perform all these operations are in the MSP430 processor. The SMCLK clock, which is basically running at 1 MHz, is actually oscillating between 0.9 MHz and 1.1 MHz. Time difference of both clocks can be used for random number

generation (Fujdiak, Misurec, Mlynek & Raso, 2013a). MSP430 clocks can be seen in Figure 2.14.

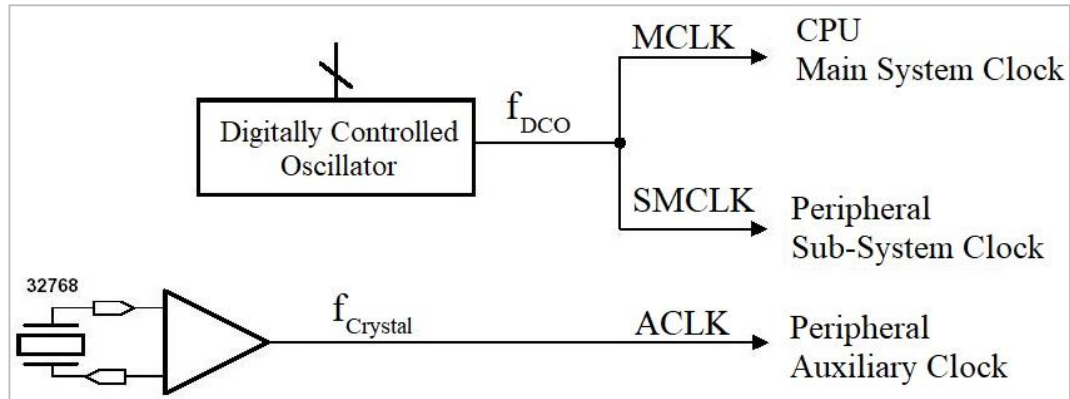


Figure 2.14 MSP430 clock system (redrawn from Wisman, 2015)

2.10 Random Number Generators

Random numbers are the numbers satisfy following two conditions: Numbers should distribute uniformly in a known interval or set. It cannot be predictable to current or following numbers by using past ones. Results of studies has shown that the real randomness can only be achieved by the physical events occurring in nature (Chaitin, 2012). Examples of these physical events include thermal noise of semiconductor chips, radioactive sources such as the sun, or very small variations in the numerical representation of the temperature that changes during the day.

Random number generators are the tools used to generate random numbers. In addition to physical tools, there are software tools based on the algorithm, functions based on probability distribution and data generated by people. Although the true random number can be provided by physical tools, thousands of random numbers may be needed in a short time period to solve some problems. It is not possible to obtain a large amount of random numbers from physical tools in a short time easily. This is why software-based random number generators are one of the most important areas of work today. Since these random numbers are generated by sequential mathematical operations, they are not exactly random numbers (Wetzel, 1999). Therefore, these randomly generated numbers are called the pseudorandom numbers (PRN) and their generators are called Pseudorandom Number Generators (PRNG).

In some applications, both physical sources and software-based generators are used in mixed way. We can call these type of the generators as Mixed/Hybrid Random Number Generators (HRNG).

2.10.1 Area of Use and Importance of Random Numbers

In this section, we will give information about the area of use and importance of random numbers and random number generators. Random number generators have a wide range of use in luck games, statistical sampling, computer simulations, cryptography, purely random designs, computer games (such as random distribution of parts) and other areas that produce unpredictable results. If we look at it deeper, the random numbers serve a wide range from the security sector to the entertainment sector. In the case of security applications, physical and hardware-based random number generators are more preferred than pseudorandom number generators. Since the pseudorandom number generators produce the same random numbers for a given initial value, they are very useful in the problems where the same random numbers should be used repeatedly. This initial value used in pseudorandom numbers is called seed value. One of the best example for problems requiring the same random number sequence again and again is Monte-Carlo simulation experiment.

Pseudorandom number generators are also used in cryptography for the use of symmetric encryption. In cryptographic operations, by using the same random number sequence in their own devices, as a key value in the cryptographic process users can communicate with each other in a coded manner by using a common feed value that is shared between the sender and the recipient of the message. Today, when the banks cannot send a short message service (SMS) message for internet banking login process (such as going abroad), there are applications such as password matic that gives the same seed value to the algorithm that generates the random number and uses this pseudorandom number in the login process. The security of encrypted systems is based on the fact that confidential data (such as confidential outline) is known to those who are authorized to know and cannot be predicted or known by others. Here there is a need to solidify the confidentiality of the data and random values provide security against others. Malicious users can create security vulnerabilities by using the weaknesses of random number generating methods.

2.10.2 True Random Number Generators

The hardware random number generator (true random number generator) used in the programming. These generators produce random numbers by using a physical operation, not using a computer program. Such devices generally use events that include microscopic and statistically random events such as noise signals, thermal noise, beam splitter using the photoelectric effect and other quantum effects. These stochastic processes are not predictable in theory and are subject to experimental test results according to the proposition of theory itself.

2.10.3 Pseudorandom Number Generators

The pseudorandom number generators are used to generate number sequences such as real random number generators. These generators are an algorithm and generate sequences with properties close to TRNG properties. They can also be a random number generator manufacturer. PRNG can't produce real random numbers. As we can understand from its name, they can produce a pseudorandom number sequence. These generators begin to produce numbers with an initial value (seed) and the sequences of numbers they produce depends on this seed. Although these algorithms use different seed sources, they often use the time factor as their initial value (Dutang & Wuertz, 2009). The pseudorandom number generators are very important because they generate numbers quickly and can be used or produced repeatedly.

2.10.4 Hybrid/Mixed Random Number Generators

This type of random number generators is not purely mathematical; they also include a physical source. On the other word, they use combination of two source. The word "hybrid" refers to the use of two basic methods to build this generator. These two basic methods are mathematical expressions or algorithms and physical components, process or hardware. On the other word, this generator is a mix of a TRNG and a PRNG.

2.11 Genetic Algorithm

Genetic Algorithms (GA) is an intuitive search method based on biological characteristics of natural genetics such as selection, mutation and inheritance. These algorithms utilize evolution and natural selection mechanics to produce optimization solutions. They are a kind of Evolutionary Algorithms (EA) class. The GA is based on the assumption that individuals in nature adapt to environmental conditions and that new populations are formed with randomly determined choices. In addition, some mutated properties and inheritance can be varied. In this way, it produces a solution to survive in nature and to keep up with the environment. GA produces generations with more dominant characteristics. It needs a fitness function for all these operations. This function is used to identify individuals to be selected for new generations. Fitness function is crucial to produce accurate and powerful generations (Davis, 1991).

2.12 Authentication Protocols

For RFID systems, authentication is the important and first step that defends the system against wireless attacks. The server authenticates the RFID tag and validate its identity so the tag become a trusted party of the communication. The reader can access the authenticated tag's data after authentication process is success.

Hash function, message verification code (MAC), cyclic redundancy code (CRC) and pseudo-random number generators (PRNG) are common solutions used for security needs in lightweight protocols. Symmetric encryption is used by the full-fledged protocols. Many authentication protocols are designed to perform a single reader-single tag authentication process. In real life, there are multiple readers-single tag, multiple tag- single reader, and a multiple tag- multiple reader cases. On the other word, a single reader can simultaneously authenticate multiple tags or a single reader can simultaneously authenticate the multiple reader. This type of protocols is called as grouping proof authentication protocols.

CHAPTER THREE

LITERATURE REVIEW

In this chapter, a literature review was given. This chapter includes three main parts. In the first part it is focused on the literature about the random number generation methods. In second part the literature about the authentication protocols related with our thesis was reviewed. In the last part literature about some general security problems for RFID systems was given.

3.1 Random Number Generation Methods

RNGs are mostly studied from the point of view of their proximity to true randomness and reliability until last decade since they were in use in relatively powerful computers. But when smart mobile devices gained in popularity and eventually the IoT concept introduced, the idea of moving RNGs to mobile devices with (sometimes very) limited capabilities brought performance issues to the sight. In example, the standardization document of randomness recommendations for security, RFC 1750, suggests data encryption standard (DES), cryptographic hash functions and some other (not so simple) methods to provide randomness however, there is no concern for lightweight devices (Eastlake, Crocker & Schiller, 1994).

Before proceeding to the RNG algorithms, it would be accurate to take a glance at the review criteria for these RNGs. There are several popular randomness tests (or more accurately, test suites) used to examine generator functions, those are supposed to have random outcomes. The ones we'll refer to are, the Diehard battery of Marsaglia (Marsaglia, 1995), TestU01 suite of L'ecuyer and Simard (L'Ecuyer & Simard, 2007), which consists of 6 test batteries, and the NIST test suite having 15 tests (Bassham et al., 2010). The more recent publications of NIST, namely SP800-90 (a, b, c), are also taken into account (Barker & Kelsey, 2012; Barker & Kelsey, 2016; Turan, Barker, Kelsey, McKay, Baish & Boyle, 2016). The generators we proposed and the TestU01 suite comply with most (if not all) of the instructions given in these papers. In example, SP800-90b (Turan et al., 2016) implies the need of a dataset containing at least 1,000,000 values, yet TestU01 produces around 50,000,000 numbers to test the

generators (L'Ecuyer & Simard, 2007). Our ultimate generator, can be used in the frameworks mentioned by NIST (Barker & Kelsey, 2016), with no or very little modification depending on the scenario.

Table 3.1 Some popular and known Pseudorandom Number Generators

Generator	Date	First proponents	References
Middle-square method	1946	J. Von Neumann	(Von Neumann, 1951)
Lehmer generator	1951	D. H. Lehmer	(Lehmer, 1951)
Linear congruential generator (LCG)	1958	W. E. Thomson; A. Rotenberg	(Thomson, 1958; Rotenberg, 1960)
Lagged Fibonacci generator (LFG)	1958	G. J. Mitchell and D. P. Moore	(Knuth, 1998)
Linear feedback shift register (LFSR)	1965	R. C. Tausworthe	(Tausworthe, 1965)
Wichmann–Hill generator	1982	B. A. Wichmann and D. I. Hill	(Wichmann & Hill, 1982)
Rule 30	1983	S. Wolfram	(Wolfram, 1983)
Inversive congruential generator (ICG)	1986	J. Eichenauer and J. Lehn	(Eichenauer & Lehn, 1986)
Park-Miller generator	1988	S. K. Park and K. W. Miller	(Park & Miller, 1988)
MIXMAX generator	1991	G. K. Savvidy and N. G. Ter-Arutyunyan-Savvidy	(Savvidy & Ter-Arutyunyan-Savvidy, 1991)
Add-with-carry (AWC)	1991	G. Marsaglia and A. Zaman	(Marsaglia & Zaman, 1991)
Subtract-with-borrow (SWC)	1991	G. Marsaglia and A. Zaman	(Marsaglia & Zaman, 1991)
Maximally periodic reciprocals	1992	R. A. J. Matthews	(Matthews, 1992)
KISS	1993	G. Marsaglia	(Marsaglia & Zaman, 1993)
Multiply-with-carry (MWC)	1994	G. Marsaglia; C. Koç	(Marsaglia, 1994; Koç, 1995)
Complementary-multiply-with-carry (CMWC)	1997	R. Couture and P. L'Ecuyer	(Couture & L'Ecuyer, 1997)
Mersenne Twister (MT)	1998	M. Matsumoto and T. Nishimura	(Matsumoto & Nishimura, 1998)
Xorshift	2003	G. Marsaglia	(Marsaglia, 2003)
Well equidistributed long-period linear (WELL)	2006	F. Panneton, P. L'Ecuyer and M. Matsumoto	(Panneton, L'ecuyer & Matsumoto, 2006)
A small noncryptographic PRNG (JSF)	2009	Bob Jenkins	(Jenkins, 2009)
Advanced Randomization System (ARS)	2011	J. Salmon, M. Moraes, R. Dror and D. Shaw	(Salmon, Moraes, Dror & Shaw, 2011)
Threefry	2011	J. Salmon, M. Moraes, R. Dror and D. Shaw	(Salmon et al., 2011)
Philox	2011	J. Salmon, M. Moraes, R. Dror and D. Shaw	(Salmon et al., 2011)
SplitMix	2014	G. L. Steele, D. Lea and C. H. Flood	(Steele, Lea & Flood, 2014)
Permuted Congruential Generator (PCG)	2014	M. E. O'Neill	(O'Neill, M. E., 2014)
Random Cycle Bit Generator (RCB)	2016	R. Cookman	(Cookman, 2016)
xorshiftR+	2017	U.C. Çabuk, Ö.Aydın, G. Dalkılıç	(Çabuk, Aydın & Dalkılıç, 2017)
xorshiftL+	2018	Ö. Aydın, G. Dalkılıç	(Aydın & Dalkılıç, 2018)
Xoroshiro128+	2018	D. Blackman, S. Vigna	(Blackman & Vigna, 2018)

In fact, TestU01 suite contains, Small Crush, Crush, Big Crush, Alphabit, Rabbit batteries and a pseudo-NIST battery. This NIST battery contains 13 original tests out of 15 included in the real NIST suite, and the remaining 2 are replaced by more advanced variations (L'Ecuyer & Simard, 2007). Therefore, it will be used one of our experiments. Apart from the TestU01 suite (and the NIST suite), there is another measure of randomness for some (especially for RFID-compatible) lightweight RNGs, defined in the EPCTM UHF RFID Generation-2 standardization document. It is used to test the suitability of our ultimate generator for UHF RFID devices.

The development of the pseudo-random number generator (PRNG) has a long historical process. In 1971, Vernam discovered an arbitrary set of codes called one-time pad (Zeng, Yang, Wei & Rao, 1991). Since then, pseudo-random number generators have been used in many scientific and technical fields. (Schneier & Sutherland, 1995; Kotulski, 2001). Some of the areas where pseudo-random signals are typically telecommunications (Wesołowski, 2006; Stępień & Walczak, 2012), stream cipher, scrambling and descrambling circuits. It can be seen a list of well-known PRNGs in Table 3.1.

Physical generators are the first used method. Dice throwing, throwing money or roulette wheels are the most widely used ones. Although these methods are preferred in gaming or paper games, they are not suitable for use in areas such as statistical and cryptography that require many random numbers in a short time because of the low number of random number generation. As the physical number generator, there are also applications that produce a random number using the amplitude of atmospheric noises and applications that offer this as a service. In a study on the production of random numbers by quantum technology in 2014, random number production studies were performed by using the photons obtained from picture frames taken from phone cameras. they showed that they formed unpredictable random numbers (Sanguinetti, Martin, Zbinden & Gisin, 2014). Since quantum technology usage in random number generating costs are highly expensive, it is not used widely. But there is a thinking when the time comes and quantum technology becomes popular it will be also used in random number generating (Sanguinetti et al., 2014).

If we think about random number generators according to the probability distribution, some random number generators use probability density functions to produce a random number. These methods produce a random number equivalent to the true random numbers because they provide a steady random number generation. Lastly, there are studies to produce random numbers over the data generated by people. For example, the speed and hardness of a person typing at the computer keys of the keyboard is measured by sensitive devices. Stored data is used to generate the random number. But since it cannot provide true randomness as much as other methods it is not very common to use.

According to the standard of EPC Global Class 1 Generation 2 (EPC Gen-2), communication between the RFID reader and RFID tag was shown in Figure 3.2. As we understand from the figure, there is no cryptographic algorithms although there exists features like random number generation (RNG) and CRC function usage.

The second point is the CRC function that can create security vulnerabilities on authentication protocol. CRC function is a linear function and has homomorphism feature so it is not secure to use cryptographically (Peris-Lopez et al., 2009b). Despite renewed 2013 version of the EPC Gen2 has same security vulnerabilities, it is recommended to use cryptographic algorithms on tags due to the increasing processing power. Theoretical approach and method which will be followed during my thesis is beginning generating random number firstly. As we know, there is no chance to interfere standard RFID tags' microcontroller, I prefer to use WISP (Sample, Yeager, Powledge, Mamishev & Smith, 2008) RFID tag. There is random number generator function named rand() on WISP tag but this function has a low randomness (Fujdiak et al., 2013a).

True random number will be created from temperature (Voris, Saxena & Halevi, 2011), accelerometer sensors, voltage sampling on ADC and the difference between the two clocks' pulses (Fujdiak et al., 2013a). True random numbers can be created by using the method below:

Datas read from temperature and accelerometer sensors can be used for random number generation but if the temperature is stable or WISP is fixed on an unmoving

material so these two sensors won't be enough to generate random numbers. For this situation, it is possible to use MSP430FR5969 microcontroller's two different clocks' pulses. These two different clocks can be used to generate signal for the two different timers. SMCLK clock can be set to 1,8,12 and 16 MHz frequencies. On the other hand, ACLK has 12 KHz frequency. MSP430FR5969 microcontroller has all registers for these operations. SMCLK clock is running at 1 MHz and oscillating between 0.9 MHz and 1.1 MHz. These two clocks pulse difference can be used for random number generation (Fujdiak et al., 2013a).

MSP430FR5969 microcontroller has a 12 bits Analog to Digital Converter(ADC). It is possible to get digital values by sampling voltage from ADC. We can use least significant bit of these digital values for random number generation (MSP430FR5969, 2019). In each sampling we can create only one bit so random number can be created by 16 rounds. In each round number will be shift left for one bit. At the end of the round we will have a 16 bits random number.

A hardware random number generator generally includes a power amplifier and other electrical circuits for converting a type of physical power into an electrical signal. On the other word, they use a kind of analog to digital converter for converting the output into digital data. Generally, the output is the 0 and 1 which are the elements of binary number systems. Taken number samples generate sequential random numbers. Cryptography is the primary use of the hardware random number generators. It is important to generate crypto keys for secure data transmission. It is also widely used in internet encryption protocols such as Secure Sockets Layer (SSL). The hardware random number generators can also be constructed using random macro operations such as coin toss, roll roulette wheel and lottery machines. Unpredictability of these events can be explained by using unstable dynamic systems and chaos theory. Even though the macro operations cannot be random according to the Newton Mechanics. The results of a well-designed roulette wheel cannot be determined in practice because it depends on initial values, precisely. Although the dice were generally used as a part of the game in gambling, the Victorian scientist Francis Galton defined a path in 1980 to enable the dice to be used for producing scientific numerical numbers (Galton, 1890).

One of the most basic pseudo-random number generators is the linear feedback shift register (LFSR) generator (Golomb, 1982). The bit series created by this generator are mainly produced as deterministic series. (Haykin, 2001; Goresky & Klapper, 2006). LFSR is simply implemented on hardware and software. It can be widely used as a basic generator and is also used as a mixture of multiple generators (Schneier & Sutherland, 1995).

Marsaglia in his paper (Marsaglia, 2003) has shown a class of RNGs, called “xorshift RNGs”, consists of consecutive bitwise xor and shift operations using seeds. He claimed that the algorithms are extremely fast and reliable in terms of randomness. Since the speed of this algorithm family is proven, they will be in our scope in this research. However, that reliability claim is invalidated in some later studies (O’Neill, 2014). Figure 3.1 shows the main members of the xorshift family of RNGs according to their development timeline.

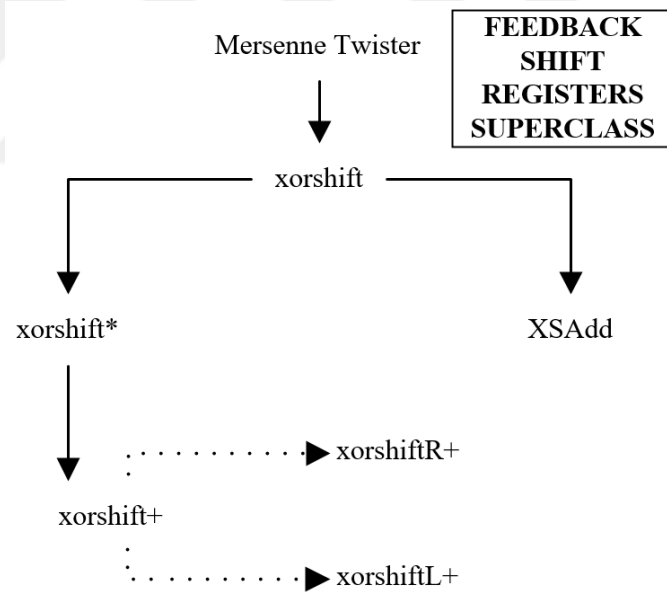


Figure 3.1 Development lapse of (some) xorshift based RNGs (Çabuk et al., 2017; Aydın & Dalkılıç, 2018)

Saxena and Voris used built-in sensors to generate random numbers using the ambient noise. In their study they focused on low cost sensors such as accelerometers. They aimed to analyse the ways of random number generation at low cost RFID tags. They claimed that the results of the study will be applied to present a procedure for

generating random numbers considering the previous works' security lacks (Saxena & Voris, 2009).

Arnault et al. has developed a linear feedback shift registers considering encryption algorithms (Arnault, Berger & Pousse, 2011). The hardware compatibility with the used devices, long period of random numbers, ease of the mathematical analysis has led to the wide use of these generators.

Hennebert et al. aimed to evaluate some sensors such as accelerometer, thermometer and magnetometer as an entropy source. They wrote that minimum entropy estimation of that sources is a challenge. They followed the NIST recommendations. They found the accelerometer, the vibration sensor, the magnetometer and the internal clock are interesting because these sources had less entropy than it was expected. They claimed that the temperature and air pressure sensors provided very little entropy. They showed that the vibration, magnetic sensor and accelerometer are the best sources to produce entropy (Hennebert, Hossayni & Lauradoux, 2013).

Kumar and Chatterjee used genetic algorithm in their paper to generate key for stream ciphers. Fitness function of the used genetic algorithm determines the key selection process. In each iteration repeated for key selection, the key was selected if it has highest fitness value. These selected keys were non-repeating/unique. They claimed that selected keys are random, highly encrypted and more secure for encryption of the data (Kumar & Chatterjee, 2016).

One of the parallel researches by Vigna (Vigna, 2016) revealed another xorshift based RNG, called xorshift*. The RNG performed well when compared to its predecessors, but was worse than the successors. However, it showed that xorshift family is flexible and is very open to further development.

After the famous and widespread “Mersenne Twister” (Matsumoto & Nishimura, 1998), in a follow-up study, Saito and Matsumoto came up with another similar algorithm “XSadd”, which is claimed to be more reliable. This claim is verified

according to the results of the tests of TestU01 suite (Vigna, 2017), however it is reported that, inverse of this algorithm fails some of the tests of the same suite.

A later and yet very recent study of Vigna again (Vigna, 2017), introduced another xorshift based RNG, “xorshift+” claimed to pass all tests of the TestU01 suite, and even inverse. Which is not an invalidated claim in any newer publication we could analyse. For this obvious reason, this RNG is chosen as a starting point in our study. In later sections of this paper, it is referred as the original xorshift+.

Some previous studies have been conducted to create random numbers or pseudo-random number generators using the Genetic Programming (GP) method, which is mostly used in optimization processes. Koza did a study to produce a PRNG using the genetic algorithm (GA) in 1991 (Koza, 1991). This study is one of the first studies using GP to create a PRNG. This generator aims to convert sequential non-random numbers to the random numbers. Recursive seeds were not used because it uses a sequence of consecutive numbers. He described the binary series with high entropy values as having higher randomness and used entropy calculation as a fitness function. Koza used simple tests such as gap and frequency to evaluate the randomness. More complex and extensive tests such as NIST STS have not been used. On the other hand, some mathematical operators are used such as modulus and division in that study. These operators are problem for many IoT devices with resource and time constraints. Therefore, it is not suitable for the devices with limited sources.

Tomassini et al.’s work generated pseudorandom sequences directly by using a cellular automaton (CA) instead of generating a deterministic PRNG (Tomassini, Sipper, Zolla & Perrenoud, 1999). The GA in this work is used for producing a suitable CA. This GA evolves the cells of a single, nonuniform CA, which has cell rules that are initialized randomly. This CA generator achieved good results within the Diehard test suite.

Mitra proposed a true random number generator using operational amplifiers. The circuit includes a positive feedback operational amplifier. Operational amplifiers of the MSP430 and IC LM741 were used in the study. Data-rate was low so generated data volume was low too. She claimed that generated numbers passed the NIST test

and can be used as a seed value. This generator was implemented on WISP (Mitra, 2012).

In Chlumecky et al.'s study, methodology and software using a GA and a newly created hydro random number generator (HRNG) for the optimization of a rainfall-runoff model were introduced (Chlumecký, Buchtele & Richta, 2017). Their new HRNG generates random numbers based on hydrological data and gives better results than PRNGs, as they claimed. Nevertheless, this paper does not include structural or statistical analysis of the HRNG's security.

Çabuk et al. (2017) in their work, proposed a new xorshift-based lightweight PRNG, called xorshiftR+, which is specially designed for the constrained devices of the IoT and tested on a WISP. The authors also obtained very satisfying results in terms of speed and randomness from NIST STS and a more comprehensive suite, TestU01.

Ibrahim and Dalkılıç focused on the security of healthcare systems, and especially authentication protocols, in their article (Ibrahim & Dalkılıç, 2017). They developed a new mutual authentication protocol using elliptic curve cryptography and the advanced encryption standard (AES) for RFID tags. This protocol was tested and coded on WISP passive RFID tags. Although they claimed that it overcomes the security problem for healthcare systems using RFID tags, they used a WISP 5 built-in random number generator for their authentication protocol. RNGs should be examined structurally and statistically to say that they are secure, and they did not give any information about the security status of that PRNG.

Knežević (2017), in his study, stated that evolutionary computation could be used to create PRNGs, which has an important role against side channel attacks. Knežević concluded that the GP method can speed up exponential computation or can be used to break the Merkle–Hellman cryptosystem in the asymmetric key cryptography and also can create Boolean functions or good quality S-boxes for symmetric key cryptography. In this article, the claims were not directly supported by experimental results, and only articles that support the assertion were reviewed.

Challa et al. investigated the latest authentication protocols proposed for implantable medical devices (IMDs) and WISPs in their study (Challa, Wazid, Das & Khan, 2018). In this paper, they gave comparative details considering IMDs' computation and communication costs and functionality features among the existing authentication schemes. They called attention to some difficulties that need to be mitigated for authentication schemes to be designed for IMDs. Although they mentioned the importance of random number generators in their paper, there was no evaluation about the resource usage and security analysis of those RNGs that have one of the key roles for authentication protocols.

Hernandez-Castro et al. used the strict avalanche criterion and the avalanche effect to calculate and measure the fitness of PRNGs (Hernandez, Seznec & Isasi, 2004). It generates an initial set of random numbers with the Mersenne Twister generator and randomly changes a single bit. After that, it calculates the Hamming distance between these values and the output of their PRNG.

Hernandez-Castro and Tomassini's studies produced good results with Diehard, but they did not contain the NIST STS results, which is a more comprehensive package than the Diehard test package (Sýs & Říha, 2014). Moreover, it was mentioned that only successful results were obtained with statistical tests and the results were not given for the measurements and tests performed on any equipment.

LAMED is another project that also uses the strict avalanche criterion as its fitness function (Peris-Lopez, Hernandez-Castro, Estevez-Tapiador & Ribagorda, 2009b). As in our project, a PRNG for low-cost, low-power RFID tags is produced considering the EPC standard, and a hardware implementation has been designed to test that PRNG, too. However, instead of running the algorithm on real hardware, the authors of LAMED preferred to create only a conceptual design.

Khan and Bhatia's work applied a GA on a binary sequence that was already generated and selected a key from that sequence (Khan & Bhatia, 2012). The coefficient of correlation calculation was used in the fitness function. Even though it has encouraging results, no known statistical tests were provided along with those results.

Leonard and Jackson's high entropy RNG production with single-node genetic programming (SNGP) work was also based on Koza's bit-entropy calculation method (Koza, 1991; Leonard & Jackson, 2015). It also uses the same operator set that consists of arithmetical operators as in Koza's study. SNGP is a form of GP, which uses graphs and dynamic programming methods that improve efficiency. Although the resulting RNG has some good results, it could not pass all of the NIST STS tests.

Picek et al. used Cartesian genetic programming method and generated a deterministic RNG function for lightweight devices (Picek, Sisejkovic, Rozic, Yang, Jakobovic & Mentens, 2016). The expressions are shown as an indexed graph in this method. The fitness function of this method uses the approximate entropy and avalanche criterion tests. This study has not got any test result although it generates the PRNG using a different GA approach.

Kösemen et al. worked on generating pseudorandom number generator by using genetic programming method (Kösemen et al., 2018). In their work, they used Shannon entropy calculation the fitness function of the genetic programming. Mathematical and logical operators were used to generate a PRNG. As a result, they created a PRNG that passes NIST tests and satisfies EPCGen2 standards.

In Lawnik's study (2018), using adequate chaotic transformation with uniform distribution, he advised a pseudo-random generating method. This method changes continuous distributions to uniform distributions by flattening, so it becomes possible to generate pseudo-random numbers by continuous distribution with the method. It uses the frequency of the occurrence of successive chaotic transformation's branches for flattening process. Standard normal distribution example is used to analyse this method.

Rose made an (2018) analysis about cryptographic quality of the KISS ('Keep it Simple Stupid') PRNG. Marsaglia and Zaman first specified KISS in 1993 (Marsaglia & Zaman, 1993) and Marsaglia published C code in 1998 (Marsaglia, 1999). Some authors argued that KISS PRNG is cryptographically secure, although Marsaglia himself never claimed this. Rose showed that KISS PRNG does not meet certain cryptographically secure PRNG criteria, demonstrating that the initial state of the KISS

PRNG can be recovered with 70 output words, and takes about 2 hours depending on the computer hardware. Rose also pointed out that Marsaglia's 2011 version of KISS is vulnerable to divide-and-conquer attack, so KISS is not suitable for applications needing cryptographically secure generator.

3.2 Authentication Methods

We can divide authentication protocols as Chien (2007) stated according to their computational cost and the operations they support. These groups can be listed as follows:

- Fully fledged protocols: This type of protocols support public and private key encryption, and a one-way function (Tuyls & Batina, 2006; Feldhofer, Dominikus & Wolkerstorfer, 2004).
- Simple protocols: This type of protocols support random number generator (RNG) and hash function (Tsudik, 2006; Weis, Sarma, Rivest & Engels, 2004).
- Lightweight protocols: This type of protocols involve random number generator and cyclic redundancy check (CRC) (Gilbert, Robshaw & Sibert, 2005; Bringer, Chabanne & Dottax, 2006; Duc, Park & Lee, 2006; Chien & Chen, 2007).
- Ultra-lightweight protocols: These protocols are suitable for lightweight devices that have limited hardware resources. They support simple bitwise operations such as OR, AND and XOR (Li & Deng, 2007; Li & Wang, 2007).

Security and privacy are so important requirements in wireless communication systems. If a unit wants to get data from server and send data to the server so first it should validate its identity. In that stage, there are so many security risks. Therefore it is important that the authentication protocols should address the following security services and resist most or all of listed security threats to create a secure communication and protect privacy and critical data.

- Denial-of-service (DoS) attack: DoS attack is a big problem for all systems and environments that can apply. For RFID systems, DoS attack is implemented by

sending too much data to the tags so the tags cannot operate properly. This attack can also cause de-synchronization problems between the tag and the reader.

- **Tracking attack:** In this case, the attacker violates the privacy of the RFID and gets a chance to track the tag and causes serious privacy loss for the users. Specific data or the ID and the owner would be easy to track. In this attack type, tag is monitored by an attacker without permission or knowledge of the authorized user or owner. To implement this attack, attacker has to have each tag's unique product code (EPC). To prevent from this attack, product code should not be transferred as a plaintext between the tag and the reader.

- **Desynchronization attack (Ahmadian, Salmasizadeh & Aref, 2013):** This attack is caused by blocking the communication between the reader and the tag so stored data or keys that may be synchronized won't be same. Moreover, one of the parts can't update shared updated values so the connection will be ended for the next session. It is necessary to make sure each part knows the updates.

- **Man-in-the-middle attack:** The attacker locates itself between the reader and the tag so gets messages from one party and repeats to other party. IoT makes it for both directions. Therefore, it will know all communication content and the reader and the tag won't be aware of this.

- **Impersonation attacks** aims the attacker to impersonate a trusted RFID in an attempt to gain access to critical or private data.

- **Cloning attack (Lehtonen, Ostojic, Ilic & Michahelles, 2009):** In this type of attack, the attacker is trying to trick the RFID reader into believing that they are exchanging data with a legal tag. This is done by creating a copy of an original tag by printing the necessary authentication information on the blank RFID tag. It is a process to make a clone copy of the tag. This attack is only possible if the attacker reads all secret fields of the tag. If some area is not accessible to the attacker so this attack can be blocked.

- **Full-disclosure attack (Shao-hui, Sujuan & Danwei, 2012):** The attacker can extract all secret information of the tags by using full-disclosure attack. The attacker

captures all data including secret ones. To prevent from this attack, data must be encrypted by a strong and secure cryptographic algorithm.

- **Eavesdropping:** It can be defined as listening a private connection channel between reader and tag illegitimately. This is communal issue of wireless communication.

- **Replay attack (Bolic, Simplot-Ryl & Stojmenovic, 2010):** The attacker listens the secure channel and gets transmitted messages between the reader and tag. If the attacker gets the authentication messages, it uses these messages and transmits it to reader to cheat the authentication process. The attacker aims to make its unauthorized tag to be authenticated illegitimately. This active attack is made against to a wireless communication. Output data of the RFID tag is recorded by the attacker and sends this data to the reader again. This attack can break the authentication between reader and tag or attacker can copy the messages and use them in another communication session. Against to this attack, a timestamp can be added to the messages or a random number can be used as a nonce.

- **Mutual authentication:** RFID systems are wireless networks. The first step in these systems is mutual authentication. The corresponding authentication process is also the first step in ensuring secure data communication. The RFID reader can get the content of the RFID tag after authentication process is successful.

- **Confidentiality:** This subject refers to the secure transmission of the confidential data. Therefore, it encrypts the tag's data and transmits it to the reader. Only the server can decrypt the encrypted text.

- **Availability:** The parties must be kept synchronized during and after the authentication process. This issue is very important to maintain communication.

- **Forward / backward security:** An attacker should not be able to obtain future data even though he has captured the previous or live private transmitted data.

- **Ownership transferable:** If an owner gives essential information to the new owner, the privacy of these two owners should not be violated in the new status, either.

- Tag anonymity: This is the most important security requirement for confidentiality. It means that the tag cannot be tracked. If this feature cannot be satisfied so the attacker who has compatible RFID reader can continuously track the owner of a particular tag.
- Traceability. An adversary can track a person's movement and personal information by identifying the RFID tag belongs to that person by using the appropriate RFID readers.
- Location privacy: The location of a tag should not be tracked and location confidentiality should be maintained. This feature blocks the tracking of an RFID tag location by an attacker.
- Information privacy: This means that the RFID content and transmitted information can be accessed by an authorized server or reader. Otherwise, the attacker can track the tag, clone privacy or be successful using other types of attacks.



Figure 3.2 A typical UHF RFID tag reading scenario (Özcanhan, Dalkılıç & Utku, 2013)

In Figure 3.2, we can see a typical UHF RFID tag reading scenario. This scenario shows us a simple authentication process steps in which 16-bit random number was used. It becomes a requirement to make changes on the authentication protocol shown in Figure 3.2 because of increasing processing power of tags which have EPC Gen 2 standard every day and addition of cryptographic algorithms to EPC Gen2 standard.

First part of this change is random number generation. Pseudo random number generation method is usually used for random number generation on RFID tags. This method occurs low randomness (Che, Deng, Tan & Wang, 2008). In this thesis, random numbers are generated from the hardware sources of the RFID tag.

Liao & Hsiao (2014) prepared an authentication protocol with an ID-verifier transfer protocol using the strong structure of the ECC. Then, some studies proved that their proposed protocol's performance was inefficient and it had security problems. (He, Kumar, Chilamkurti & Lee, 2014; Moosavi, Nigussie, Virtanen & Isoaho, 2014; Zhao, 2014).

Rathore and Hussain (Rathore & Hussain, 2015) proposed a secure, scalable, simple, dynamic, lightweight and efficient mutual authentication protocol for the nodes in WSN. Each node is a sensor and these nodes get a token from the base station (BS) so the protocol is based on these tokens. After getting the tokens nodes authenticate the other nodes using their tokens without communicating BS. This protocol was evaluated on scyther tool theoretically. They said that no potential attacks were detected. The protocol was not modelled or tested on a real hardware.

Rostampour et al. (Rostampour, Bagheri, Hosseinzadeh & Khademzadeh, 2016) focused on creating an approach to identify a set of object tagged using RFID technology simultaneously. This approach is named as RFID grouping proof authentication protocol. They used an encryption method called authenticated encryption and created a new authentication method that is scalable and secure by using authenticated encryption. They claimed that this protocol provides message integrity and confidentiality by not exceeding resource constraints for RFID. In the paper, they have done security analysis theoretically.

İbrahim and Dalkılıç (İbrahim & Dalkılıç, 2017) proposed a new mutual authentication protocol based on advanced encryption standard (AES) and elliptic curve cryptography (ECC) on the area of healthcare systems. The protocol was designed and implemented on WISP. As a random number generator, they used WISP's built-in PRNG. The new mutual authentication protocol cannot satisfy

resource usage limits because of the ECC' needs and also they did not give any structural and statistical analysis about the PRNG.

Zhou et al. (Zhou, Liu, Liu & Wang, 2017) proposed a secure and scalable grouping proof protocol with an encryption method for low-power and low-cost devices. It works in two stages; in the first stage it checks the group of the tag, in the second stage it verifies the tag's id. Encryption in this protocol is used for confidentiality and message integrity. The ECC is used as the encryption method. This protocol also uses offline approach.

Liew et al. demonstrated a new authentication protocol using grouping proof to minimize the theft of the cargo that includes the tagged items while transport (Liew, Tsai, Luo & Yang, 2017). This protocol provides the integrity of all items in the cargo and transferability of the ownership of the tagged item. Security analysis of the proposed protocol has been made, theoretically. The protocol was not run on a real RFID system. It is also observed that the implementation of the protocol has not been examined in terms of time and resource usage, as well as the fact that they have not performed the realization.

Zhang et al. (Zhang, Qin, Wang, Wu & Yi, 2018) proposed an RFID grouping proof protocol that is scalable and lightweight. The RFID grouping proof protocol confirms that a group of tags belong to the same group at the same time. This protocol uses exclusive-or (xor) operations and pseudorandom number generation which makes it suitable for large scale grouping proof of lightweight devices. Firstly, the reader broadcasts a request message and all tags around it respond with response messages. After the tag responses, reader generates a grouping proof based on the responses. They used an anti-collision algorithm to identify the response messages.

Zhou et al. (Zhou, Liu, Liu & Wang, 2018) proposed an ECC-based offline grouping proof protocol. The reader is authorized to validate the tags without knowing the identities of them. This approach is claimed to protect the privacy and the security of the tags, and to defence against replay and impersonation attacks. In offline mode, a continuous connection between the reader and the verifier is not needed, so the reader

can do the grouping proof operation without the verifier. After proofing, the reader sends the data to the verifier so it makes the connection requirement more flexible.

Lee et al. (Lee, Chen, Li, Cheng & Lai, 2019) recommended a cloud-based RFID ownership transfer protocol that meets all major security requirements, including forward / backward confidentiality, location and data confidentiality, mutual authentication, ownership integrity and privacy. They claimed that their protocol's communication cost is low. They showed that it is logically correct and provides security and privacy protection considering cost and performance.

3.3 Other Security and Privacy Enhancing Methods in RFID

Because of having many advantages in Auto-ID systems, RFID technology is exposed to attacks at different points. One of the biggest threats in wireless systems is the security and privacy of the data. Personal and object-related information may be targeted by some threats. These threats can be listed as: Unauthorised access, data tampering, tag spoofing, deactivation of tags and removal of tags, jamming, blocking, reader spoofing, eaves dropping (listening to the communication between the reader and the tag without permission), tracking, relay and reply attacks, RFID-Malware, (Bayrak Meydanoğlu, 2008) denial of service.

In the literature, there are many security enhancing methods. Solutions such as physical mechanism, access control, authentication protocol and encryption algorithm are the recommended strengthening security protection methods in the literature. Message authentication code (MAC), cyclic redundancy code (CRC) and pseudo-random number generator (PRNG) is suitable for lightweight systems. Symmetric encryption algorithms are mainly used by middleweight systems.

Mitra proposed a new scalable security algorithm. She claimed that it resists the tracking and clonning attacks. The proposed algorithm is not implemented. The strength against other known attacks has not been tested. The parameters such as the power usage and the speed of execution are not mentioned (Mitra, 2008).

In a US patent numbered US20070289775A1, a wallet design has been proposed to prevent unauthorized access to RFID tags integrated with credit cards, personal

identification cards and other electronic cards (Potts, Shore & Wood, 2007). Moreover, Shoarinejad and Soltan (2009) invented another patent numbered US7952481B2 and titled “Systems and methods for RFID security”. They suggested a system where the transmitted information was encrypted using a synchronized key.

The use of RFID technology is becoming widespread. Consumers using RFID technology are reluctant due to various concerns. There are serious question marks, especially regarding privacy. Compatible readers can read RFID tags that carry unique and personal data of the consumers. Consumers are also concerned that their shopping, travelling and other activities can be tracked (Glasser, Goodman & Einspruch, 2007).

Anonymity and privacy are two basic requirements for RFID systems. If the RFID tag on the product purchased from a shopping store has not been killed, the threat of confidentiality can occur. Due to this tag, person who has compatible reader can track what the customer buys. The RFID tags used in the Smart card also bear serious risks. Badges given to participants at a conference are examples of privacy risks (Labay & Anderson, 2006).

RFID tags can be cloned, exchanged, imitated, deliberately disabled, damaged or impaired. RFID technology may be compromised when used in an insecure system. If RFID tags provide communication with weak encrypted or unencrypted data, the data can be accessed publicly. This occurs serious problems in sensitive areas such as health, logistics and etc. In addition, server-side databases are very important. Confidentiality violations to the database from outside or inside of the server are a major risk (Lockton & Rosenberg, 2006). Unauthorized users can access critical data and this will cause leakage or abuse of information in the database. The database must be correct, steady and up-to-date distributed so that the system can be accessible and survivable. Reliable systems can be created by special requirements, appropriate security, binding integrity, privacy-protecting encryption and etc. (Neumann & Weinstein, 2006).

Some European scientist declared that there is a serious risk about the infection of viruses to the RFID tags in some of their research (So & Liu, 2006). This can cause major problems at supermarkets and airports. In that researches, they gave information

about numerous types of exploits. RFID tags can execute exploits such as malicious code insertion, buffer overflows and SQL injection to the RFID middleware (Alder, 1998).

Juels (2006) declared that a serious privacy risk arises when an RFID tag and personal information is matched. For example, one can define shopping habits and categorize individuals by looking at their purchases with credit card. Garfinkel et al. (2005) overviewed the potential threats and they tried to address for both public policy and technology (Garfinkel, Juels & Pappu, 2005). On the other hand, Yamamoto et al. (2008) proposed a new tamper detection method. It can be used without or with current locking mechanisms (Yamamoto, Suzuki, Hada, Mitsugi, Teraoka & Nakamura, 2008).

CHAPTER FOUR

PROPOSED SECURITY ENHANCING METHODS FOR RFID

This section describes the security enhancements for the RFID. Used materials, methods and process steps are explained. A brief overview is given in the first title. Then, the generated TRNGs using hardware resources on the WISP RFID tag are described. These hardware sources are temperature and accelerometer sensors, DCO-VLO clocks and ADC. Two different methods were used to generate TRNG with accelerometer sensor unlike the other hardware resources. Then, developed three new PRNGs were explained. These PRNGs are MLFSR, xorshiftR+ and xorshiftL+. Moreover, we try to use genetic algorithm to generate random numbers and pseudorandom number generators. Finally, the proposed grouping proof authentication protocol is described.

4.1 Proposed True Random Number Generators

As it is mentioned before, WISP RFID tag's hardware sources were used to generate random numbers. Under this section the methods how we created random numbers using these hardware sources were explained.

4.1.1 True Random Number Generation by Using Analog to Digital Converter

As seen in Figure 4.1, a 16-bit real random number (GRS) production was performed by sampling voltage over ADC for 16 times and the least significant bits of each of these samples were taken as a bit. At each step, the number was shifted to the left once. You can see the source code for sampling the voltage over the ADC in Appendix 1. This code shows a single step of the sampling process that is repeated 16 times. 1,000,000 random numbers were generated using this method.

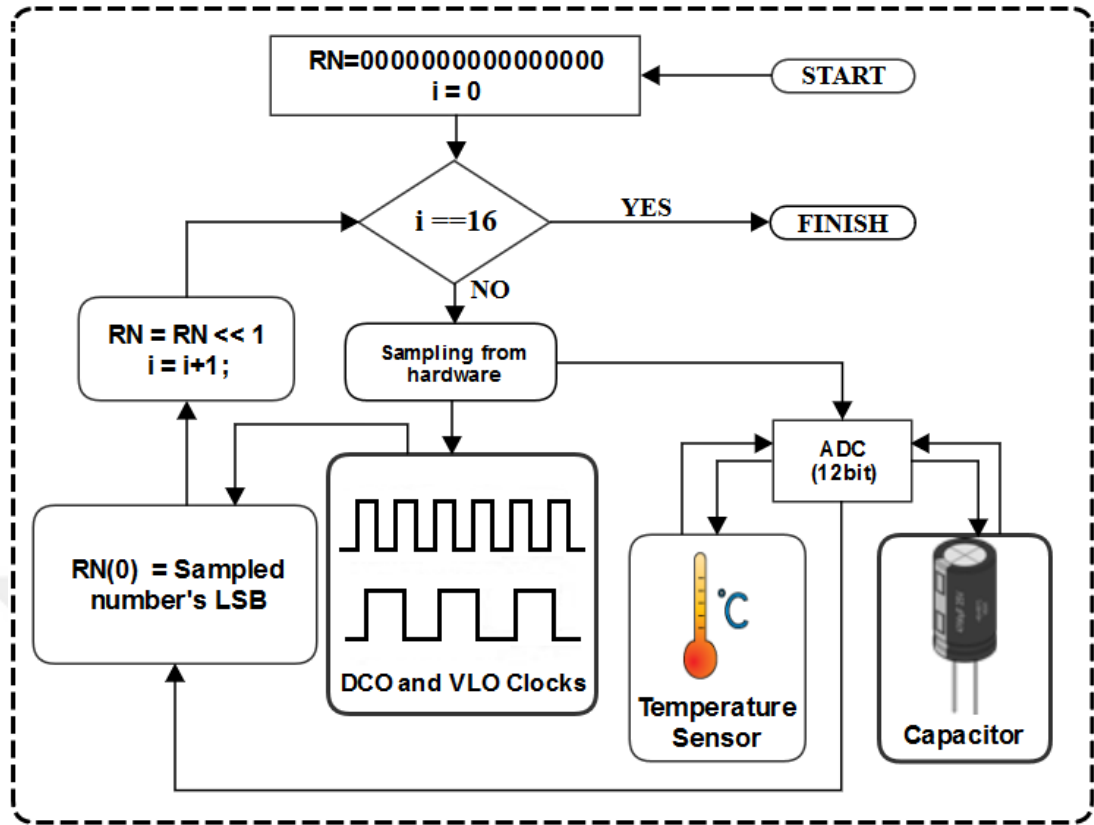


Figure 4.1 Flow chart for true random number generation by using temperature sensor, voltage sampling and difference between clock pulses (Dalkılıç, 2016)

4.1.2 True Random Number Generation by Using Temperature Sensor

Sampling was performed from the temperature sensor on WISP by the C code given in Appendix 2 loaded on WISP. The least significant bit of the sampled data was used for random number generation. A 16-bit random number was generated as a result of 16 samplings. After each sampling, the number is logically shifted to the left. As a result of the sampling, 1,000,000 random numbers were generated with using the data obtained from the temperature sensor. The process flow chart for one step of this process can be seen in Figure 4.1.

4.1.3 True Random Number Generation by Using Difference Between Clock Pulses

As mentioned earlier, there are two clocks running on different frequencies called SMCLK (which uses DCO source) and ACLK (uses VLO source) on WISP. From these WISP clocks, SMCLK (DCO) can be set to 1, 8, 12 and 16 MHz frequencies. On the other hand, ACLK (VLO) can be set and operated by 12 KHz frequency. The

MSP430FR5969 microcontroller on WISP 5.1 has all registers for these operations. Due to the difference between these two clocks, real random number generation is performed (Fujdiak et al., 2013b). In Figure 4.1, implemented TRNG's flow chart can be seen and C code was given in Appendix 3. In this way, 1.000.000 numbers were generated.

4.1.4 True Random Number Generation by Using Accelerometer

One of the sensors on the WISP 5.1 is the accelerometer sensor. The accelerometer ADXL362 on the WISP is a very low energy accelerometer with 3 axes less than 2 μ A and it has an integrated temperature sensor (ADXL362, 2016).

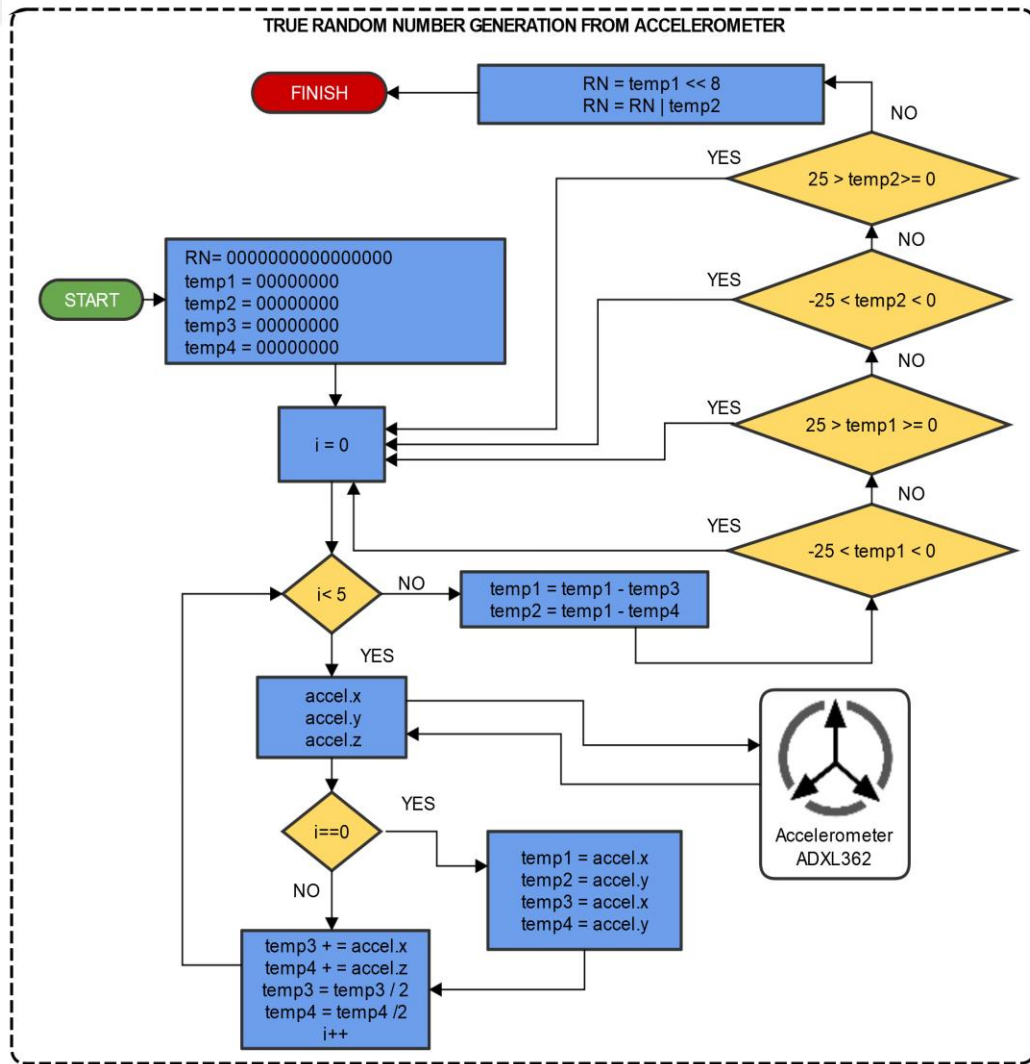


Figure 4.2 Flow chart for true random number generation by using accelerometer (1st method) (Dalkılıç, 2016)

The random number generation using accelerometer sensor is done by a different method compared with the random number generation process using other sensors. In cases where WISP is stationary, some mathematical operations have been performed in order to produce different numbers in each sampling and the results are used in random number generation. The steps of the generation are shown in detail in Figure 4.2 and the C code of these processes is given in Appendix 4.

As shown in Figure 4.2, the values of x, y and z were taken from the accelerometer. These values were taken 5 times. Values are assigned to variables named temp1, temp2, temp3 and temp4 and mathematical operations are performed on each variable as shown in Figure 4.2. After this process, temp1 and temp3 variables were used to obtain temp1. temp2 and temp4 variables were used to obtain temp2 variable. If the values of temp1 and temp2 are in the range of -25 and 25, all operations are repeated from the beginning. If it is not within this range, the last 8 bits of temp1 are shifted to the left and replaced with the last 8 bits of temp2. As a result of this process, a 16-bit random number was generated using 3 values from the accelerometer. Since WISP is predicted to remain stationary, 5 cycles were performed. If the difference in the difference is between -25 and +25, it is observed that the random numbers are repeated and therefore the number of repetition is assumed to be not large enough. If both temp1 and temp2 is not between -25 and 25 at the same time, these numbers can be taken to calculate random number. 1,000,000 random numbers were generated using the process steps in Figure 4.2 (Dalkılıç, 2016).

After trying the method given in Figure 4.2, it was concluded that it would be more logical to make changes in the real random number algorithm generated from the accelerometer. The algorithm of the accelerometer is modified as shown in Figure 4.3. The lowest significant bit was used for this process, as in other generators. The 16-bit random number was generated using the lowest significant bit of the x, y and z coordinate values obtained from the accelerometer, respectively. To do this, the C code has been updated as in Appendix 5.

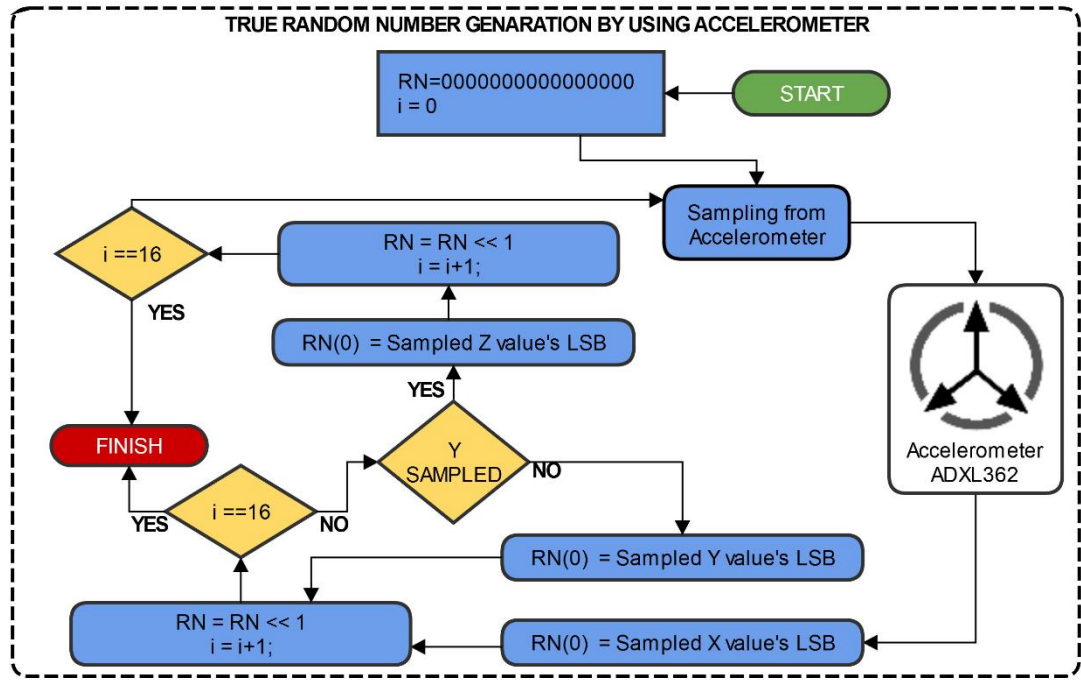


Figure 4.3 Flow chart for true random number generation by using accelerometer (2nd method) (Dalkılıç, 2016)

4.2 Merged Linear Feedback Shift Register

In the linear feedback shift register, the newly generated results, or in other words, the new state, are obtained by a linear function of the previous state. An overview of the LFSR is shown in Figure 4.4. When general use is observed, the system operates in the form of writing the new state to the registers by subjecting the bits of the previous state into the XOR operations of the registers. Since the utilizing process here is a linear function, the entire series of numbers produced is also linear. In addition, the series of generated numbers are often in a limited range so same number series can be generated. On the other hand, the randomness of the numbers can be increased and the repeat cycle can be extended by a well-designed function.

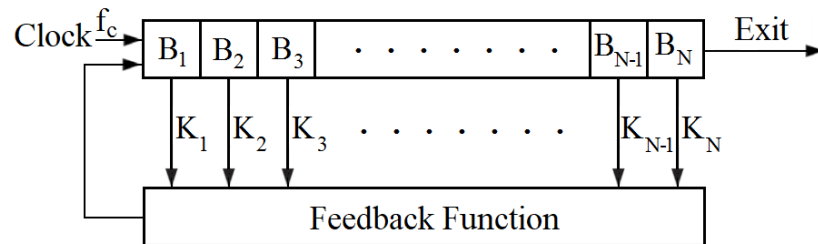


Figure 4.4 LFSR general structure (Schneier & Sutherland, 1995)

Because of limited resources in Internet of Things devices such as radio frequency identification tags, all kinds of software are designed with attention to these limits. Security is also affected by the lack of resources. Since the inadequacy of resources may lead to the failure to provide secure transmission of information over the network, it is necessary to provide the best security with the best available resources. Therefore, the use of well-designed LFSR with high performance and low resource consumption in IoT devices will provide significant benefits.

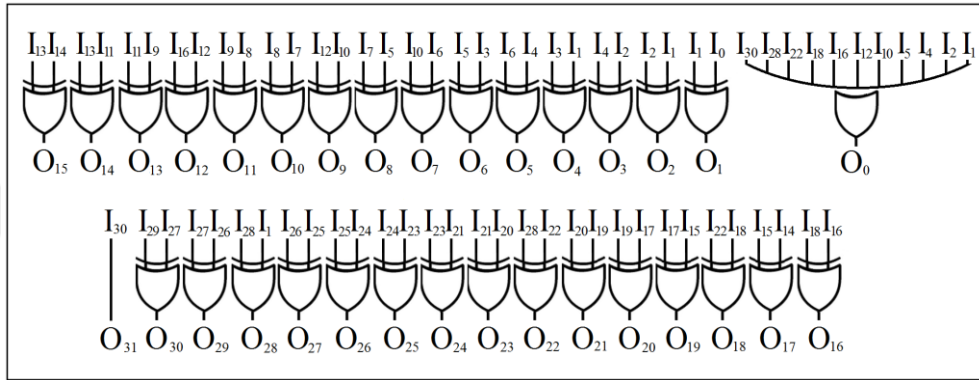


Figure 4.5 MLFSR input and outputs (Aydın & Dalkılıç, 2016)

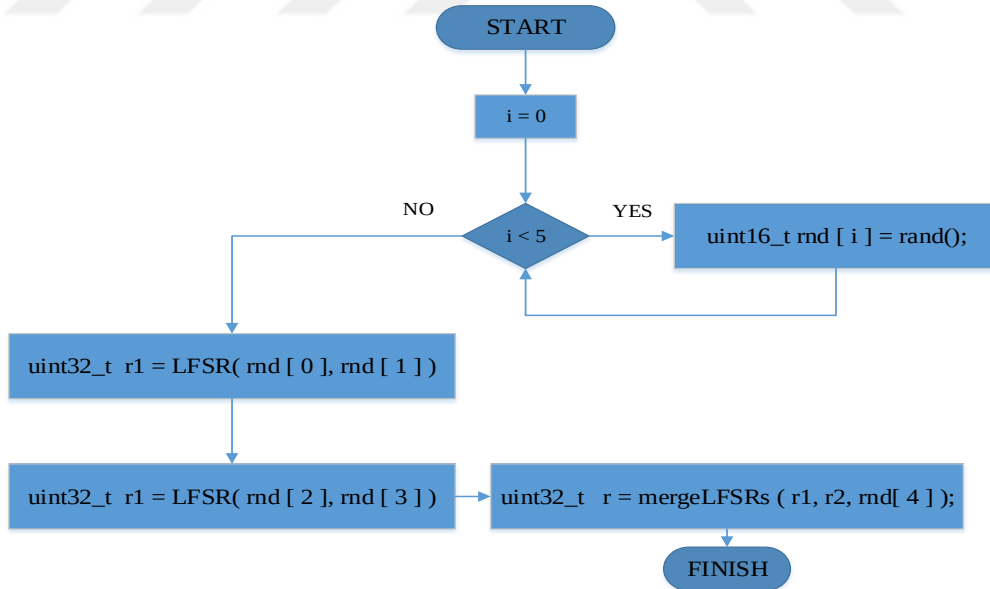


Figure 4.6 MLFSR flow chart (Aydın & Dalkılıç, 2016)

Here to create a merged LFSR, a 32-bit linear feedback shift register is used. The two 16-bit numbers generated by the C rand() function outputs are given to this

register. LFSR generated a 32-bit number from these two 16-bit numbers. The input and the outputs of the linear feedback shift register are detailed in Figure 4.5.

The process shown in Figure 4.5 was repeated twice as shown in the flow chart given in Figure 4.6. As a result, two 32-bit numbers were generated (r1, r2). These two 32-bit numbers obtained on the LFSR were merged in a way, resulting in a 32-bit random number. The merge operation's C code is given in Figure 4.7. The 32-bit number obtained by this operation will be used as a random number.

```
uint32_t mergeLFSRs(uint32_t r1, uint32_t r2, uint16_t rnd) {  
    uint32_t r = rnd;  
    result = (r1 ^ rnd) ^ ((r << 16) ^ r2);  
    return r;  
}
```

Figure 4.7 Merge function C code for LFSRs (Aydın & Dalkılıç, 2016)

4.3 Modified Xorshift+ Algorithm: XorshiftR+

Since the computational limitations may be very stringent on lightweight devices, many of the complex algorithms considered as successful in related works are not considered as eligible for this purpose. Xorshift+, a recent algorithm of the known xorshift family is in the focus with its simple structure (Vigna, 2017). It is also a main part of the study to force this algorithm to run on a variety of input and output sizes. Moreover, the original xorshift+ algorithm is manipulated by removing some last steps to make it even more compact O'Neill also mentioned this reduction idea (O'Neill, 2014), though not tested systematically.

Within this work, we have made random scrambling (based on our predictions) of the original xorshift+ and produced many generators with slight differences, in order to derive a better xorshift RNG. Nevertheless, we have eliminated many of these, which are considerably weak or do not sufficiently mitigate the complexity. Our efforts to find a suitable reduced xorshift+ version (which is to be named as xorshiftR+, where R stands for reduced) resulted these 3 candidate algorithms (later called as variations or shortly var), given as follows (in C notation);

```

1  uint64_t s[2]; // seeds
2  uint64_t xorshift128plus(void) {
3      uint64_t x = s[0];
4      uint64_t const y = s[1];
5      s[0] = y;
6      x ^ = x << 23; // a, shift & xor
7      x ^ = x >> 17; // b, shift & xor
8      //-----
9  Var1: x ^ = y ^ (y >> 26); // c, xor
10     s[1] = x;
11     return x + y;
12     //-----
13  Var2: x ^ = y ^ (y >> 26); // c, xor
14     s[1] = x + y;
15     return x + y;
16     //-----
17  Var3: x ^ = y ^ (y >> 26); // c, xor
18     s[1] = x + y;
19     return x + y; }

```

The body of the code above (from beginning, until the first dashed line) is untouched and common for all variations (in fact step c is same for all, too), while the struck code snippets are removals from and the black-highlighted parts are additions to the original code of the xorshift+. The variations given under three consequent tags are applied singly to the body of the code, so that a full RNG algorithm is obtained.

Ending variation 1 excludes 26 right shifts and 1 xor operation per generated random number. Ending variation 2 excludes 26 right shifts and 1 xor but adds 1 addition. Lastly, variation 3 excludes 26 right shifts and 1 xor, removes 1 addition and adds 1 addition, which makes it identical with variation 1 in terms of operation count. These manipulations supposedly produce a big difference in applications where many numbers are generated sequentially. However, from the codes themselves, it is not possible to accurately estimate the time gain without experiments. That's because the run-time highly depends on the processor type (Singh & Kumar, 2012). Thus, we make use of our test completion times to measure the algorithms' performance on a lightweight IoT device, given in chapter 5.

Memory gain, on the other hand, is easier to calculate or estimate. The memory usage of generators is mostly dependent on the period of their output and the number of their input (seed) states. These two factors are determinative of the number and the size of the variables used in their source code. To make a standard, we fixed the integer types to 32 bits as mentioned above. But please note, the original xorshift+ (and our reduced variations) uses two seed inputs which makes it occupying 64 bits of input memory, while most RNGs use only one, so it is implanted using 64 bits of seed state. Nevertheless, because of its very poor performance, we didn't use 32-bit seed/input version of the linear feedback shift register (LFSR), but 128-bit version instead. The output kept as 32-bit to fix the output sizes of all subjects. In order to see the effect of the output or the period of the RNGs on their randomness scores, please refer to another paper (Brent, 2007).

Other than our xorshift+ variations, we also tested an earlier xorshift based algorithm, namely xorshift64* (Matsumoto & Nishimura, 1998), a simple linear congruential generator (LCG) and an LFSR implementation to compare the results and estimate the benefits of using each RNG. XSadd is intentionally excluded, since it is already superseded by xorshift+. A comparative breakdown is provided for a brand new xorshift based RNG, called xoroshiro128+ that is published at the time of writing of this paper. LCG is included because of its popularity, even though not recommended as a reliable RNG anymore (Hellekalek, 1998; O'Neill, 2014). It is based on the one implemented in TestU01 suite as "CreateLCG()". The LFSR is included because of its different bit-level structure and high speed nature. It is also based on the one introduced in the suite with the name "CreateLFSR113()".

In fact, we had momentous efforts to functionalize DES as in such a purpose as recommended in RFC 1750 (Eastlake et al., 1994) with relevant modifications. Nevertheless, later we discarded it during our pre-evaluation phase because of its extremely poor performance results. It wasn't even close to be "lightweight" when compared to other RNGs presented here. Information that explains why and how much DES, advanced encryption standard (AES) and derivatives are slower than these lightweight algorithms can be found elsewhere (Thakur & Kumar, 2011).

4.4 Lightweight Version of the Xorshift Algorithm: XorshiftL+

Our proposed HRNG (Aydın & Dalkılıç, 2018) has a PRNG that is a member of “Mersenne Twister” (MT) (Matsumoto & Nishimura, 1998), well-known feedback shift register superclass, and a TRNG which uses temperature sensor as the hardware source.

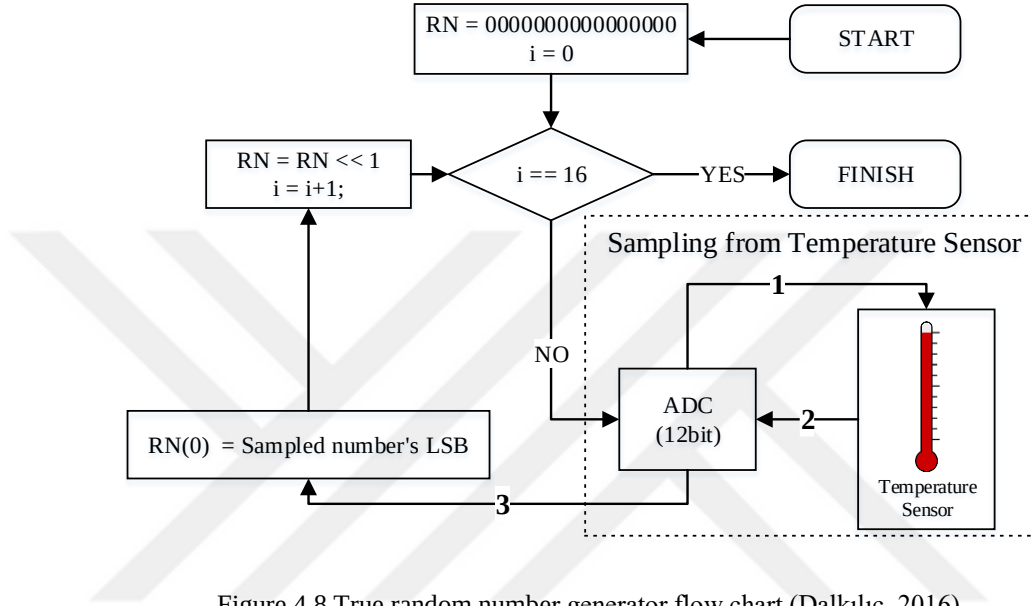


Figure 4.8 True random number generator flow chart (Dalkılıç, 2016)

Figure 4.8 shows the 16-bit true random number generation flow chart with a temperature sensor, and an analog to digital converter (ADC). The ADC produces 12 bit values. The least significant bit (LSB) of the 12 bit values is set as the random number's first bit, and the random number is shifted logically to the left. This process continues for 16 samplings, thus generating a 16-bit true random number.

```

int32_t x = TRNG ();
int32_t xorshiftLplus (int32_t s) {
    int32_t y = s + x;
    x = x ^ (x << 3);
    x = x ^ (s) ^ (x >> 5) ^ (x >> 2);
    return y;
}

```

Figure 4.9 HRNG C code implementation

In the scope of our study, we changed the original xorshift+ by random scrambling made by our predictions. We generated many PRNGs with fewer shift operations and

shorter seeds compared with the original xorshift+ and selected the best NIST results of the PRNGs. This newly created HRNG produces 32 bit random numbers. C code implementation of the HRNG tested on WISP RFID is shown in Figure 4.9.

4.5 Genetic Algorithm and Random Number Generation

True random numbers which are generated from hardware sources will be used to generate seeds and genetic algorithm's next generation. The aim of this process is selecting random numbers and production of a genetic algorithm output that passes randomness tests. You can see genetic algorithm's flow chart in Figure 4.10.

Genetic algorithms (GA) starts with an initial population which consists of randomly generated individuals. Genetic Algorithm's most important step is selecting suitable fitness function. Fitness function will be select by following the steps below (Mishra & Bali, 2013):

Firstly, Shannon Entropy value ($H(X)$) will be estimated (Shannon, 1948).

$$H(X) = -(p * (\log_2(p))) - ((1-p) * (\log_2(1-p)))$$

Chi-Square (Chi-Square - χ^2) value will be estimated by using Shannon Entropy value.

$$\chi^2 = (\text{observed_}H(X) - \text{expected_}H(X))^2 / (\text{expected_}H(X))$$

Autocorrelation coefficient will be estimated by Chi-Square (Chi-Square - χ^2) value

$$\phi^2 = \chi^2 / \text{number of sampling}$$

Autocorrelation coefficient will be used as fitness function's result value.

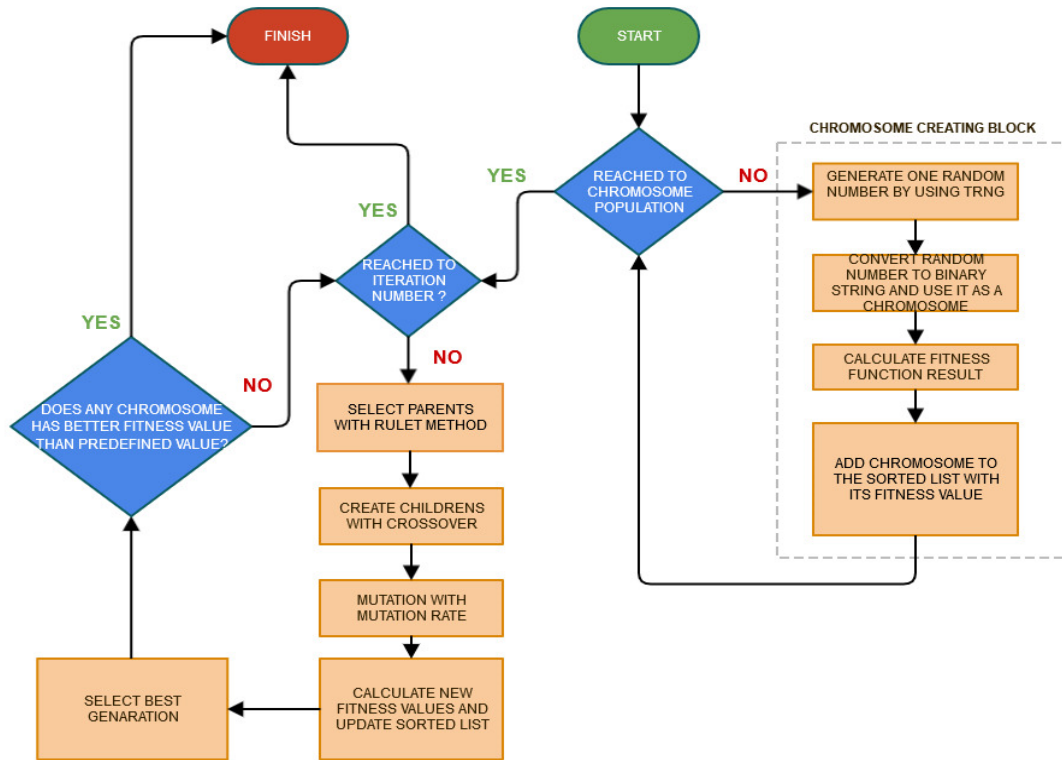


Figure 4.10 Flow diagram of genetic algorithm

At first, numbers were tried to be generated by using the method shown in Figure 4.10. We decided to change this method as a result of the failed NIST test results given in the chapter 5. In second method, each individual is selected as a mathematical prefix expression.

Population size is selected as 50. The bigger population sizes do not affect the solution according to our experiments. For generating the initial population, an operator set must be determined. The operator set used is;

$\{ \&, |, ^, +, !, > \}$

They are consecutively; bitwise AND, bitwise OR, bitwise XOR, addition, unary bitwise NOT and left rotation operators. Multiplication, division, modulus and etc. mathematical operators cost much for lightweight devices such as WISP, so they are not appropriate to be used.

There are also integer numbers and input parameters which are used in expressions. Input is indicated with letter K. Integer numbers are selected between 1 and 16.

The first step in the genetic algorithm is tournament selection (Miller & Goldberg, 1995). Tournament selection randomly selects a number of individuals, defined as 5 in this study and chooses the best one. After two tournament selection operation is applied to the population, two individuals are gathered.

These two selected individuals with tournament selection are the inputs of the single-point crossover operation. For single-point crossover, that two expressions are changed parts from one randomly selected operator node. The fittest of the new two individuals is selected as a result. Only selected operator nodes are swapped between expressions so, total token count does not change. Assuming these two input expressions like below;

$\wedge K > \wedge ! K 4 13$

$\wedge K + > K 7 | > ! K 10 5$

After crossover these two expression might be like this;

$\wedge K > + > K 7 | > ! K 10 5 13$

$\wedge K \wedge ! K 4$

Or this;

$> K 7$

$\wedge K + \wedge K > \wedge ! K 4 13 | > ! K 10 5$

Maximum token count is limited as 80. That means, there cannot be an expression which have more than 80 tokens as a result of a crossover operation.

After crossover operation returns a new individual, mutation operation is applied to it. Only one operator of the expression is being changed by mutation operation with a chance of 5%. For example, if a binary operator is selected as a target of mutation operation, it is replaced with a new randomly selected binary operator. Finally, this individual is become a new member for next population.

Also, fittest individual is preserved and remains unchanged in the next generation. This method is known as elitism in GA.

Fitness function determines how good the solution is. In that case, individual with the best entropy value considered as fittest. Shannon entropy method is used for calculating entropy of generated binary sequences (Shannon, 1948). To encode the data, it basically calculates minimum number of symbols like in data compression.

Genetic Algorithm's most important step is selecting suitable fitness function. Fitness function will be select by following the steps below:

H value is separately calculated for every n -bit entropy value. b is 2 because data is encoded into binary sequences.

In this work, 9 can be the maximum entropy value for an individual because 1 to 8 bit entropies are calculated.

In order to calculate the suitability of an individual, a seed value is given to it and 2^{18} numbers are generated. The expression's input and output values are signed and 64 bit integers. Output value is the next input seed of the PRNG. The entropy of the 2^{18} integers is calculated for two different seeds. Then the individual's fitness value is determined. Best expression produced with this experiment is Eq. (4.1) in prefix form (Kösemen et al., 2018);

$$+ K + > + ! K 8 16 ! > K 13 \quad (4.1)$$

This expression has a fitness value of 17.99429. It takes 11 generations to obtain the solution. C code of the generated PRNG is given in Figure 4.11.

```
int64_t a1 = 0;
int64_t K;
void createRngNext() {
    int64_t a1 = (~K + 8);
    K = (K + (((a1 >> 16) | (a1 << 48)) + (~((K >> 13) | (K <<
        51)))));
}
```

Figure 4.11 C code for generated PRNG by GA (Kösemen et al., 2018)

4.6 Grouping Proof Authentication Protocol Using XorshifL+

In this section, a new grouping proof authentication protocol is explained. Figure 4.12 shows the components of the RFID system implemented by the grouping proof authentication protocol developed within the scope of this thesis. Figure 4.13 shows the protocol and Table 4.1 includes all the notations used in the protocol. The protocol developed in this study consists of three phases detailed in subsections.

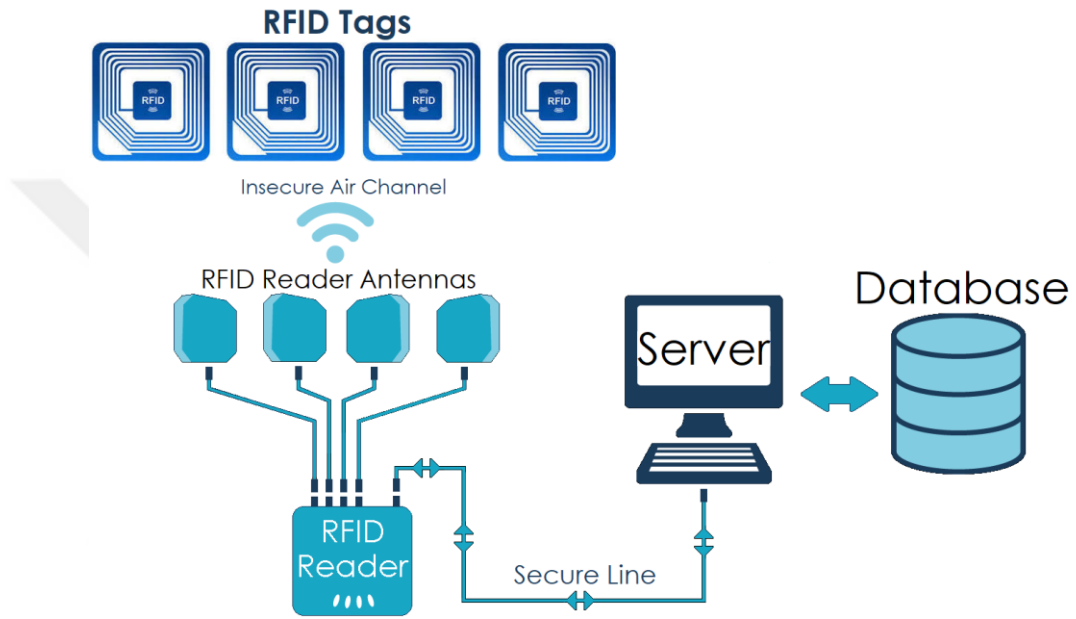


Figure 4.12 RFID system components with group authentication protocol applied

Table 4.1 Notations

k_g	A secret key for the group members
k_t	A secret key for the tag
ID_g	An identification code for the group
ID_t	An identification code for the tag
N_t	A random number generated by tag
N_r	A random number generated by RFID reader
MAC	Message Authentication Code
$\oplus$	Exclusive or operation
$E(X, k)$	X encrypted with key k using AES
$D(E, k)$	E decrypted with key k using AES
m	The number of the tags in each group
$M2^i$	Transmitted M2
ID_g^i	Calculated ID_g (After Decryption)
$M4^i$	Calculated M4 (After Decryption)
$M3^i$	Calculated M3

RFID tags are located on the objects such as pallets, cases. Data communication between the RFID tags and the reader is on insecure air channel. The communication between the RFID reader and the server is assumed to be a secure connection. The database can be located on the same server, or on a separated server, and this communication line should also be secured.

4.6.1 Registration Phase

At this stage, the server initiates the reader, the tags and the groups with necessary records. There exists a record for each group in the server's database since the GPAPXL+ protocol is based on the grouping proof method.

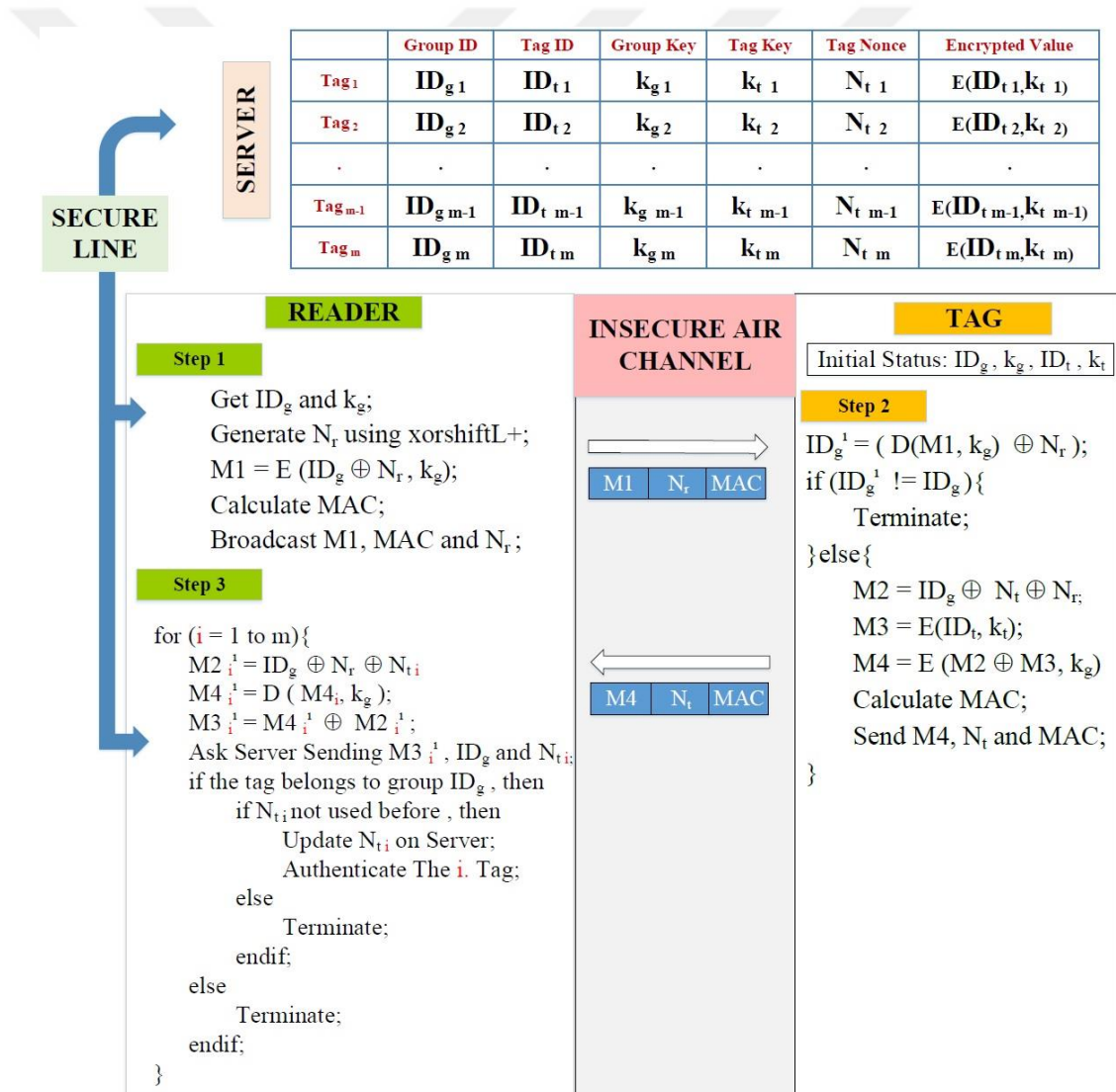


Figure 4.13 Group authentication protocol steps

Two encryption keys, group key (k_g) and tag key (k_t) were set in each RFID tag. The server database contains a record for each tag. During the registration phase, the system behaves as follows: This entry contains ID_t , k_t , ID_g , k_g , N_t and $E (ID_t, k_t)$. In addition, each RFID tag stores ID_t , k_t , ID_g and k_g . Figure 4.13 shows the functioning of the grouping proof authentication protocol between an RFID tag, a reader, and a server that includes a database.

4.6.2 Authentication Phase

In this phase, the tags of the desired group were authenticated by running the following protocol steps below.

- Step 1

The server sends the required information to the RFID reader for the requested group.

The reader runs xorshiftL+ algorithm and generates 8 random numbers ($8 \times 32 = 256$ bit) and concatenates them to form the nonce value. This nonce value is added to the message package to be sent to the RFID reader.

The ID of the desired group and N_r are being exposed to XOR operation and then the result is encrypted with k_g (desired group key) by using AES-256 and $M1$ is generated as shown in Eq. (4.2).

$$M1 = E (ID_g \oplus N_r, k_g) \quad (4.2)$$

Reader concatenates $M1$, N_r and MAC, then broadcasts this information to all tags in its active field.

- Step 2

After receiving the encrypted message, each tag decrypts message $M1$ using k_g , the result and transmitted N_r is being exposed to XOR operation and finds the calculated group number from the message, called ID_g' . It compares the group id with the previously stored group id on its own, and a match shows that the tag is a member of

the desired group. Otherwise, it is verified that the tag does not belong to the group being inquired by the reader, and the communication is interrupted at this point.

If the group id matches, the tag generates a random number (N_t) using our new algorithm xorshiftL+ as a nonce value. On the tag side, first zeros are appended to the 32-bit random number for the missing bits of 32 bits and XOR operation is proceeded by matching two numbers bit by bit. M2 and M3 are as in Eq. (4.3) and Eq. (4.4):

$$M2 = ID_g \oplus N_t \oplus N_r \quad (4.3)$$

$$M3 = E(ID_t, k_t) \quad (4.4)$$

M2 and M3 are encrypted with (k_g) after being exposed to XOR operation shown as in Eq. (4.5).

$$M4 = E (M2 \oplus M3, k_g) \quad (4.5)$$

MAC, N_t and M4 are concatenated and sent back to the reader as a response message.

- Step 3

The reader always waits for tag responses. If the response is received, reader calculates M2', M4' and M3' respectively as in Eq. (4.6), Eq. (4.7), Eq. (4.8) for each tag response separately.

$$M2' = ID_g \oplus N_r \oplus N_t \quad (4.6)$$

$$M4' = D (M4, k_g); \quad (4.7)$$

$$M3' = M4' \oplus M2'; \quad (4.8)$$

Then it sends M3' along with ID_g , and N_t to the server and waits for a response from the server.

4.6.3 Verification Phase

The server retrieves $M3'$ sent from the reader and searches the database. It compares $M3'$ with $E(ID_t, k_t)$ encrypted text on the database, and, if it finds a match, compares the N_t value sent by the reader with the N_t value in the corresponding row in the database. If there is a match in N_t , the server returns a replay attack warning, reporting that the nonce value was previously used. When $M3$ values are matched and if there is a new nonce value, the existing N_t value in the database is updated with the new value from the reader, and the values such as ID_t, k_t of the corresponding record are returned to the reader.



CHAPTER FIVE

TESTS AND EXPERIMENTAL RESULTS

This section gives the test, experimental results and analyses of the proposed TRNGs, PRNGs and authentication protocol. In the first section, test results of the proposed TRNGs were given. In addition, test and experimental results for the new proposed PRNGs (MLFSR, xorshiftR+, xorshiftL+), xorshift+, xorshift*, reduced versions of xorshift+, simple LCG, LFSR113 and xoroshiro+ were given. For the tests, ENT (Walker, 2008), Diehard (Marsaglia, 1995), NIST (Bassham et al., 2010) and TestU01 test tools were used. Moreover, NIST test results for the random numbers and pseudorandom number generators generated by genetic algorithm are presented. Finally, test results and obtained analysis of the proposed grouping proof authentication protocol is given.

5.1 Proposed Random Number Generators

Under this title, we gave the ENT, Diehard and NIST test results of the proposed TRNGs and PRNGS. Firstly, we consider four hardware sources to check if they are successful in ENT and Diehard test suites. Then, we gave the NIST test results of the each TRNGs using WISP's hardware sources. At the end, we gave test results for MLFSR, xorshift+, xorshift*, simple LCG, LFSR113, xoroshiro+ and reduced versions of the xorshift+ (Reduced xorshift+ var3 ($s[1] = x + y$, return x) is named as xorshiftR+), xorshiftL+ in which "L" stands for "lightweight" and genetic algorithm generated numbers and a PRNG.

5.1.1 ENT and Diehard Test Results of the Random Numbers Generated from Four Different Sources

When the Ent test results in Table 5.1 are examined, it is seen that all the sources produce statistically bad results. The results indicate that the randomness of the numbers generated from that sources is low.

Table 5.1 ENT test results of the true random numbers generated from four different sources

Statistical Test	Accelerometer	Dco-Vlo	Temperature sensor	ADC Sampling
Serial Correlation Coefficient (Byte)	0.144813	0.136734	-0.016667	-0.003799
Chi-square statistics	134.54	134.74	165.05	127.47
Arithmetic mean (127.5 = random)	48.3786	48.6232	48.7726	48.5305
Monte Carlo π estimation	4.000000000 (error 27.32%)	4.000000000 (error 27.32%)	4.000000000 (error 27.32%)	4.000000000 (error 27.32%)
Entropy (bits/byte)	0.957074	0.955771	0.773429	0.997318
Compression Rate	13625152	13625152	62808768	13625152

Table 5.2 Diehard test results of the true random numbers generated from four different sources

Statistical Test	P-value			
	Accelerometer	Temperature sensor	DCO-VLO	ADC
Birthday spacings	1.000000	1.000000	1.000000	1.000000
Overlapping Permutations	1.000000	1.000000	1.000000	1.000000
Ranks of 31×31 and 32×32 Matrices	1.000000	1.000000	1.000000	1.000000
Ranks of 6×8 Matrices	1.000000	1.000000	1.000000	1.000000
Monkey tests on 20-bit words	1.000000	1.000000	1.000000	1.000000
Monkey Test OPSO	1.000000	1.000000	1.000000	1.000000
Monkey Test QSO	1.000000	1.000000	1.000000	1.000000
Monkey Test DNA	1.000000	1.000000	1.000000	1.000000
Count the 1's in a stream of bytes	1.000000	1.000000	1.000000	1.000000
Count the 1's in specific bytes	1.000000	1.000000	1.000000	1.000000
Parking lot test	1.000000	1.000000	1.000000	1.000000
Minimum distance test	1.000000	1.000000	1.000000	1.000000
Random spheres test	1.000000	1.000000	1.000000	1.000000

When the results given in Table 5.2 above are analysed, it is seen that the numbers produced from 4 different hardware sources fail all tests. If the p-value is in the range 0.0001-0.009 in the Diehard tests, the test is considered successful (Kim, Lee, Tanaka & Park, 2012). Otherwise the test result is unsuccessful.

5.1.2 Test Results for Random Number Generation by Using Temperature Sensor

When we have generated 1000000 numbers with temperature sensor and tested them using statistical tests in NIST test environment, we obtained results as given in Table 5.3. The generated random numbers have failed in all statistical tests.

Table 5.3 NIST test results of the true random numbers generated using temperature sensor

Statistical Test	TRNG using Temperature Sensor		
	P-value	Proportion	Status
Frequency	0.00000	0.00000	Failed
Block Frequency	0.00000	0.00000	Failed
Cumulative Sums(Average)	0.00000	0.00000	Failed
Runs	0.00000	0.00000	Failed
Longest Run	0.00000	0.00000	Failed
Rank	0.00000	0.00000	Failed
FFT	0.00000	0.00000	Failed
Nonperiodic Templates(Average)	0.00000	0.00170	Failed
Overloading Templates	0.00000	0.00000	Failed
Universal	0.00000	0.00000	Failed
Approximate Entropy	0.00000	0.00000	Failed
Random Excursions(Average)	0.00000	0.00000	Failed
Random Excursions Variant(Average)	0.00000	0.00000	Failed
Serial(Average)	0.00000	0.00000	Failed
Linear Complexity	0.000089	1.00000	Failed

5.1.3 Test Results for Random Number Generation by Using Analog to Digital Converter

1000000 true random numbers were generated by sampling via analog to digital converter.

Table 5.4 True random number generator which generates numbers by sampling voltage on analog to digital converter

Statistical Test	P-value	Proportion	Status
Frequency	0.00000	0.00000	Failed
Block Frequency	0.00000	0.00000	Failed
Cumulative Sums(Average)	0.00000	0.00000	Failed
Runs	0.00000	0.00000	Failed
Longest Run	0.00000	0.28486	Failed
Rank	0.00000	0.00000	Failed
FFT	0.00000	0.00000	Failed
Nonperiodic Templates(Average)	0.00000	0.00170	Failed
Overloading Templates	0.00000	0.00000	Failed
Universal	0.00000	0.00000	Failed
Approximate Entropy	0.00000	0.00000	Failed
Random Excursions(Average)	0.00000	0.00000	Failed
Random Excursions Variant(Average)	0.00000	0.00000	Failed
Serial(Average)	0.00000	0.00000	Failed
Linear Complexity	0.213309	0.87500	Passed

The NIST test results of these numbers can be seen in Table 5.4. As we can see in the table, true random number generator has failed all tests except Linear Complexity test. According to these results, it can be said that this generator is not statistically random and secure.

5.1.4 Test Results for Random Number Generation by Using Difference Between Clock Pulses

1000000 numbers are produced by using WISP's two clocks working at different frequency. We can see the NIST test results of this number series in Table 5.5. According to the test results, all tests except Linear Complexity test failed. According to these results, we can say that the generated numbers are not random and statistically secure.

Table 5.5 NIST test results for true random number generators by using difference between DCO & VLO clock pulses

Statistical Test	TRNG using DCO & VLO frequency difference		
	P-value	Proportion	Status
Frequency	0.00000	0.00000	Failed
Block Frequency	0.00000	0.00000	Failed
Cumulative Sums(Average)	0.00000	0.00000	Failed
Runs	0.00000	0.00000	Failed
Longest Run	0.00000	0.00000	Failed
Rank	0.00000	0.00000	Failed
FFT	0.00000	0.00000	Failed
Nonperiodic Templates(Average)	0.00000	0.00170	Failed
Overloading Templates	0.00000	0.00000	Failed
Universal	0.00000	0.00000	Failed
Approximate Entropy	0.00000	0.00000	Failed
Random Excursions(Average)	0.00000	0.00000	Failed
Random Excursions Variant(Average)	0.00000	0.00000	Failed
Serial(Average)	0.00000	0.00000	Failed
Linear Complexity	0.20443	1.00000	Passed

5.1.5 Test Results for Random Number Generation by Using Accelerometer

X, y and z coordinate values are produced using the WISP's accelerometer sensor. From these values, 1000000 random numbers were generated using first method given in Figure 4.5. These numbers have been tested with the NIST test. The test results are shown in Table 5.6. According to these results, only the Linear complexity test was successful. This result means that the randomness of this number series is low.

Table 5.6 NIST test results for true random number generator using accelerometer (1st method)

Statistical Test	P-value	Proportion	Status
Frequency	0.00000	0.00000	Failed
Block Frequency	0.00000	0.00000	Failed
Cumulative Sums(Average)	0.00000	0.00000	Failed
Runs	0.00000	0.00000	Failed
Longest Run	0.00000	0.00000	Failed
Rank	0.00000	0.00000	Failed
FFT	0.00000	0.00000	Failed
Nonperiodic Templates(Average)	0.00000	0.00170	Failed
Overloading Templates	0.00000	0.00000	Failed
Universal	0.00000	0.00000	Failed
Approximate Entropy	0.00000	0.00000	Failed
Random Excursions(Average)	0.00000	0.00000	Failed
Random Excursions Variant(Average)	0.00000	0.00000	Failed
Serial(Average)	0.00000	0.00000	Failed
Linear Complexity	0.21331	1.00000	Passed

After the results in Table 5.6, the random number generation algorithm was changed. Using the algorithm in the flow chart given in Figure 4.6, 1000000 numbers have been generated. Although improvements were observed in NIST test results, it failed all tests except Linear Complexity, again. Results can be seen in Table 5.7 for second method.

Table 5.7 NIST test results for true random number generator using accelerometer (2nd method)

Statistical Test	P-value	Proportion	Status
Frequency	0.00000	0.03125	Failed
Block Frequency	0.00000	0.00000	Failed
Cumulative Sums(Average)	0.00000	0.00000	Failed
Runs	0.00000	0.00000	Failed
Longest Run	0.00000	0.00000	Failed
Rank	0.00000	0.00000	Failed
FFT	0.00000	0.28125	Failed
Nonperiodic Templates(Average)	0.00000	0.07161	Failed
Overloading Templates	0.00000	0.00000	Failed
Universal	0.00000	0.00000	Failed
Approximate Entropy	0.00000	0.00000	Failed
Random Excursions(Average)	0.00000	0.00000	Failed
Random Excursions Variant(Average)	0.00000	0.88889	Failed
Serial(Average)	0.00000	0.00000	Failed
Linear Complexity	0.53415	1.00000	Passed

5.1.6 Test Results for Merged Linear Feedback Shift Register

We generated both 1000000 16-bit and 32-bit random numbers using C rand() function. These numbers were tested with NIST statistical randomness test suite. The test results can be seen in Table 5.8. We can see that all tests failed.

Table 5.8 NIST test result for 16-bit and 32-bit random numbers generated using C rand() function

P-Value		Proportion		Statistical Test	Status	
16-bit	32-bit	16-bit	32-bit		16-bit	32-bit
0.00000	0.00000	0.0000	0.0000	Frequency	Failed	Failed
0.00000	0.00000	0.0000	0.0000	Block Frequency	Failed	Failed
0.00000	0.00000	0.0000	0.0000	Cumulative Sums	Failed	Failed
0.00000	0.00000	0.0000	0.0000	Cumulative Sums	Failed	Failed
0.00000	0.00000	0.0000	0.0000	Runs	Failed	Failed
0.00000	0.00000	0.0000	0.0000	Longest Run	Failed	Failed
0.00000	0.00000	0.0000	0.0000	Rank	Failed	Failed
0.00000	0.00000	0.0000	0.0000	FFT	Failed	Failed
0.000003	0.00000	0.41005	0.4176	Non Overlapping Template (Average)	Failed	Failed
0.00000	0.00000	0.0000	0.0000	Overlapping Template	Failed	Failed
0.00000	0.00000	0.0000	0.0000	Universal	Failed	Failed
0.00000	0.00000	0.0000	0.0000	Approximate Entropy	Failed	Failed
----	----	----	----	Random Excursions (Average)	Failed	Failed
----	----	----	----	Random Excursions Variant (Average)	Failed	Failed
0.00000	0.00000	0.0000	0.0000	Serial	Failed	Failed
0.03517	0.46860	0.9375	1.0000	Serial	Passed	Passed
0.21331	0.94960	1.0000	1.0000	Linear Complexity	Passed	Passed

One million 32-bit random numbers produced by MLFSR were tested using NIST statistical randomness test suite. The test results can be seen in Table 5.9. Random numbers generated by MLFSR were successful in all tests.

Table 5.9 NIST test result for 32-bit random numbers generated by merged linear feedback shift register

P-value	Proportion	Statistical Test	Status
0.53415	1.0000	Frequency	Passed
0.35049	1.0000	Block Frequency	Passed
0.40709	1.0000	Cumulative Sums	Passed
0.73992	1.0000	Cumulative Sums	Passed
0.80434	1.0000	Runs	Passed
0.86234	1.0000	Longest Run	Passed
0.35049	1.0000	Rank	Passed
0.06688	1.0000	FFT	Passed
0.41736	0.9903	Non Overlapping Template (Average)	Passed
0.67178	1.0000	Overlapping Template	Passed
0.80434	1.0000	Universal	Passed
0.17828	1.0000	Approximate Entropy	Passed
0.09803	0.9853	Random Excursions (Average)	Passed
0.07889	0.9902	Random Excursions Variant (Average)	Passed
0.60246	0.9688	Serial	Passed
0.46860	0.9688	Serial	Passed
0.40709	1.0000	Linear Complexity	Passed

We can see the p-value and proportion values of the statistical tests in NIST test suite in Table 5.8 and Table 5.9. If p-value is greater than 0.01 (Marton & Suciu, 2015) and the proportion value is greater than 0.95 for each test, the test result is successful (Bassham et al., 2010; Özcanhan, Dalkılıç & Gürle, 2012).

The developed MLFSR was tested on the WISP passive RFID tag. Number production times were measured in the tests. For this process 1000 numbers were generated in each round. The time elapsed for this operation was measured. First, the time for the number generation using C rand() function was measured. After that, 1000 numbers were generated for 3 times. The obtained generation time values can be seen in Table 5.10. The C rand() function and MLFSR generated random numbers in approximately 88 ms. Only MLFSR produced numbers in approximately 57 ms. So we can say that MLFSR is suitable for the generation of random numbers under EPC Gen2 Class 1 standards (Feldhofer & Wolkerstorfer, 2009).

Table 5.10 Merged linear feedback shift register working times on WISP

	Number Generation Time (ms)		
	1	2	3
MLFSR	57.994	57.644	57.645
MLFSR + rand() function	88.395	88.403	88.449

5.1.7 Test Results for XorshiftR+

The RNGs are tested using the Big Crush battery of tests, which is the most comprehensive test battery in the TestU01 suite. It contains 106 different tests and makes a total of 160 runs (some tests are repeated with different parameters) (L'Ecuyer & Simard, 2007; L'Ecuyer & Simard, 2013).

L'ecuyer and Simard defined 3 failures out of 15 tests of Small Crush battery and 7 failures out of 38 tests of Rabbit battery as “clearly unacceptable”. Which make 80% and 81.6% respectively. So, in order to be called as successful, any RNG must obtain a clearly higher score than these, preferably 100% depending on the scenario. Nevertheless, the rate of passed tests is not solely enough to decide, but the result values of each failed test should also be taken into account. TestU01 checks p-values to determine if RNGs pass certain tests or not, for all tests, with a pass interval of 0.001 to 0.999. It also introduces two epsilon values, namely eps and eps1, which represent very small numbers (latter being much smaller) and are practically equal to 0 (L'Ecuyer & Simard, 2007; L'Ecuyer & Simard, 2013). Hence, if an RNG fails at least one test

with a value of ϵ (or ϵ_1) then that RNG will most likely to fail this test in each run and possibly for all seeds, which means a systematic fail. But if the p-value is very close to the limits, (while the term “very close” is essentially vague), for instance $0.999 < p < 0.9999$, then the result is not clear. Even though, L’ecuyer and Simard propose repeating the suspicious tests is the best practice to get rid of the suspicious results. Our experiments show that in the case when a p-value falls in the range given above, the test can be accepted as a pass without repeating it, especially if it is the only fail or there are very few, since a repeat with a different seed will most likely to result a pass. However, if the p-value is not so close to the limits, though not ϵ , then repeating the tests might be necessary. Note that, the same seed most likely will cause exactly the same result in further repeats. If repeating the tests is not feasible, these tests should be taken as fails. Remember, the number of total test runs are more than the number of different tests, since some of the tests are repeated with various parameters in all batteries (L’Ecuyer & Simard, 2013).

Other than the randomness levels, since we also have lightness concerns, run time, seed and output sizes of each algorithm are taken into account to determine their memory usage. Memory usage is not in the center of the focus since no tests are seen as required besides the register size calculations, which can be done using function definitions.

For evaluating the time performance, two measures are considered; total time lapsed during the tests and time period required to generate 100 million numbers (not bits). Total duration of the tests is a good indicator of speed of the generators because it represents a comprehensive usage scenario. However, it is not always very consistent as number of tests applied may differ for some RNGs depending on their algorithmic structure. Thus, we also recorded the time required to create 100 million numbers to be sure about their pure speed. This is done using the native timer function of the TestU01 suite, with the default method GetU01. In order to improve the solidity of our tests and recommendations, we have run most of the algorithms, namely xorshift based ones and LFSR113, also on a WISP device ten thousand times sequentially and recorded the total creation time likewise. This additional step made us sure, our claims, verified on a PC are also valid for a real lightweight IoT device and vice versa.

When an RNG is forced to use different seed numbers without completing its regular run for its full period, i.e. seed is changed after each random number output, then the RNG will highly probable lose its ability to produce randomness (except for some, i.e. LFSR derivative scramblers). In this case, degree of the randomness of the output will depend on the randomness of the seed string, because every seed input will cause the RNG to produce the same set of random numbers in its each run. This prevented us from using hardware generated so-called true random numbers as a seed chain. Instead, we used only a single seed (duplicated when RNG required multiple seeds) for each subject generator. A seed of an RNG is only changed to renew a test if there are suspicious results.

5.1.7.1 Empirical Results

In order to make the final evaluation, all subject generators are tested with Big Crush battery of the TestU01 suite, which is the biggest and toughest battery in the suite, while other batteries (like Small Crush and NIST) are used to make preliminary tests.

Table 5.11 Comparative data of empirical test results from Big Crush for all subject RNGs

Empirical Results	Big Crush score (# of runs)	Big Crush score (# of tests)	Run time of Big Crush	Time for 1 number on PC	Time for 1 number on WISP
xorshift+	159/160 99.3%	105/106 99.1%	7h:35:03	0.02 μ s	0.5515 ms
xorshift*	156/160 97.5%	102/106 96.2%	7h:28:10	0.0202 μ s	0.2703 ms
Reduced xorshift+var1	155/160 96.9%	103/106 97.2%	7h:27:02	0.0191 μ s	0.1955 ms
Reduced xorshift+var2	160/160 100%	106/106 100%	7h:32:40	0.0192 μ s	0.1954 ms
Reduced xorshift+var3 [xorshiftR+]	160/160 100%	106/106 100%	7h:30:07	0.0191 μ s	0.1953 ms
Simple LCG	91/153 59.5%	57/106 53.8%	7h:25:45	0.0177 μ s	N/A
LFSR113	154/160 96.3%	100/106 94.3%	6h:47:57	0.0175 μ s	0.1641 ms
C rand()	0/160 0%	0/106 0%	N/A	N/A	N/A

As a recall, Big Crush contains 106 tests and makes a total of 160 runs, since some of the tests are repeated with different parameters. Furthermore, in order the find out the average time required for generation of a single random number for each algorithm;

100 million numbers on PC and 10,000 numbers on WISP are generated. Hence, the average time required for 1 number is calculated and shown in the Table 5.11.

5.1.7.2 Testbeds

Our PC testbed was a generic server computer with Intel Xeon E5540 processor @2.53 GHz running on Windows utilizing 4 GB of RAM. Nevertheless, we observed that the software of TestU01 only occupies a single core during the whole test runs. This leads a more linear base time comparison and prevented us to use our bed at full rate, but this might also be positive since a lightweight system, highly probable, will only have single core dedicated to random number generation. Our IoT testbed is a standalone WISP5 device which includes a built-in MSP 430 processor. Results of the tests are as follows;

5.1.7.3 Xorshift+

The original xorshift+, not so surprisingly, demonstrated excellent randomness according to Big Crush tests as Vigna claimed (Vigna, 2017). It passed 105 of the 106 different tests and 159 of the 160 test runs in our initial attempt.

Here, the p-value for this test is very close to the limit and as explained in Section 4, it can be marked as a false positive and be ignored. In fact, repeated attempts show that this fail was not systematic, but a coincidence. So, the ultimate scores might also be taken as 100% for both measures. We still kept this fail in our records for the sake of temporal consistency. It was however, somewhat slow in terms of speed on the PC and the slowest on WISP, when compared to other RNGs, as can be seen in the Table 5.11, which makes our efforts more significant.

*5.1.7.4 Xorshift**

The original xorshift* obtained, not surprisingly, slightly lower randomness scores on Big Crush suite, though not bad. It failed four tests systematically. On PC, its speed was unsatisfactory, actually slightly the worst among all. However, on the WISP device it was (only) better than xorshift+. Please see the Table 5.11. Hence, it cannot

be recommended as a lightweight IoT solution, but might be used on PC platforms depending on the scenario.

5.1.7.5 *Reduced Xorshift+ var1* ($s[1] = x$, return $x + y$)

Our first variation obtained test scores very close to the original xorshift*, they also share 81. LinearComp test as a fail. On the other hand, our variation was super-fast. On PC, it was the fastest (along with the variation 3) among other xorshift based subjects. It failed five tests systematically.

From the p-values of the failed tests, we can say that these are clear fails, because they are far out of the limits. However, considering its speed, this might still be a useful generator since its score is much higher than the (bad) examples given in Section 4 (which were 80% and 81.6%).

5.1.7.6 *Reduced Xorshift+ var2* ($s[1] = x + y$, return $x + y$)

Our second variation got excellent scores (which is 100%) from Big Crush suite by passing all the tests in all runs with no exception. Even though it wasn't the fastest, it was still clearly faster than the original xorshift+ and the xorshift*. This agility difference is much more obvious on WISP, rather than the PC. On PC, all our variations showed very similar results.

5.1.7.7 *Reduced Xorshift+ var3* ($s[1] = x + y$, return x)

This third variation (which we have named as xorshiftR+) demonstrated excellent scores (again 100%) by passing all the tests in all runs without any exception, too. In fact, variation 3 was slightly faster than the variation 2 both on PC and on WISP. It was also slightly faster than the variation 1 on WISP, while having the same speed on PC. Hence, it is the fastest of its kind. Considering its perfect and consistent randomness scores and very good time performance, it can safely and properly replace the original xorshift+. Because, there is no statistics, in which the original xorshift+ is better than our variation 3 as can be seen in the Table 5.11.

A bitmap graph using TestU01's PlotUnif function is seen in Figure 5.1. It shows how the bits are randomly distributed under function's drawing rules.

Even though the TestU01 suite is already superior to the well-known NIST suite, because of its widespread use and fame, we have tested xorshiftR+ using the NIST suite, too. The result was a complete pass for all instances of all tests in the suite shown in Table 5.12, as expected. The test was made with 1,000,000 generated 64-bit numbers.

Table 5.12 NIST STS results for reduced xorshift+ var3

Statistical Test	P-value	Proportion	Status
Frequency	0.195163	1.000000	Passed
Block Frequency	0.602458	1.000000	Passed
Cumulative Sums (Average)	0.519866	1.000000	Passed
Runs	0.804337	0.984375	Passed
Longest Run	0.964295	0.984375	Passed
Rank	0.122325	0.984375	Passed
FFT	0.637119	1.000000	Passed
Non Overlapping Template (Average)	0.470341	0.989970	Passed
Overlapping Template	0.060239	0.984375	Passed
Universal	0.253551	0.984375	Passed
Approximate Entropy	0.437274	0.984375	Passed
Random Excursions(Average)	0.421135	0.979730	Passed
Random Excursions Variant(Average)	0.339985	0.990991	Passed
Serial(Average)	0.545201	0.976562	Passed
Linear Complexity	0.437274	0.984375	Passed

To assure the algorithm's compliance with the RFID tags' security standards, three conditions mentioned in the EPC™ Gen-2 Class 1 document were tested, too. The first implies that the probability of a single 16-bit random number should be $0.8/2^{16} < P(\text{RN16}) < 1.25/2^{16}$ for 2^{30} numbers. xorshiftR+ satisfies this condition, moreover, even for 2^{26} numbers, the result was $0.852/2^{16} < P(\text{RN16}) < 1.18/2^{16}$. The second mentions that the probability of simultaneously identical sequences for 10000 tags should be $< 0.1\%$. With xorshiftR+, since there are two 64-bit seeds, this probability is calculated as $(1/2^{64}) \times (1/2^{64}) = 1/2^{128}$ then the result is, $[10000 \times (1/2^{128})] * 100 = 2.94\% \times 10^{-34} < 0.1\%$. Lastly, the third rule can be explained as an RN16 generated from a tag's RNG has to have a predictable probability less than 0.025%. This is proven via the ENT suite, another old yet popular test suite. There, we have also used the original xorshift+ for the sake of a consistent comparison; and xorshiftR+ demonstrated a very high performance that is practically equivalent to the original

xorshift+. Detailed information regarding these conditions, as well as the test results can also be found on the link given above.

Table 5.13 ENT test result for original xorshift+

	Entropy	Compression Rate	X ² Statistic	Arithmetic Mean	Monte Carlo π	Serial Correlation Coefficient
1st run seeds: 0x17AF340B84F0F0F09 0x20F470F076F6F082	7.999915	%0	236.87 %78.61	127.5494	3.143247143 (0.05% error)	0.000472
2nd run seeds: 0x17AF343B84F0F6F09 0x20F470F076F6F082	7.999914	%0	238.79 %75.93	127.5182	3.144255144 (0.08% error)	-0.000554
3rd run seeds: 0x17AF343B84F0F6F09 0x23F473F276F6F782	7.999901	%0	274.00 %19.75	127.5069	3.139911140 (0.05% error)	-0.000211

In table 5.13, ENT test results for original xorshift+ was given. Moreover, we gave the ENT test results for xorshiftR+ for the same seeds in Table 5.14. Overall, for xorshiftR+, ENT, NIST and EPC Gen-2 Class 1 examinations were successful, too, in addition to TestU01.

Table 5.14 ENT test result for xorshiftR+

	Entropy	Compression Rate	X ² Statistic	Arithmetic Mean	Monte Carlo π	Serial Correlation Coefficient
1st run seeds: 0x17AF340B84F0F0F09 0x20F470F076F6F082	7.999913	%0	241.44 %71.97	127.5187	3.142359142 (0.02% error)	0.000125
2nd run seeds: 0x17AF343B84F0F6F09 0x20F470F076F6F082	7.999916	%0	232.26 %84.35	127.4130	3.148203148 (0.21% error)	-0.000097
3rd run seeds: 0x17AF343B84F0F6F09 0x23F473F276F6F782	7.999911	%0	246.28 %64.10	127.5543	3.138699139 (0.09% error)	0.000229

5.1.7.8 Simple Linear Congruential Generator

The simple LCG, implemented with parameters (2147483647, 16807, 0, 12345); was very weak in terms of randomness, since it can only pass a little more than the half of the tests and test runs, which is far below the unacceptable examples given in Section 4 and the lowest among our subjects. Figure 5.1, generated by scatter of TestU01, reveals a repeating pattern and thus clearly shows that the generated bits are not random. Total number of test runs was lower because some tests were not repeated

as a result of the RNG's poor performance as described in Section 4. Names and parameters of the failed tests were not presented because of their huge count. All fails were clear considering their p-values. Note that, the run time for the suite given in the Table (which is very short) might be misleading since the number of test runs were fewer than the others (because of its poor randomness performance). Anyway, number creation times show that this RNG is very fast in the scale (not tested on WISP). Although the applied LCG is rapid, it should not be used in any application that requires high randomness and is not recommended for any scenario.

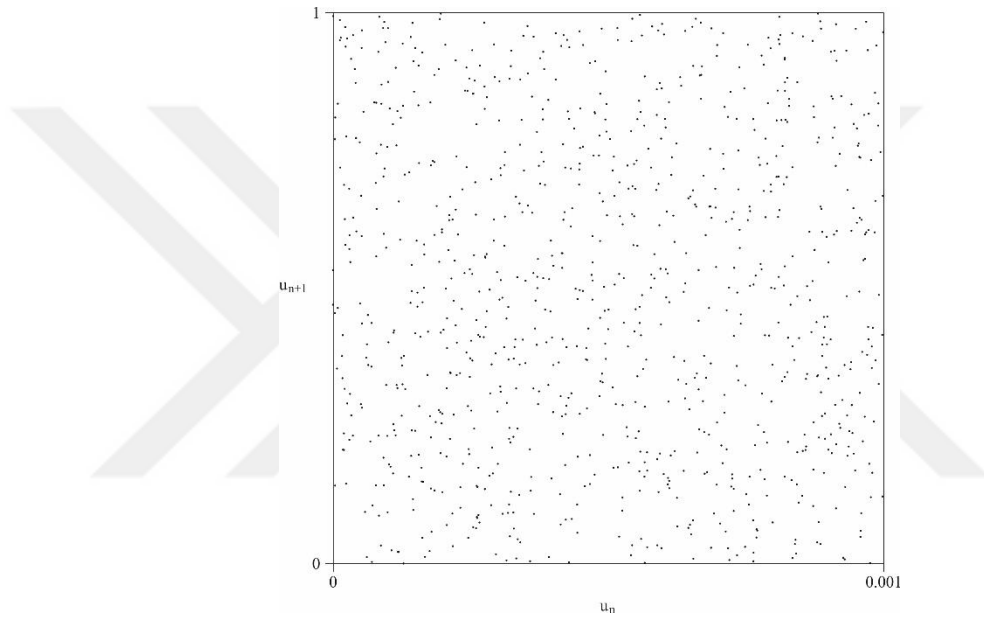


Figure 5.1 Bitmap of xorshiftR+; 1000 points plotted using scatter in TestU01 (u_n and u_{n+1} stand for sequential random numbers. See definitions in (L'Ecuyer & Simard, 2013))

5.1.7.9 Linear Feedback Shift Register: LFSR113

LFSR113, with parameters (12345, 12345, 12345, 12345); was the fastest RNG on the PC among our candidates with a decent difference when compared to xorshift based subjects including our variations. It was approximately 8% faster than all our variations on PC. Nevertheless, it is remarkable that it demonstrated even better performance on the WISP. Conversely, LFSR has indecisive test scores, which can be found acceptable depending on the scenario, but surely not better than xorshift family. The generator failed six tests systematically. This RNG is recommendable on PC and

in a case where the randomness requirements are loose but time limitation is a more dominant concern.

5.1.7.10 Xoroshiro+

After we have concluded our tests and most of the study as well; a brand new xorshift based generator, namely xoroshiro128+, was announced by Vigna. It is claimed that, this is the fastest of all. Hence, we could not ignore and made a comparative analysis, though we could not run a new TestU01.

xoroshiro128+ contains the following operations: 105 left shifts, 37 right shifts, 1 addition (64-bit number), 2 ORs (64-bit numbers), and 2 XORs (64-bit numbers). A total of 462 bitwise operations. While our xorshiftR+ only contains, 23 left shifts, 17 right shifts, 2 XORs (64-bit numbers), and 1 addition (64-bit number); which makes 232 bitwise operations. Since types of the operations required are the same for both, and the number of operations for xorshiftR+ are almost half of the ones for xoroshiro+; xorshiftR+ is clearly lighter and expected to run faster. Thus, xorshiftR+ is apparently superior to xoroshiro+, too.

5.1.7.11 Further Notes

As a further information, test results of the given xorshift based generators represent their 32-bit output versions, instead of the contemporarily used 64-bit (or more) versions, where the integers and the outputs are 64-bit numbers. Because, on the lightweight family of devices, 32-bit is a common upper limit. Presumably, 64-bit versions get the same or better results from the tests. In contrast, 16-bit integer sized versions of the generators (including the original xorshift+) fail almost all tests even in the Small Crush battery. This is because, many tests of the TestU01 assume an output of at least 30 bits and fails the subject generator otherwise (L'Ecuyer & Simard, 2013). Anyway, test results of these 16-bit versions are still very weak (fail most of the tests in the suite) and are not statistically valuable. This also explains why standard rand() function of C language (initiated by the srand(time(NULL)) code) fails all the tests in the Big Crush suite (obtaining a score of 0% in both measures). Because of their very poor performance, rand() and simple LCG are not tested on WISP.

Please note, the runtime or number generation time difference between the RNGs may look small or even negligible, but surely not. When slower machines are used as

hosts or extensive repeats will be made, these differences shall become more obvious. All three of our reduced variations were faster than the xorshift+ itself. But, our variation 1 couldn't pass all the tests and so is discarded during the final evaluation. Even though variation 2 also passed all the tests, yet variation 3 was the fastest among all our variations and consequently selected as the xorshiftR+. It can be used wherever xorshift+ can be used and it can safely replace the xorshift+ in further studies and applications.

Additionally, xorshift* can be considered as deprecated after xorshift+ (and xorshiftR+) is introduced, this was also mentioned in (Vigna, 2017). The simple LCG was expectedly very weak (in terms of randomness) and should not be used in any scenario without further modifications. LFSR was very fast however, weaker than all the xorshift family members given here. Lastly, bitmap graphs are not drawn for other subject generators, since it wasn't possible for human eye to detect a pattern from them as well. So, only the best and the worst generators' graphs are shown. Figure 5.1, generated by scatter of TestU01, shows bitmap of xorshiftR+.

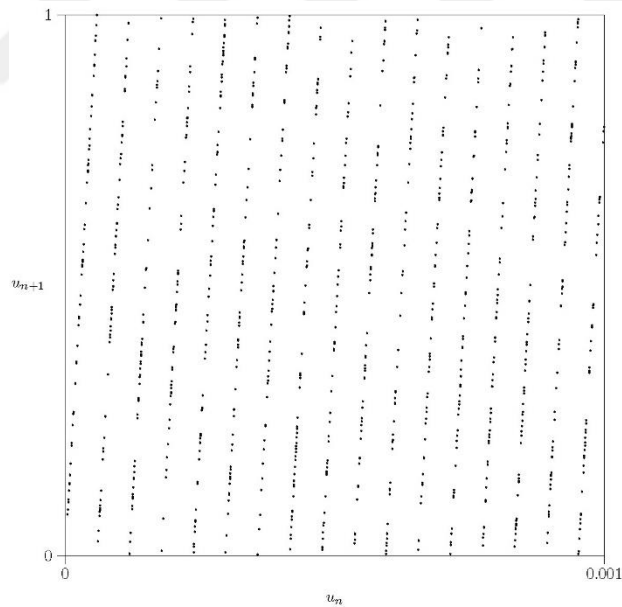


Figure 5.2 Bitmap of simple LCG; 1000 points plotted using scatter in TestU01 (u_n and u_{n+1} stand for sequential random numbers. See definitions in (L'Ecuyer & Simard, 2013)

5.1.8 Test Results for XorshiftL+

Proposed HRNG was tested with NIST test suite. Statistical randomness test results are given in Table 5.15. It is also compared with well-known algorithms listed in Table

5.16. All of the algorithms ran on WISP and PC. XorshiftL+ was also checked for EPCGen2 standards. Three conditions of EPCGen2 standard were examined and xorshiftL+ satisfies all these three conditions.

The new HRNG was tested using the well-known test suite, NIST, which has a battery of statistical tests, such as Rank, Runs, Serial, Frequency, FFT. It makes 188 runs for 15 different tests by running certain tests with different parameters (Bassham et al., 2010). 16 million bits are generated by the HRNG and tested with NIST STS. NIST test results show that the new HRNG is successful according to test results, and also show that HRNG produces statistically random number sequences. NIST test suite results are given in Table 5.15.

Table 5.15 NIST STS results for xorshiftL+

Statistical Test	P-Value	Proportion	Status
Cumulative Sums	0.066882	1.000000	Passed
Runs	0.148094	0.937500	Passed
Longest Run	0.213309	1.000000	Passed
Rank	0.671779	1.000000	Passed
FFT	0.739918	1.000000	Passed
Non Overlapping Template (Average)	0.427527	0.987542	Passed
Overlapping Template	0.468595	1.000000	Passed
Universal	0.407091	1.000000	Passed
Approximate Entropy	0.407091	0.906250	Passed
Random Excursions (Average)	0.178357	0.781250	Passed
Random Excursions Variant (Average)	0.254208	0.781250	Passed
Serial (Average)	0.264708	0.984375	Passed
Linear Complexity	0.100508	1.000000	Passed

Table 5.16 shows that xorshiftL+ is faster than rival algorithms. In the time evaluation of the xorshiftL+ algorithm, the PRNG time value only is calculated and TRNG is excluded because it works once at the beginning of the algorithm to produce a seed, and the time required is negligible. XorshiftL+ generates random numbers about 16% faster than the closest rival. It also outperforms the rival algorithms in terms of number of operators used and variable lengths. Compared with the xorshift128+ algorithm, it is seen that the operations in the xorshift128+ algorithm are performed on 64-bit length numbers. It is also observed that there are 1 mathematical addition operation, 23 logical left shifts, 23 logical right shifts and 4 XOR operations in this algorithm. In contrast, all operations in the xorshiftR+ algorithm are 64 bits length variables. xorshiftR+ has 1 mathematical addition, 23 logical left shift, 17 logical right

shift and 2 XOR operations. In our new proposed xorshiftL+ algorithm, there are 1 mathematical addition, 3 logical left shifts, 7 logical right shifts and 4 XOR operations. This new algorithm has been developed to ensure the faster generation of numbers by applying less processing with 32 bit variables.

Table 5.16 Time comparison with the known pseudorandom number generator algorithms

PRNG algorithm	Elapsed time on WISP (ms)	Elapsed time on PC (ms)
Uprng (1000 numbers generated)	35695	0.437100
AKARI 1(1000 numbers generated)	39873	0.227833
AKARI 2(1000 number generated)	25022	0.171398
xorshiftL+(10000 number generated)	1934	0.023285
xorshiftR+(10000 numbers generated)	2309	0.027410
xorshift128+(10000 numbers generated)	3183	0.033485
xorshift1024+(10000 numbers generated)	3307	0.043265
xorshift64*(10000 numbers generated)	3868	0.040750
xorshift1024*(10000 numbers generated)	4001	0.043925
well1024(10000 numbers generated)	5242	0.041230
LFSR113(10000 numbers generated)	6489	0.058850
LFSR258(10000 numbers generated)	10545	0.134425

We tested the HRNG for three conditions, as pointed out in the EPC™ Gen-2 Class 1 document (EPCGlobal GS1, 2015), to ensure it meets the RFID tags' security standards. Each 16-bit random number generated in 2^{30} numbers should have the probability of $0.8/2^{16} < P(RN16) < 1.25/2^{16}$. The HRNG should satisfy this condition.

Table 5.17 ENT test results for the proposed hybrid random number generator

Test	(1 st run seeds)		(2 nd run seeds)		(3 rd run seed)	
	x (int32_t)	s (int32_t)	x (int32_t)	s (int32_t)	x (int32_t)	s (int32_t)
	1,976,529,755	-1,089,211,351	-2,112,299,155	438,506,025	154,5271,002	269,685,289
Entropy	1.000000 bits/bit		1.000000 bits/bit		1.000000 bits/bit	
Compression Rate	%0		%0		%0	
X ² Statistic	0.23 %62.84		1.56 %21.18		0.08 %78.22	
Arithmetic Mean	0.5001		0.4998		0.5000	
Monte Carlo π	3.158355158 (0.53% error)		3.160407160 (0.60% error)		3.157515158 (0.51% error)	
Serial Correlation Coefficient	-0.005478		-0.005967		-0.005634	

Our proposed xorshiftL+ satisfies this condition with the probability $0.917/2^{16} < P(\text{RN16}) < 1.048/2^{16}$ for 2^{30} numbers. Secondly, simultaneously identical sequences' probability for 10000 tags should be less than 0.1%. XorshiftL+ has two inputs, and these two seeds are 32-bit integers, so calculating the probability as $(1/2^{32}) \times (1/2^{32}) = 1/2^{64}$, the result is $[10000 \times (1/2^{64})] * 100 = 5.42\% \times 10^{-18} < 0.1\%$, this condition is also satisfied. The final condition is that an RN16 drawn from a tag's RNG shall not be predictable with a probability greater than 0.025%. This was proved using an ENT test suite detailed and defined on www.fourmilab.ch/random (Walker, 2008). The detailed results and test parameters can be seen in Table 5.17.

5.1.9 Test Results for Random Number Generation Using Genetic Algorithm

By using first method shown in Figure 4.10 random numbers were generated and wrote in a txt file as binary string. Then, we tested this content with NIST test suite. You can see test results in Table 5.18. Generated numbers failed in all tests except Linear Complexity.

Table 5.18 NIST test results for the numbers generated by GA (1st Method)

P-Value	Proportion	Statistical Test	Status
0.000000	0.000000	Frequency	Failed
0.000000	1.000000	Block Frequency	Failed
0.000000	0.000000	Cumulative Sums	Failed
0.000000	0.000000	Cumulative Sums	Failed
0.000000	0.000000	Runs	Failed
0.000000	0.000000	Longest Run	Failed
0.000000	0.000000	Rank	Failed
0.000000	0.000000	FFT	Failed
0.118712	0.568202	Non Overlapping Template (Average)	Failed
0.000000	0.000000	Overlapping Template	Failed
0.000000	0.218800	Universal	Failed
0.000000	0.000000	Approximate Entropy	Failed
----	0.875000	Random Excursions (Average)	Failed
----	0.527778	Random Excursions Variant (Average)	Failed
0.000000	0.000000	Serial	Failed
0.000000	0.281200	Serial	Failed
0.299251	1.000000	Linear Complexity	Passed

Test result for 16,000,000 bits (1,000,000 16-bit number) generated from the PRNG generated by GA (2nd method) can be seen in Table 5.19. PRNG which successfully passes the NIST STS is produced with GA. It shows that a statistically good PRNG can be generated with GA using entropy calculation as fitness function.

Table 5.19 NIST test results for the output of the PRNG generated by GA (2nd Method)

Statistical Test	P-Value	Proportion	Status
Frequency	0.12233	0.90625	Passed
Block Frequency	0.53415	0.96875	Passed
Cumulative Sums (Average)	0.13939	0.96875	Passed
Runs	0.40709	1.00000	Passed
Longest Run	0.73992	0.96875	Passed
Rank	0.91141	1.00000	Passed
FFT	0.29925	1.00000	Passed
Non Overlapping Template (Average)	0.42055	0.98944	Passed
Overlapping Template	0.46860	1.00000	Passed
Universal	0.17828	0.93750	Passed
Approximate Entropy	0.67178	0.96875	Passed
Random Excursions (Average)	0.19137	0.97321	Passed
Random Excursions Variant (Average)	0.19576	0.97222	Passed
Serial (Average)	0.70340	0.98438	Passed
Linear Complexity	0.46860	0.93750	Passed

5.2 Results and Analysis for the Grouping Proof Authentication Protocol Using XorshiftL+

GPAPXL+ was evaluated against some well-known RFID attacks to discover whether or not it was strong against them. In addition, the protocol was simulated with a special programming language and evaluated using the Scyther protocol security analysis tool developed by Cremers (2018).

This section contains an analysis of the security level based on various methods of the proposed protocol. GPAPXL+ is evaluated against some well-known RFID attacks and also a simulation of the protocol was evaluated using the Scyther protocol security analysis tool developed by Cremers (2018).

Firstly, it is assumed that the attacker has following abilities, access, information and resources:

- The attacker can listen to all messages between the tags and the reader.
- The attacker may block the transmitted data on the communication channel.
- The attacker may send a message to the other party as a tag or a reader.
- The attacker has no access to hidden parameters, but can access all functions or operations, such as PRNG, encryption and XOR.

- The attacker can process all transmitted messages, such as creating, modifying and deleting them and returning them to the communication channel.

5.2.1 Traceability Attack Analysis

The traceability attack can be performed if the tag sends fixed responses to the reader. In the proposed protocol, to prevent this type of attack the tag adds the nonce value (N_t) to each message. N_t is produced by a PRNG “xorshiftL+” tested by NIST STS, as mentioned in the previous section. As well as the addition of the N_t value at the end of messages, the encrypted text is also changed in each message, since this value is also added to the messages. Therefore, the proposed protocol includes defence against traceability attack. However, it is possible to observe the group to which the tag belongs. In this case, any attacker who stores and sends a valid M_1 , nonce, and MAC value will receive a response from any tag in the group, although it cannot associate responses with a specific tag.

5.2.2 Replay Attack Analysis

It is assumed that an attacker could process or record messages transmitted during the successful authentication process in this type of attack. At the same time, when an RFID reader issues a request, it is assumed that the attacker can participate in the process instead of a valid tag. M_1 value is always changing because it includes N_r value. The attacker can attempt to intervene in this process by resending the recorded information, but cannot recreate a valid M_4 value. This is because M_4 value is encrypted, and N_t value in M_2 , one of the values that makes up M_4 , is continuously changing. Also, this attack will not be successful because previous N_t value was saved on the server, and updated in every successful authentication process.

5.2.3 Physical Disclosure Attack Analysis

In this type of attack, it is assumed that the attacker can physically gain access to a tag which is a member of the specified group, and all its hidden information. In this case, if an attacker can execute attacks such as traceability, identity impersonation, or message repetition attacks, this attack is known as a physical disclosure attack. In our proposed grouping proof authentication protocol, if able to access the tag physically,

an attacker can seize the hidden parameters of a tag, such as ID_g , k_g , ID_t and k_t . Although $M4$ sent by other tags will be decrypted using k_g , the attacker will not be able to acquire the ID_t information of other tags in the group, because each tag encrypts its ID_t with its own unique k_t . Thus, the attacker cannot produce $M4$ for a fake tag, and the proposed protocol is strong against physical disclosure attacks. It is not possible to threaten protocol security by seizing hidden parameters of a tag.

5.2.4 Impersonation Attack Analysis

In this type of attack, the attacker convinces the RFID reader that the received messages were sent by a valid tag. On the tag's side, the values of $M3$ and $M4$ are encrypted using AES algorithm. While the attacker needs ID_t and k_t values to calculate and encrypt $M3$ value, to calculate $M4$ value it needs k_g value in addition to $M3$. The attacker does not have this information, therefore the tag cannot impersonate the identity, and the protocol can resist this attack.

5.2.5 Remove Unwanted Tags Analysis

It is possible that within the interaction area of the reader, some tags do not belong to the corresponding group; therefore, it is important to prevent the authentication protocol from including valid but unwanted tags in the authentication process. First of all, our grouping proof protocol checks the membership of the tag. The reader calculates $M2^i = ID_g \oplus N_t \oplus N_r$ and decrypts $M4$ value and calculates $M4^i$ value. It calculates $M3^i$ as $M3^i = M4^i \oplus M2^i$. Finally, $M3^i$ is sent to the server, which searches for it in the database. If it does not belong to the desired group, the server sends a reject message. This is the strategy in the proposed protocol to remove valid but unwanted tags.

5.2.6 Partial-Proof Analysis

The new protocol has the ability to verify the identity of a number of tags in a group or subgroups, called partial-proof. Partial-proof is performed on the server side. The server transfers the requested group information to the RFID reader, which sends the responses of the tags to the server, and runs the authentication process. The server compares each tag in the desired group with the value of $E(ID_t, k_t)$ stored in the

database. Partial-proof is valid if the server can find the record of the desired group. Also if one or more tags are separated from the group, the server can run the authentication process based on the remaining tags. Thus, the proposed protocol has the ability to perform the authentication for groups of one or more tags, regardless of the number of tags in the groups.

5.2.7 Desynchronization Attack Analysis

In GPAPXL+ scheme, M_4 and N_t nonce value are taken by the reader and M_4^1 created using N_t and the values stored on reader side. N_t value is also used to create M_4 . M_4 value is encrypted data and can't be changed by the attackers. Therefore, we can say that proposed scheme can resist desynchronization attacks.

5.2.8 Denial of Service Attack Analysis

The proposed protocol is not compatible to the denial-of-service attacks because it has resistance to desynchronization attacks. Reader and the tag is available to accept communication all the time.

5.2.9 Man in The Middle Attack Analysis

Transferred data between the reader and the tag is encrypted by AES-256. The attacker has to break AES-256 encryption to apply Man-in-the-middle attack. This is not possible.

5.2.10 Cloning Attack Analysis

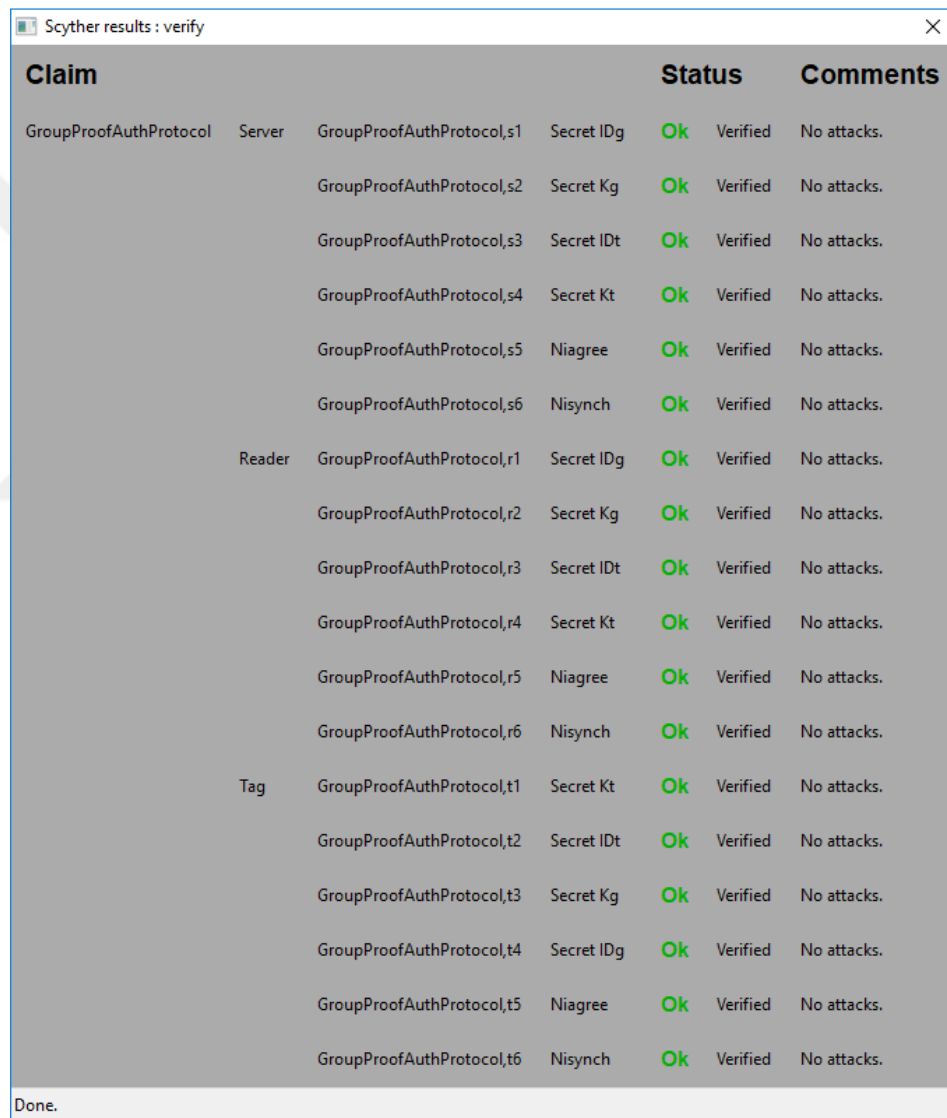
Cloning attack is not applicable for proposed scheme because obtaining the tag ID is computationally infeasible under the AES-256.

5.2.11 Eavesdropping Analysis

Attacker can listen the communication between the tag and the reader but sent critical data is encrypted so attacker can't estimate the plain text of that critical data.

5.2.12 Security Analysis with Scyther

The Scyther security analysis tool provides an interface to the user using the Python code infrastructure. This interface allows the intended users to perform security analysis of the protocol and to understand the results without difficulty. Our newly proposed protocol was implemented using Scyther tool's special programming language. Scyther tool has the ability to try multiple attacks to the authentication protocols and shows the results of these attacks.



Claim				Status	Comments
Server	GroupProofAuthProtocol,s1	Secret IDg	Ok	Verified	No attacks.
	GroupProofAuthProtocol,s2	Secret Kg	Ok	Verified	No attacks.
	GroupProofAuthProtocol,s3	Secret IDt	Ok	Verified	No attacks.
	GroupProofAuthProtocol,s4	Secret Kt	Ok	Verified	No attacks.
	GroupProofAuthProtocol,s5	Niagree	Ok	Verified	No attacks.
	GroupProofAuthProtocol,s6	Nisynch	Ok	Verified	No attacks.
Reader	GroupProofAuthProtocol,r1	Secret IDg	Ok	Verified	No attacks.
	GroupProofAuthProtocol,r2	Secret Kg	Ok	Verified	No attacks.
	GroupProofAuthProtocol,r3	Secret IDt	Ok	Verified	No attacks.
	GroupProofAuthProtocol,r4	Secret Kt	Ok	Verified	No attacks.
	GroupProofAuthProtocol,r5	Niagree	Ok	Verified	No attacks.
	GroupProofAuthProtocol,r6	Nisynch	Ok	Verified	No attacks.
Tag	GroupProofAuthProtocol,t1	Secret Kt	Ok	Verified	No attacks.
	GroupProofAuthProtocol,t2	Secret IDt	Ok	Verified	No attacks.
	GroupProofAuthProtocol,t3	Secret Kg	Ok	Verified	No attacks.
	GroupProofAuthProtocol,t4	Secret IDg	Ok	Verified	No attacks.
	GroupProofAuthProtocol,t5	Niagree	Ok	Verified	No attacks.
	GroupProofAuthProtocol,t6	Nisynch	Ok	Verified	No attacks.

Done.

Figure 5.3 Scyther security analysis tool results

Our Scyther model whose code can be seen in Appendix 6 examines three claims, namely Secret, Nisynch and Niagree. These claims can be defined as follows: Niagree

is Non-injective synchronization and ensures there is no interference in the content of the message exchanged between the receiver and sender and communication is completed in the protocol. Nisynch guarantees that communication packets are exchanged in the expected order, and also the run of protocol is Niagree (Duan, Mjøl̂snes & Tsay, 2013). "Secret claims" guarantee that the messages between the sender and the receiver are secret (Rathore & Hussain, 2015). As a result of the examination, it was determined that the protocol passed all the attack tests, as shown in the Scyther result screen in Figure 5.3. The Scyther result show that it was impossible to attack the authentication operations in each layer, therefore the authentication protocol is considered secure according to the Scyther results.

5.2.13 Security Service Analysis

The reader sends its encrypted group ID to the tag. Tag decrypts the sent ID and compare it with stored group ID. If they are equal so tag authenticates the reader. On the other hand, tag sent encrypting some it's value to the reader. The reader decrypts the sent data and asks the server if the server says everything is ok so reader authenticates the tag and the mutual authentication process completes.

Proposed protocol resists the desynchronization and denial-of-service (DoS) attacks so we can say that the availability is maintained.

The tag ID and all critical data is sent to the reader as ciphertext so it is not available without breaking the AES-256 encryption. We can say that confidentiality tag anonymity, traceability, location and information privacy are maintained by that reason.

The previous/current confidential information can't be compromised by the attacker because all critical data including the tag ID is transmitted as encrypted text. Forward and backward security is maintained.

The transfer of ownership means that the new owner's server has inherited the tag authorization and therefore the identity and the authorization must be transferred securely. In our scenario, authorization and other critical information is stored in the server's database. If this critical data transferred to other server so it is available to transfer the ownership.

5.2.14 Examination and Comparisons of Authentication Protocols

In Alaadin and Dalkılıç's (2019) article, 22 recently proposed RFID authentication protocols are compared. The GPAPXL+ was added in that tables and compared with others. The article names and publication years of the 23 protocols are presented together with their references in Table 5.20.

In table 5.21, protocols are grouped according to whether they are EPC compliant or not, the encryption methods they use, and whether they are fully fledged, simple, lightweight or ultra-lightweight. The first column in Table 5.21 lists the publication name of 23 protocols according to their classes. The classes of the protocols are grouped as fully fledged, simple, lightweight and ultra-lightweight. In the second column, the used function is listed. Validation tools are listed in the third column. The fourth column of table shows the compatibility of the protocol with EPC C1-G2. The next column gives the list of the other compared protocols. The class of the authentication protocol is given in the last column.

Protocols are also compared considering the security services and threats in Table 5.22. Security threats; DoS, Man-In-The-Middle, full disclosure attacks, eaves dropping, tracking, impersonation, De-Synchronize, cloning, replay and security services; confidentiality, availability, mutual authentication, ownership transferable, tag anonymity, forward security, backward security, location privacy, information privacy, traceability are compared in that table. The protocol presented in our thesis is a fully fledged protocol. As it can be seen in Table 5.22, GPAPXL+ provides better security than the others. It overcomes 10 threats and provides 10 security services. This protocol is also lightweight and using AES-256 to secure communication between the reader and tag.

Table 5.20 Examined protocols' reference numbers and names

No	Ref.	Paper Name and Publishing Year
1	(Zhang & Qi, 2014)	A secure RFID mutual authentication protocol for healthcare environments using elliptic curve cryptography (2015)
2	(Rostampour et al. 2014)	A novel mutual RFID authentication protocol with low complexity and high security (2014)
3	GPAPXL+	Grouping proof authentication protocol using xorshiftL+ (GPAPXL+)
4	(Farash, Nawaz, Mahmood, Chaudhry & Khan, 2016)	A provably secure RFID authentication protocol based on elliptic curve for healthcare environments (2016)
5	(Ibrahim & Dalkılıç, 2017)	An advanced encryption standard powered mutual authentication protocol based on elliptic curve cryptography for RFID, proven on WISP (2017)
6	(Dalkılıç et al., 2014)	Increasing key space at little extra cost in RFID authentications (2014)
7	(Özcanhan, Dalkılıç & Utku, 2014)	Cryptographically supported NFC tags in medication for better inpatient safety (2014)
8	(Hakeem, Raahemifar & Khan, 2013)	HPAP: A novel authentication scheme for RFID systems (2013)
9	(Srivastava, Awasthi, Kaul & Mittal, 2015)	A hash based mutual RFID tag authentication protocol in telecare medicine information system (2015)
10	(Shen, Tan, Moh, Chung & Wang, 2016)	An efficient RFID authentication protocol providing strong privacy and security (2016)
11	(Li, Zhou & Wang, 2017)	Cryptanalysis of the LMAP protocol: A low-cost RFID authentication protocol. (2017)
12	(Kardaş, Celik, Arslan & Levi, 2013)	An efficient and private RFID authentication protocol supporting ownership transfer (2013)
13	(Fan, Li, Li, Liang, Shen & Yang, 2014)	RSEL: revocable secure efficient lightweight RFID authentication scheme (2014)
14	(Ren, Xu & Li, 2013)	An one-way hash function based lightweight mutual authentication rfid protocol (2013)
15	(Maarof, Labbi, Senhadji & Belkasm, 2016)	A novel mutual authentication scheme for low-cost RFID systems (2016)
16	(Zhang, Liu, Wang, Yi & Wu, 2017)	An efficient lightweight RFID authentication protocol with strong trajectory privacy protection (2017)
17	(Pang, Li, He, Alramadhan & Wang, 2014)	Secure and efficient lightweight RFID authentication protocol based on fast tag indexing (2014)
18	(Lin & Song, 2013)	An improvement in HB-family lightweight authentication protocols for practical use of RFID system (2013)
19	(Tounsi, Cuppens-Boulahia, Garcia-Alfaro, Chevalier & Cuppens, 2014)	KEDGEN2: A key establishment and derivation protocol for EPC Gen2 RFID systems (2014)
20	(Gao, Ma, Shu & Wei, 2014)	An ultralightweight RFID authentication protocol with CRC and permutation (2014)
21	(Luo, Wen, Su & Huang, 2016)	SLAP: Succinct and lightweight authentication protocol for low-cost RFID system (2016)
22	(Tewari & Gupta, 2017)	Cryptanalysis of a novel ultra-lightweight mutual authentication protocol for IoT devices using RFID tags (2017)
23	(Kardaş, Çelik, Bingöl, Kiraz, Demirci & Levi, 2015)	K-strong privacy for radio frequency identification authentication protocols based on physically unclonable functions (2014)

Table 5.21 Techniques, verifications and classes of authentication protocols (C: class, F: fully fledged, S: simple, L: lightweight, UL: ultra-lightweight)

Protocol	Function Based on	Verified by	EPC	Compared with	C
(Zhang & Qi, 2014)	ECC		X	(He et al., 2014; Zhang & Qi, 2014; Zhao, 2014)	F
(Rostampour et al., 2014)	Public Key Cryptography	BAN logic	X	(Chen & Deng, 2009; Han, Poddar & Chang, 2007; Qian, Chen, You & Lu, 2012; Qingling, Yijiu & Yonghua, 2008; Tian, Chen & Li, 2012)	F
(Farash et al., 2016)	ECC	Random oracle model	X	(Chou, 2014; Liao & Hsiao, 2014; Zhang & Qi, 2014; Zhao, 2014)	F
(Ibrahim & Dalkılıç, 2017)	ECC & AES	Tested and realized on real devices	X	(Chou, 2014; Liao & Hsiao, 2014; Zhao, 2014; Zhang & Qi, 2014)	F
GPAPXL+	AES	Tested and realized on real device	X	(Rostampour et al., 2016)	F
(Dalkılıç, Özcanhan & Çakır, 2014)	Increasing key space using nonces	AVISPA	X	(Ha, Moon, Nieto & Boyd, 2007; Liu, 2008; Toirnuul & Lee, 2006)	F
(Özcanhan et al., 2014)	Cryptographically supported NFC tags in Medication	Manually as in (Peris-Lopez, Safkhani, Bagheri & Naderi, 2013)	X	(Peris-Lopez et al., 2013)	F
(Hakeem et al., 2013)	One-way hash function and semi randomized encryption keys	Manually as in (Changqing, Jixiong, Zhengyan & Shengye, 2008)	X	(Changqing et al., 2008; Chatmon, van Le & Burmester, 2006; Tsudik, 2007; Tuyls & Batina, 2006; Weis et al., 2004)	F
(Srivastava et al., 2015)	Hash based		X	(Chatmon et al., 2006; Cho, Yeo & Kim, 2011; Tsudik, 2007)	S
(Shen et al., 2016)	Hash based	Manually	X	(Chien, 2007; He, Jin, Zhang & Li, 2009; Peris-Lopez, Hernandez-Castro, Estevez-Tapiador & Ribagorda, 2006a; Peris-Lopez, Hernandez-Castro, Estevez-Tapiador & Ribagorda, 2006b; Peris-Lopez, Hernandez-Castro, Estevez-Tapiador & Ribagorda, 2006c; Rahman, Soshi & Miyaji, 2009; Tan, Sheng & Li, 2008)	S
(Li et al., 2017)	Hash and PRNG based	GNV logic (Zhu, Yang & Zhang, 2007)	✓	(Cho, Jeong & Park, 2015; Jeon & Yoon, 2013; Mujahid, Najam-ul-Islam & Shami, 2015)	S
(Kardaş et al., 2013)	Constant-time complexity	Byzantine adversarial model (Changqing et al., 2008)	X	(Kardaş, Levi & Murat, 2011)	S

Table 5.21 continues

Protocol	Function Based on	Verified by	EPC	Compared with	C
(Fan et al., 2014)	Hash operation & RNG	GNV logic (Zhu et al, 2007)	X	(Gao, Xiang, Wang, Shen, Huang & Song, 2004; Henrici & Muller, 2004; Li & Ding, 2007; Molnar & Wagner, 2004; Ohkubo, Suzuki & Kinoshita, 2004; Sarma, Weis & Engels, 2003; Weis et al., 2004)	S
(Ren et al., 2013)	One-way Hash Function	GNV logic (Zhu et al, 2007)	X	(Ning, Liu, Mao & Zhang, 2011; Song & Mitchell, 2008; Weis et al., 2004)	S
(Maarof et al., 2016)	XOR, PRNG and CRC based	Manually	✓	(Chen & Deng, 2009; Huang & Jiang, 2012).	L
(Zhang et al., 2017)	XOR and PRNG based	Manually as (Zhang, Wu, Liu, Huang, Guo & Hsu, 2016)	✓	(Alomair, Clark, Cuellar & Poovendran, 2012; Ha, Ha, Moon & Boyd, 2006; Habibi, Alagheband & Aref, 2011; Xiao, Zhou, Zhou & Niu, 2013)	L
(Pang et al., 2014)	Fast tag indexing, CRC & PRNG	A Simulation Program	✓	(Choi, Lee & Lee, 2005; Dimitriou, 2005; Henrici & Muller, 2004; Song & Mitchell, 2008; Weis et al., 2004; Zhang, Zhou & Luo, 2008; Zhou, Zhang, Luo & Wong, 2010)	L
(Lin & Song, 2013)	Learning parity with noise	Manually as in (Juels & Weis, 2005)	✓	(Hopper & Blum, 2001; Juels & Weis, 2005; Leng, Mayes & Markantonakis, 2008; Munilla & Peinado, 2007)	L
(Tounsi et al., 2014)	A Pseudo Random Generator Shared Between Readers and Tags	AVISPA	✓	(Asadpour & Dashedi, 2011; Burmester & Munilla, 2011; Brusó, Chatzikokolakis & Den Hartog, 2010; Hanatani, Ohkubo, Matsuo, Sakiyama & Ohta, 2012; Kim, Oh, Kim, Jeong & Choi, 2008; Van Le, Burmester & De Medeiros, 2007)	L
(Gao et al., 2014)	CRC and permutation	Simple Promela Interpreter (SPIN)	✓	(Chien, 2007; Peris-Lopez et al., 2006a; Peris-Lopez, Hernandez-Castro, Estevez-Tapiador & Ribagorda, 2009a; Tian et al., 2012)	L
(Luo et al., 2016)	A security ultralightweight bitwise conversion	Manually	✓	(Chien, 2007; Mujahid et al., 2015; Peris-Lopez et al., 2006b; Peris-Lopez et al., 2006c; Peris-Lopez et al., 2009c; Tian et al., 2012; Zhuang, Zhu & Chang, 2014)	UL
(Tewari & Gupta, 2017)	XOR bitwise rotation based	Manually	✓	(Chien, 2007; Peris-Lopez et al., 2006a; Peris-Lopez et al., 2006b; Peris-Lopez et al., 2006c; Peris-Lopez et al., 2009a; Tian et al., 2012)	UL
(Kardaş et al., 2015)	Physically Uncloenable Functions PUF		✓	(Bassil, El-Beaini, Itani, Kayssi & Chehab, 2012; Ranasinghe, Engels & Cole, 2004; Tuyls & Batina, 2006)	UL

Table 5.22 Comparison of APs in terms of security threats and security services (F: fully fledged, S: simple, L: lightweight, UL: ultra-lightweight)

Protocol No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
SECURITY THREATS																							
Tracking attack			✓		✓			✓		✓		✓			✓					✓			
DoS	✓	✓	✓		✓	✓	✓	✓						✓				✓		✓			✓
De- Synchronize	✓	✓	✓		✓	✓	✓	✓	✓			✓			✓		✓	✓		✓			
Man-In-The-Middle	✓	✓	✓	✓	✓	✓	✓	✓								✓		✓			✓		
Impersonation attack	✓		✓	✓	✓		✓	✓	✓	✓			✓		✓			✓					✓
Cloning attack	✓		✓		✓		✓			✓			✓	✓	✓		✓						
Full disclosure attack			✓		✓		✓		✓											✓	✓		
Eaves dropping		✓	✓		✓				✓									✓	✓				
Replay attack	✓	✓	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓		✓		✓	✓
SECURITY SERVICES																							
Mutual authentication	✓	✓	✓	✓	✓	✓	✓	✓				✓		✓	✓				✓			✓	✓
Confidentiality	✓	✓	✓		✓	✓	✓	✓			✓							✓		✓		✓	
Availability	✓		✓		✓																		
Forward security	✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓		✓		✓		✓				
Backward security			✓	✓	✓						✓	✓					✓		✓				✓
Ownership transferable			✓									✓											
Tag anonymity	✓		✓		✓																	✓	
Traceability	✓	✓	✓	✓	✓		✓		✓		✓	✓											
Location privacy			✓	✓	✓			✓									✓						✓
Information privacy	✓		✓		✓	✓		✓									✓		✓				
CLASS																							
Class	F	F	F	F	F	F	F	F	S	S	S	S	S	S	L	L	L	L	L	L	UL	UL	UL

CHAPTER SIX

CONCLUSION

This thesis presents a proposal for a solution for existing and pending security challenges on lightweight devices in consideration of resource and time constraints. New solutions are offered for random number generation and authentication protocol, which are the two key points of security for lightweight devices. The WISP passive RFID tag was used as the lightweight device to conduct tests and experiments. WISP has built-in sensors, 256-bit AES encryption and can be programmed, and so was selected for its high usability and applicability in the scope of future technologies.

Firstly, WISP's built-in hardware were tried to be used to generate random numbers. Temperature and accelerometer sensors, analog to digital converter and built-in clocks were used to generate the signals. These true random sources had bad results from randomness test suites so they are not suitable to be used directly for a random number source. They can be used to generate seed values for pseudo random number generators.

Secondly, a new pseudo-random number generator using linear feedback shift registers by merging was created. The newly generator was tested using the NIST test package and a complete randomness was obtained. One million numbers generated from the C programming language's rand() function were tested by NIST. Then two 16-bit numbers were merged to produce a 32-bit number and tested in the NIST test package. When NIST results were considered, it can be seen that it is failed in all tests. On the other hand, if 1,000,000 random numbers were generated through the new method presented in this thesis, we see that all tests were successful. In the same way, number generation time was estimated and it took 57 ms to generate a number on WISP. The results show us this generator is suitable to be used for so many IoT devices, RFID tags and WISP. This generator also produces number series fast and secure. The fact that the operations performed here are bitwise, this gives great advantages in terms of performance. In other words, a significant improvement has been made on communication security with this generator. In the authentication

process, more secure, faster and lightweight solution was obtained with the numbers produced by the generator.

Thirdly, xorshift algorithm was used and a new random number generator is tried to be created by using its structure. For the new modified version of the xorshift+ was called XorshiftR+. According to our experiments, the xorshiftR+ (also mentioned as reduced xorshift+ variation 3, R stands for reduced, pronounced xorshifter plus) has demonstrated outstanding results when compared to other candidates, in terms of both randomness and speed performance. It passes all the tests in the Big Crush battery of TestU01, in all runs and is slightly (but noticeably) superior to its ancestors, especially the xorshift+, with its higher speed. This superiority becomes even more obvious on lightweight environments. It also occupies less memory due to its compacted nature. Good to mention that it also passes all tests of NIST and ENT suites and complies with EPC-Gen 2 Class 1 security standards. This RNG, xorshiftR+, is strongly recommended to use in both powerful computer environments and lightweight devices, where high level of randomness and temporal performance are desired. As an addition to the conclusion given in Vigna's study in 2007, our experiments showed that, even if integer sizes are set to 32 bits (this makes 64-bit input/seed and 32-bit output), xorshift+ and xorshiftR+ can still pass all TestU01 tests in all runs systematically. LFSR113 on the other hand, was the fastest on PC, though wasn't able to pass all tests. Plus, it occupies vast memory. As its original form (used here), it can only be recommended when randomness requirements are loose but time limitations are dominant. If it is possible to make it fail-free hereafter, then it might be a good alternative to xorshiftR+. Such improvements on the LFSR could definitely be a worthy future work.

Fourth, a new HRNG was developed by reducing the bitwise operations of the original xorshift and using a TRNG as a seed. Marsaglia's well-known Mersenne Twister based xorshift random number generator was modified to produce a new PRNG. To initialize the PRNG, WISP RFID tag's temperature sensor was used to obtain a true random number by performing 16 temperature samplings in a cycle. Our newly created HRNG was proposed as a combination of a PRNG and a TRNG. The time required for random number generation using the HRNG was estimated and

compared with some of previous well-known random number generators shown in Table 5.16 in the previous section. A similar approach was taken to investigating whether the random number generator satisfies 3 conditions for EPC™ Gen-2 Class 1 standards. The quality of the random number series generated by the HRNG was examined using the NIST test suite. These tests and evaluations revealed that the new HRNG satisfies 3 conditions for EPC™ Gen-2 Class 1 standards, passes all NIST test suite tests and generates random numbers approximately 16% faster than the closest rival.

Fifth, genetic algorithm used to create random numbers. At first, random numbers were generated by using hardware sources. Numbers generated from hardware sources were given to genetic algorithm as an individual and created new generations using these individuals by passing the selection, elitism and mutation operations. As a result, this method was failed. Then a new PRNG was tried to be created by using genetic algorithm. Logical operators are selected as inputs of the genetic algorithm and entropy calculation was used as fitness function. This method generated PRNGs. When the generated PRNG's outputs were tested, some of them were successful at NIST's statistical tests. It is important to note that passing NIST STS does not prove that it is safe to use this PRNG in every cryptographic application. For future works, other statistical properties can be combined with the bit entropy calculation to improve the fitness function and generate better PRNGs.

Finally, previous authentication protocols were examined in the scope of this thesis and grouping proof authentication protocols were considered capable of obtaining effective and flexible solutions under the desired resource limitations. This resulted in the development of the new grouping proof authentication protocol discussed in this thesis. A secure solution is produced using WISP's built-in 256 bit AES encryption with this new protocol and xorshiftL+, proposed in this thesis, is used for random number generation required for nonce values on both RFID reader and RFID tag. This protocol was called as GPAPXL+. This thesis describes the protocol's structural design, phases and process steps, and also provides the security analysis of the protocol based on an evaluation of its effectiveness against known attack methods. In addition,

protocol analysis was performed by using the Scyther security analysis tool. It was observed that the protocol successfully prevented from well-known attacks.



REFERENCES

- AD2210 (2019). Retrieved April 4, 2019, from <http://www.analog.com/media/en/technical-documentation/data-sheets/AD22100.pdf>.
- ADXL362 (2019). Retrieved April 4, 2019, from <http://www.analog.com/en/products/mems/mems-accelerometers/adxl362.html#product-overview>.
- ADXL362, (2016). *Micropower, 3-Axis, ± 2 g/ ± 4 g/ ± 8 g digital output MEMS accelerometer*, Retrieved April 4, 2019, from <http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL362.pdf>.
- Ahmadian, Z., Salmasizadeh, M., & Aref, M. R. (2013). Desynchronization attack on RAPP ultralightweight authentication protocol. *Information Processing Letters*, 113(7), 205-209.
- Ahson, S. A., & Ilyas, M. (2008). *RFID handbook: applications, technology, security, and privacy*. Florida: CRC press.
- Alder, G. S. (1998). Ethical issues in electronic performance monitoring: A consideration of deontological and teleological perspectives. *Journal of Business Ethics*, 17(7), 729-743.
- Alomair, B., Clark, A., Cuellar, J., & Poovendran, R. (2012). Scalable RFID systems: a privacy-preserving protocol with constant-time identification. *IEEE Transactions on Parallel and Distributed Systems*, 23(8), 1536-1550.
- Andre, M. (2019). *DASH7 alliance presentations*. Retrieved April 4, 2019, from <http://www.dash7-alliance.org/presentations/>.

- Arnault, F., Berger, T., Minier, M., & Pousse, B. (2011). Revisiting LFSRs for cryptographic applications. *IEEE Transactions on Information Theory*, 57(12), 8095-8113.
- Asadpour, M., & Dashti, M. T. (2011). A privacy-friendly RFID protocol using reusable anonymous tickets. In *IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 206-213. IEEE.
- ASTM International (2019). *Detailed overview*. Retrieved April 4, 2019, from https://www.astm.org/ABOUT/full_overview.html.
- Atlas RFID Solutions (2019). *What is RFID? | The beginner's guide to RFID systems*. Retrieved April 4, 2019, from <https://www.atlasrfidstore.com/rfid-beginners-guide/>.
- Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. *Computer networks*, 54(15), 2787-2805.
- Aumasson, J. P., Henzen, L., Meier, W., & Naya-Plasencia, M. (2013). Quark: A lightweight hash. *Journal of Cryptology*, 26(2), 313-339.
- Aydın, Ö., & Dalkılıç, G. (2016). Nesnelerin İnterneti için Sözderastsal Sayı Üretici: Birleştirilmiş Doğrusal Geri Beslemeli Öteleyici Saklayıcı. In *Akıllı Teknoloji & Akıllı Yönetim* (121-129), İzmir: Gülermat Matbaacılık.
- Aydın, Ö., & Dalkılıç, G. (2018). A hybrid random number generator for lightweight Cryptosystems: Xorshiftplus. In *3rd International Conference on Engineering Technology and Applied Sciences (ICETAS)*, 47-47.
- Barber, G., & Tsibertzopoulos, E. (2005). An analysis of using EPCglobal class-1 generation-2 RFID technology for wireless asset management. In *MILCOM 2005-2005 IEEE Military Communications Conference*, 245-251. IEEE.
- Barker, E., & Kelsey, J. (2012) *Recommendation for random number generation using deterministic random bit generators, NIST special publication 800-90A*. Retrieved

- April 4, 2019, from <https://csrc.nist.gov/publications/detail/sp/800-90a/archive/2012-01-23>.
- Barker, E., & Kelsey, J. (2016) *Recommendation for random bit generator (RBG) constructions, NIST special publication 800-90C second draft*. Retrieved April 4, 2019, from https://csrc.nist.gov/CSRC/media/Publications/sp/800-90c/draft/documents/sp800_90c_second_draft.pdf.
- Brassard, G., & Crépeau, C. (1996). 25 years of quantum cryptography. *ACM Sigact News*, 27(3), 13-24.
- Bassham, L., Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., ... & Dray, J. (2010). *San Vo. A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications*. Retrieved April 4, 2019, from <http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf>.
- Bassil, R., El-Beaino, W., Itani, W., Kayssi, A., & Chehab, A. (2012). PUMAP: A PUF-based ultra-lightweight mutual authentication RFID protocol. *International Journal of RFID Security and Cryptography*, 1(1/2), 58-66.
- Bayrak Meydanoğlu, E. S. (2008). RFID Sistemleri ve Veri Güvenliği. *Bilişim Teknolojileri Dergisi*, 1(3), 57.
- Bazaatı, S. (2012). *İnşaat sektöründe radyo frekanslı tanımlama (RFID) (Doctoral dissertation)*. Yüksek Lisans Tezi, Çukurova Üniversitesi Fen Bilimleri Enstitüsü, Adana.
- Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., & Wingers, L. (2013). *The SIMON and SPECK families of lightweight block ciphers cryptology*. ePrint Archive. <http://eprint.iacr.org/2013/404>.
- Bhuptani, M., & Moradpour, S. (2005). *RFID field guide: deploying radio frequency identification systems*. New Jersey: Prentice Hall PTR.

- Blackman, D., & Vigna, S. (2018). *Scrambled linear pseudorandom number generators*. arXiv preprint arXiv:1805.01407.
- Boer Barkod&RFID (2019). Retrieved April 3, 2019, from <https://www.boer.com.tr/tr/b-80-0-rfid-antenler.asp>.
- Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varıcı, K., & Verbauwhede, I. (2011). SPONGENT: A lightweight hash function. In *International Workshop on Cryptographic Hardware and Embedded Systems*, 312-325. Berlin: Springer.
- Bolic, M., Simplot-Ryl, D., & Stojmenovic, I. (Eds.). (2010). *RFID systems: research trends and challenges*, (450-450), United Kingdom: John Wiley & Sons.
- Boularess, O., Rmili, H., Aguilı, T., & Tedjini, S. (2015). Analysis of electromagnetic signature of Arabic alphabet as RF elementary coding particles. *Wireless Power Transfer*, 2(2), 97-106.
- Bowden, L. (1985). The story of IFF (identification friend or foe). *IEE Proceedings A (Physical Science, Measurement and Instrumentation, Management and Education, Reviews)*, 132(6), 435-437.
- Brent, R. P. (2007). Some long-period random number generators using shifts and xors. *ANZIAM Journal*, 48, 188-202.
- Bringer, J., Chabanne, H., & Dottax, E. (2006). HB++: A lightweight authentication protocol secure against some attacks. In *Second international workshop on security, privacy and trust in pervasive and ubiquitous computing, 2006*. 28–33. IEEE.
- Brown, D. (2006). *RFID implementation*. New York: McGraw-Hill.
- Brown, R. G., Eddelbuettel, D., & Bauer, D. (2013). *Dieharder: A random number test suite. Open source software library, under development*, Retrieved April 4, 2019, from <http://webhome.phy.duke.edu/~rgb/General/dieharder.php>.

- Brusó, M., Chatzikokolakis, K., & Den Hartog, J. (2010). Formal verification of privacy for RFID systems. In *23rd IEEE Computer Security Foundations Symposium*, 75-88. IEEE.
- Burmester, M., & Munilla, J. (2011). Lightweight RFID authentication with forward and backward security. *ACM Transactions on Information and System Security (TISSEC)*, 14(1), 11-16.
- Çabuk, U. C., Aydın, Ö., & Dalkılıç, G. (2017). A random number generator for lightweight authentication protocols: xorshiftR+. *Turkish Journal of Electrical Engineering & Computer Sciences*, 25(6), 4818-4828.
- Castelluccia, C., & Soos, M. (2007). Secret shuffling: A novel approach to RFID private identification. In *Conference on RFID security*, 169-180.
- Castro, L., & Wamba, S. F. (2007). An inside look at RFID technology. *Journal of technology management & innovation*, 2(1), 128-141.
- Chaitin, G. J. (2012). *Exploring randomness*. London: Springer Science & Business Media.
- Challa, S., Wazid, M., Das, A. K., & Khan, M. K. (2018). Authentication protocols for implantable medical devices: taxonomy, analysis and future directions. *IEEE consumer electronics magazine*, 7(1), 57-65.
- Changqing, O., Jixiong, W., Zhengyan, L., & Shengye, H. (2008). An enhanced security authentication protocol based on hash-lock for low-cost RFID. In *Anticounterfeiting, Security and Identification, 2008. 2nd International Conference on ASID 2008*, 416-419. IEEE.
- Chatmon, C., van Le, T., & Burmester, M. (2006). *Secure anonymous RFID authentication protocols*, Technical Report, Florida State University, Department of Computer Science, USA.

- Che, W., Deng, H., Tan, W., & Wang, J. (2008). A random number generator for application in RFID tags. In *Networked RFID Systems and Lightweight Cryptography*, (279-287). Berlin: Springer.
- Chen, C. L., & Deng, Y. Y. (2009). Conformation of EPC class 1 generation 2 standards RFID system with mutual authentication and privacy protection. *Engineering Applications of Artificial Intelligence*, 22(8), 1284-1291.
- Chien, H. Y. (2007). SASI: A new ultralightweight RFID authentication protocol providing strong authentication and strong integrity. *IEEE Transactions on Dependable and Secure Computing*, 4(4), 337-340.
- Chien, H. Y., & Chen, C. H. (2007). Mutual authentication protocol for RFID conforming to EPC Class 1 Generation 2 standards. *Computer Standards & Interfaces*, 29(2), 254-259.
- Chiesa, M., Genz, R., Heubler, F., Mingo, K., Noessel, C., Sopiaeva, N., Slocombe, D., & Tester, J. (2002). *RFID a week long survey on the technology and its potential*. Retrieved April 4, 2019, from http://people.interactionivrea.org/c.noessel/RFID/RFID_research.pdf.
- Chlumecký, M., Buchtele, J., & Richta, K. (2017). Application of random number generators in genetic algorithms to improve rainfall-runoff modelling. *Journal of Hydrology*, 553, 350-355.
- Cho, J. S., Jeong, Y. S., & Park, S. O. (2015). Consideration on the brute-force attack cost and retrieval cost: A hash-based radio-frequency identification (RFID) tag mutual authentication protocol. *Computers & Mathematics with Applications*, 69(1), 58-65.
- Cho, J. S., Yeo, S. S., & Kim, S. K. (2011). Securing against brute-force attack: A hash-based RFID mutual authentication protocol using a secret value. *Computer Communications*, 34(3), 391-397.

- Choi, E. Y., Lee, S. M., & Lee, D. H. (2005). Efficient RFID authentication protocol for ubiquitous computing environment. In *Embedded and Ubiquitous Computing—EUC 2005 Workshops*, 945-954. Berlin: Springer.
- Chou, J. S. (2014). An efficient mutual authentication RFID scheme based on elliptic curve cryptography. *The Journal of Supercomputing*, 70(1), 75-94.
- Cisco Systems (2008). *RFID Tag Considerations. Wi-Fi Location-Based Services 4.1 Design Guide*. Retrieved April 3, 2019, from <https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Mobility/WiFiLBS-DG.html>
- Cookman, R. (2016). *Random cycle bit generator (rcb_generator)*. Technical Report. Retrieved April 4, 2019, from <https://github.com/ceorron/cycle-bit-random-number-generator>.
- Couture, R., & L'Ecuyer, P. (1997). Distribution properties of multiply-with-carry random number generators. *Mathematics of Computation of the American Mathematical Society*, 66(218), 591-607.
- Cryptography timeline. (2004). Retrieved April 4, 2019, from <http://world.std.com/~cme/html/timeline.html>.
- Dalkılıç, G. (2016). Radyo frekansı ile tanımlama etiketleri için gerçek rastgele sayı tabanlı üreteç. *Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi*, 18(54), 640-651.
- Dalkılıç, G., Özcanhan, M. H., & Çakır, H. Ş. (2014). Increasing key space at little extra cost in RFID authentications. *Turkish Journal of Electrical engineering & computer sciences*, 22(1), 155-165.
- Davis, L. (1991). *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold.
- De Canniere, C., Dunkelman, O., & Knežević, M. (2009). KATAN and KTANTAN—a family of small and efficient hardware-oriented block ciphers. In *International*

- Workshop on Cryptographic Hardware and Embedded Systems*, 272-288. Berlin: Springer.
- de Fuentes, J. M., Peris-Lopez, P., Tapiador, J. E., & Pastrana, S. (2015). Probabilistic yoking proofs for large scale IoT systems. *Ad Hoc Networks*, 32, 43-52.
- Demirel, F. (2007). *Tedarik ve lojistik yönetiminde RFID uygulamaları*. İstanbul: İstanbul Ticaret Üniversitesi, 44.
- Diffie, W., & Hellman, M. E. (1979). Privacy and authentication: An introduction to cryptography. *Proceedings of the IEEE*, 67(3), 397-427.
- Dilonardo, R. L. (2014). *EAS source tagging 20-plus years of innovation*. *Loss Prevention Magazine*. Retrieved April 4, 2019, from <https://losspreventionmedia.com/electronic-article-surveillance-eas-source-tagging/>.
- Dimitriou, T. (2005). A lightweight RFID protocol to protect against traceability and cloning attacks. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, 59-66. IEEE.
- DLR-TL001 (2019) Retrieved April 3, 2019, from <https://www.datalogic.com/eng/transportation-logistics/rfid-systems/dlr-tl001-pd-706.html>.
- Duan, S., Mjølunes, S. F., & Tsay, J. K. (2013). Security analysis of the Terrestrial Trunked radio (TETRA) authentication protocol, In *Norwegian Information Security Conference*, Stavanger, Norway.
- Duc, D., Park J., & Lee, H. (2006). Enhancing security of EPCglobal gen-2 RFID tag against traceability and cloning. In *2006 Symposium on Cryptography and Information Security*, 17–20.
- Dutang, C., & Wuertz, D. (2009). *A note on random number generation. Overview of Random Generation Algorithms*. Retrieved April 4, 2019, from <https://cran.r-project.org/web/packages/randtoolbox/vignettes/fullpres.pdf>

- Eastlake, D., Crocker, S., & Schiller, J. (1994). *Randomness recommendations for security-RFC 1750*. Internet Engineering Task Force. Retrieved April 4, 2019, from https://datatracker.ietf.org/doc/rfc1750/?include_text=1
- Eichenauer, J., & Lehn, J. (1986). A non-linear congruential pseudo random number generator. *Statistische Hefte*, 27(1), 315-326.
- Elbal, B. R. (2015). *Measurement Based Evaluation of the Wireless Identification and Sensing Platform*. Retrieved April 4, 2019, from http://publik.tuwien.ac.at/files/PubDat_240043.pdf, Master Thesis, Vienna University of Technology, Austria.
- EPCGlobal GS1, (2015). *EPC™ Radio-Frequency Identity Protocols Generation-2 UHF RFID Specification for RFID Air Interface Protocol for Communications at 860 MHz – 960 MHz Version 2.0.1 Ratified*. Retrieved April 4, 2019, from https://www.gs1.org/sites/default/files/docs/epc/Gen2_Protocol_Standard.pdf
- EPCglobal Inc. (2007). *EPCglobal Standard Specification - GS1*. Retrieved April 4, 2019, from https://www.gs1.org/sites/default/files/docs/epc/uhfc1g2_1_1_0-standard-20071017.pdf.
- Ergen, E. (2008). İnşaat sektöründe radyo frekanslı tanımlama (RFID) teknolojisi uygulamaları. *Türkiye Mühendislik Haberleri Dergisi*, 53 (451), 44-48.
- Fan, K., Li, J., Li, H., Liang, X., Shen, X. S., & Yang, Y. (2014). RSEL: revocable secure efficient lightweight RFID authentication scheme. *Concurrency and Computation: Practice and Experience*, 26(5), 1084-1096.
- Farash, M. S., Nawaz, O., Mahmood, K., Chaudhry, S. A., & Khan, M. K. (2016). A provably secure RFID authentication protocol based on elliptic curve for healthcare environments. *Journal of Medical Systems*, 40(7), 165.
- Federal Information Processing Standard [FIPS] (1999). *Data Encryption Standard (DES)*, FIPS PUB 46-3, US Department of Commerce, National Institute of Standards and Technology.

- Federal Information Processing Standard [FIPS] (2001). *Advanced Encryption Standard (AES)*, FIPS PUB 197, National Institute of Standards and Technology.
- Feldhofer, M., Dominikus, S., & Wolkerstorfer, J. (2004). Strong authentication for RFID systems using the AES algorithm. In M. Joye & J.-J. Quisquater (Eds.), *Cryptographic hardware and embedded systems CHES 2004*. (357–370). Berlin: Springer.
- Feldhofer, M., & Wolkerstorfer, J. (2009). Hardware Implementation of Symmetric Algorithms for RFID Security. *RFID Security*, 373-415.
- Finkenzeller, K. (2003). *RFID handbook: fundamentals and applications*. United Kingdom: John Wiley and Sons.
- Finkenzeller, K. (2005). *RFID handbook*, New York: Wiley Press.
- Fischer, V., & Bernard, F. (2011). True random number generators in FPGAs. In *Security Trends for FPGAs*, 101-135. Dordrecht: Springer.
- Fips 202, (2015). *SHA-3 standard: permutation-based hash and extendable-output functions*. Retrieved April 4, 2019, from <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf>.
- Fujdiak, R., Misurec, J., Mlynek, P., & Raso, O. (2013a). Random number generator in MSP430 x5xx families. *Elektrorevue*, 4(4), 70-74.
- Fujdiak, R., Misurec, J., Mlynek, P., & Raso, O. (2013b). Cryptography in ultra-low power Microcontroller MSP430. *International Journal of Engineering Trends and Technology (IJETT)*, 6(8), 398-404.
- Galton, F. (1890). Dice for statistical experiments. *Nature*, 42, 13–14.
- Gao, L., Ma, M., Shu, Y., & Wei, Y. (2014). An ultralightweight RFID authentication protocol with CRC and permutation. *Journal of Network and Computer Applications*, 41, 37-46.

- Gao, X., Xiang, Z., Wang, H., Shen, J., Huang, J., & Song, S. (2004). An approach to security and privacy of RFID system for supply chain. In *IEEE International Conference on E-Commerce Technology for Dynamic E-Business*, 164-168. IEEE.
- Garfinkel, S. L., Juels, A., & Pappu, R. (2005). RFID privacy: An overview of problems and proposed solutions. *IEEE Security & Privacy*, 3(3), 34-43.
- Gilbert, H., Robshaw, M., & Sibert, H. (2005). Active attack against HB+: A provably secure lightweight authentication protocol. *Electronics Letters*, 41(21), 1169–1170.
- Glasser, D. J., Goodman, K. W., & Einspruch, N. G. (2007). Chips, tags and scanners: Ethical challenges for radio frequency identification. *Ethics and Information Technology*, 9(2), 101-109.
- Glover, B., & Bhatt, H. (2006). *RFID essentials*. USA: O'Reilly Media.
- Golomb, S. W. (1982). *Shift register sequences*. California, Laguna Hills: Aegean Park Press.
- Gong, Z., Nikova, S., & Law, Y. W. (2011). KLEIN: a new family of lightweight block ciphers. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, 1-18. Berlin: Springer.
- Goresky, M., & Klapper, A. (2006). Pseudonoise sequences based on algebraic feedback shift registers. *IEEE Transactions on Information Theory*, 52(4), 1649-1662.
- GS1 (2019). *EPC/RFID Standards*. Retrieved April 4, 2019, from <https://www.gs1.org/standards/epc-rfid>.
- Guo, J., Peyrin, T., & Poschmann, A. (2011). The PHOTON family of lightweight hash functions. In *Annual Cryptology Conference*, 222-239. Berlin: Springer.
- Guo, J., Peyrin, T., Poschmann, A., & Robshaw, M. (2011). The LED block cipher. In *International Workshop on Cryptographic Hardware and Embedded Systems* 326-341. Berlin: Springer.

- Ha, J. C., Ha, J. H., Moon, S. J., & Boyd, C. (2006). LRMAP: Lightweight and resynchronous mutual authentication protocol for RFID system. In *International Conference on Ubiquitous Convergence Technology*, 80–89. Berlin: Springer.
- Ha, J., Moon, S., Nieto, J. M. G., & Boyd, C. (2007). Low-cost and strong-security RFID authentication protocol. In *International Conference on Embedded and Ubiquitous Computing*, 795-807. Springer, Berlin, Heidelberg.
- Habibi, M. H., Alagheband, M. R., & Aref, M. R. (2011). Attacks on a lightweight mutual authentication protocol under EPC C-1 G-2 standard. In *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication*, 254-263. Berlin: Springer.
- Hakeem, M. J., Raahemifar, K., & Khan, G. N. (2013). HPAP: A novel authentication scheme for RFID systems. In *2013 26th Annual IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 1-6. IEEE.
- Han, S., Potdar, V., & Chang, E. (2007). Mutual authentication protocol for RFID tags based on synchronized secret information with monitor. In *International Conference on Computational Science and its Applications*, 227-238. Berlin: Springer.
- Hanatani, Y., Ohkubo, M., Matsuo, S. I., Sakiyama, K., & Ohta, K. (2012). A study on computational formal verification for practical cryptographic protocol: the case of synchronous RFID authentication. In *Financial Cryptography and Data Security*, 70-87. Berlin: Springer.
- Harrison, M., McFarlane, D., Parlikad, A. K., & Wong, C. Y. (2004). Information management in the product lifecycle-the role of networked RFID. In *2nd IEEE International Conference on Industrial Informatics, 2004. INDIN'04 2004*, 507-512. IEEE.
- Haykin, S. (2001). Pseudo-noise sequences. In *Communication Systems*. John Wiley & Sons, Inc.

- He, D., Kumar, N., Chilamkurti, N., & Lee, J. H. (2014). Lightweight ECC based RFID authentication integrated with an ID verifier transfer protocol. *Journal of medical systems*, 38(10), 116.
- He, L., Jin, S. H., Zhang, T., & Li, N. N. (2009). An enhanced 2-pass optimistic anonymous RFID authentication protocol with forward security. In *5th International Conference on Wireless Communications, Networking and Mobile Computing*, 1-4. IEEE.
- Hellekalek, P. (1998). Good random number generators are (not so) easy to find. *Mathematics and computers in simulation*, 46(5-6), 485-505.
- Hennebert, C., Hossayni, H., & Lauradoux, C. (2013). Entropy harvesting from physical sensors. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, 149-154. ACM.
- Henrici, D., & Muller, P. (2004). Hash-based enhancement of location privacy for radio frequency identification devices using varying identifiers. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, 149-153. IEEE.
- Hernandez, J. C., Sez nec, A., & Isasi, P. (2004). On the design of state-of-the-art pseudorandom number generators by means of genetic programming. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, 2, 1510-1516. IEEE.
- Heys, H. M. (2002). A tutorial on linear and differential cryptanalysis. *Cryptologia*, 26(3), 189-221.
- Heys, H. M., & Tavares, S. E. (1996). Substitution-permutation networks resistant to differential and linear cryptanalysis. *Journal of Cryptology*, 9(1), 1-19.
- Hopper, N. J., & Blum, M. (2001). Secure human identification protocols. In *Advances in Cryptology—ASIACRYPT 2001*, 52-66. Berlin: Springer.

- Hossain, M. S. S., & Karmakar, N. (2006). An overview on RFID frequency regulations and antennas. In *2006 International Conference on Electrical and Computer Engineering*, 424-427. IEEE.
- Hsi, C. T., Lien, Y. H., Chiu, J. H., & Chang, H. K. C. (2015). Solving scalability problems on secure RFID grouping-proof protocol. *Wireless Personal Communications*, 84(2), 1069-1088.
- Huang, P., Mu, H., & Zhang, C. (2014). A new lightweight RFID grouping proof protocol. In *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, 869-876. Dordrecht: Springer.
- Huang, Y. C., & Jiang, J. R. (2012). An ultralightweight mutual authentication protocol for EPC C1G2 RFID tags. In *2012 Fifth International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, 133-140. IEEE.
- Ibrahim, A., & Dalkılıç, G. (2019). Review of different classes of RFID authentication protocols. *Wireless Networks*, 25:961–974.
- Ibrahim, A., & Dalkılıç, G. (2017). An advanced encryption standard powered mutual authentication protocol based on elliptic curve cryptography for RFID, proven on WISP. *Journal of Sensors*, 2017, 1-10.
- Jenkins, B. (2009). *A small noncryptographic PRNG*. Retrieved April 4, 2019, from <https://burtleburtle.net/bob/rand/smallprng.html>.
- Jeon, I. S., & Yoon, E. J. (2013). A new ultralightweight RFID authentication protocol using merge and separation operations. *International Journal of Mathematical Analysis*, 7(52), 2583-2593.
- Juels, A. (2006). FID security and privacy: A research survey. *IEEE Journal on Selected Areas in Communications*, 24(2), 381-394.
- Juels, A., & Weis, S. A. (2005). Authenticating pervasive devices with human protocols. In *Advances in Cryptology—CRYPTO 2005*, 293-308. Berlin: Springer.

- Kardaş, S., Levi, A., & Murat, E. (2011). Providing resistance against server information leakage in RFID systems. In *New Technologies, 4th IFIP International Conference on Mobility and Security (NTMS)*, 1-7. IEEE.
- Kardaş, S., Çelik, S., Arslan, A., & Levi, A. (2013). An efficient and private RFID authentication protocol supporting ownership transfer. In *International Workshop on Lightweight Cryptography for Security and Privacy*, 130-141. Berlin: Springer.
- Kardaş, S., Çelik, S., Bingöl, M. A., Kiraz, M. S., Demirci, H., & Levi, A. (2015). K-strong privacy for radio frequency identification authentication protocols based on physically unclonable functions. *Wireless Communications and Mobile Computing*, 15(18), 2150-2166.
- Keliher, L. T. (2003). *Linear cryptanalysis of substitution-permutation networks*. Phd Thesis, Queen's University, Canada.
- Khan, F. U., & Bhatia, S. (2012). A novel approach to genetic algorithm based cryptography. *International Journal of Research in Computer Science*, 2(3), 7, 2249-8265.
- Kim, H. S., Oh, J. H., Kim, J. B., Jeong, Y. O., & Choi, J. Y. (2008). Formal verification of cryptographic protocol for secure RFID system. In *Fourth International Conference on Networked Computing and Advanced Information Management*, 470-477. IEEE.
- Kim, J. H., Lee, K., Tanaka, S., & Park, S. H. (Eds.). (2012). Advanced Methods, Techniques, and Applications in Modeling and Simulation: Asia Simulation Conference 2011, *Proceedings* (4). Seoul: Springer Science & Business Media.
- Knežević, K. (2017). Combinatorial optimization in cryptography. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1324-1330. IEEE.
- Knudsen, L., Leander, G., Poschmann, A., & Robshaw, M. J. B. (2010). PRINTcipher: A block cipher for IC-printing. *Cryptographic Hardware and Embedded Systems CHES 2010, LNCS*, 6225/2010.

- Knuth, D. E. (1998). *Art of Programming, 2: Seminumerical Algorithms*. Boston: Addison-Wesley Longman Publishing Co., Inc.
- Kocamaz, A. F. (2008). *Yatarak tedavi gören şizofreni hastalarının negatif belirtilerinin RFID teknolojileri ile ölçülebilirliğinin değerlendirmesi*, Master's thesis, Trakya Üniversitesi, Edirne.
- Koç, C. (1995). Recurring-with-carry sequences. *Journal of Applied Probability*, 32(4), 966-971.
- Kotulski, Z. (2001). Generatory liczb losowych: algorytmy, testowanie, zastosowania. *Matematyka Stosowana: Matematyka Dla Społeczeństwa*, (2), 32-66.
- Koza, J. R. (1991). Evolving a computer program to generate random numbers using the genetic programming paradigm. In *Fourth International Conference on Genetic Algorithms*, 37-44. San Diego, CA, USA.
- Kösemen, C., Aydın, Ö., & Dalkılıç, G. (2018). The pseudorandom number generator generation method with genetic programming for lightweight devices. In *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, 17-21. IEEE
- Kösemen, C., Dalkılıç, G., & Aydın, Ö. (2018). Genetic programming-based pseudorandom number generator for wireless identification and sensing platform. *Turkish Journal of Electrical Engineering & Computer Sciences*, 26(5), 2500-2511.
- Krauss, J. D., & Marhefka, R. J. (2002). *Antennas for all applications*, (378-379), India: McGraw-Hill.
- Kumar, A., & Chatterjee, K. (2016, March). An efficient stream cipher using Genetic Algorithm. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2322-2326. IEEE.
- Kumar, A., Gopal, K., & Aggarwal, A. (2014). Simulation and analysis of authentication protocols for mobile Internet of Things (MIoT). In *2014*

- International Conference on Parallel, Distributed and Grid Computing*, 423-428. IEEE.
- Labay, V., & Anderson, A. M. (2006). Ethical considerations and proposed guidelines for the use of radio frequency identification: Especially concerning its use for promoting public safety and national security. *Science and Engineering Ethics*, 12(2), 265-272.
- Lahiri, S. (2005). *RFID sourcebook*. USA: IBM press.
- Lawnik, M. (2018). Generation of pseudo-random numbers with the use of inverse chaotic transformation. *Open Mathematics*, 16(1), 16-22.
- L'Ecuyer, P., & Simard, R. (2007). TestU01: AC library for empirical testing of random number generators. *ACM Transactions on Mathematical Software (TOMS)*, 33(4), 22.
- L'Ecuyer, P., & Simard, R. (2013). *TestU01: A Software Library in ANSI C for Empirical Testing of Random Number Generators--User's Guide*, Compact Version.
- Lee, C. C., Chen, S. D., Li, C. T., Cheng, C. L., & Lai, Y. M. (2019). Security enhancement on an RFID ownership transfer protocol based on cloud. *Future Generation Computer Systems*, 93, 266-277.
- Lehmer, D. H. (1951). Mathematical methods in large-scale computing units. *Proceedings of the 2nd Symposium on Large-Scale Digital Calculating Machinery*, 141-146.
- Lehtonen, M., Ostojic, D., Ilic, A., & Michahelles, F. (2009). Securing RFID systems by detecting tag cloning. In *International Conference on Pervasive Computing* (291-308). Berlin: Springer.
- Leng, X., Mayes, K., & Markantonakis, K. (2008). HB-MP+ protocol: An improvement on the HB-MP protocol. In *2008 IEEE International Conference on RFID*, 118-124. IEEE.

- Leonard, P., & Jackson, D. (2015). Efficient evolution of high entropy RNGs using single node genetic programming. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 1071-1078. Spain, Madrid: ACM.
- Leong, K. S., Ng, M. L., & Engels, D. W. (2005). *EPC Network architecture, Auto-ID labs white paper*. Retrieved April 4, 2019, from http://cocoa.ethz.ch/downloads/2014/06/None_AUTOIDLABS-WP-SWNET-012.pdf.
- Li, J., Zhou, Z., & Wang, P. (2017). Cryptanalysis of the LMAP protocol: A lowcost RFID authentication protocol. In *29th Chinese Control and Decision Conference (CCDC)*, 7292-7297. IEEE.
- Li, T., & Deng, R. (2007). Vulnerability analysis of EMAP-an efficient RFID mutual authentication protocol. In *The Second International Conference on Availability, Reliability and Security, 2007, ARES 2007*. 238–245. IEEE.
- Li, T., & Wang, G. (2007). Security analysis of two ultralightweight RFID authentication protocols. In H. Venter, M. Eloff, L. Labuschagne, J. Eloff, & R. von Solms (Eds.), *New Approaches for Security, Privacy And Trust In Complex Environments* (109–120). New York: Springer.
- Li, Y., & Ding, X. (2007). Protecting RFID communications in supply chains. In *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, 234-241. ACM.
- Liao, Y. P., & Hsiao, C. M. (2014). A secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol. *Ad Hoc Networks*, 18, 133-146.
- Liew, W. T., Tsai, K. Y., Luo, J. N., & Yang, M. H. (2017). Novel designated ownership transfer with grouping proof. In *2017 IEEE Conference on Dependable and Secure Computing*, 433-440. IEEE.
- Lin, Z., & Song, J. S. (2013). An improvement in HB-family lightweight authentication protocols for practical use of RFID system. *Journal of Advances in Computer Networks*, 1(1), 61-65.

- Lockton, V., & Rosenberg, R. S. (2005). RFID: The next serious threat to privacy. *Ethics and Information Technology*, 7(4), 221-231.
- Luo, H., Wen, G., Su, J., & Huang, Z. (2016). SLAP: Succinct and lightweight authentication protocol for low-cost RFID system. *Wireless Networks*, 1-10.
- Maarof, A., Labbi, Z., Senhadji, M., & Belkasmi, M. (2016). A novel mutual authentication scheme for low-cost RFID systems. In *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 240-245. IEEE.
- Mitra, M. (2008). Privacy for RFID systems to prevent tracking and cloning. *International Journal of Computer Science and Network Security*, 8(1), 1-5.
- Mitra, M. (2012). A low-cost lightweight random number generator implementation. *International Journal of Engineering Research & Technology (IJERT)*, 1(10), 1-9.
- Marsaglia, G. (1994). *Yet another rng*. Posted to the electronic billboard sci. stat. math, August, 1, Retrieved April 4, 2019, from [https://groups.google.com/forum/#!searchin/sci.stat.math/Yet\\$20another\\$20rng%7Csort:date/sci.stat.math/p7aLW3TsJys/QGb1kti6kN0J](https://groups.google.com/forum/#!searchin/sci.stat.math/Yet$20another$20rng%7Csort:date/sci.stat.math/p7aLW3TsJys/QGb1kti6kN0J).
- Marsaglia, G. (1995). *The DIEHARD battery of tests of randomness*. Technical Report, Florida State University, Retrieved April 4, 2019, from <http://www.csis.hku.hk/cisc/download/index2.htm>.
- Marsaglia, G. (1999). *Random Numbers for C: End, at last?*. Posting to sci. stat. math. URL <http://www.cs.yorku.ca/~oz/marsaglia-rng.html>.
- Marsaglia, G. (2003). Xorshift rngs. *Journal of Statistical Software*, 8(14), 1-6.
- Marsaglia, G., & Zaman, A. (1991). A new class of random number generators. *The Annals of Applied Probability*, 462-480.
- Marsaglia, G., & Zaman, A. (1993). *The KISS generator*. Technical Report, Department of Statistics, University of Florida, USA.

- Marton, K., & Suciu, A. (2015). On the interpretation of results from the NIST statistical test suite. *Science and Technology*, 18(1), 18-32.
- Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1), 3-30.
- Matthews, R. A. J. (1992). Maximally periodic reciprocals. *Bulletin Institute of Mathematics and Its Applications*, 28, 147-148.
- Miller, B. L., & Goldberg, D. E. (1995). Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex Systems*, 9(3), 193-212.
- Mishra, S., & Bali, S. (2013). Public key cryptography using genetic algorithm. *International Journal of Recent Technology and Engineering*, 2(2), 150-154.
- Molnar, D., & Wagner, D. (2004). Privacy and security in library RFID: Issues, practices, and architectures. In *Proceedings of the 11th ACM conference on Computer and Communications Security*, 210-219. ACM.
- Moosavi, S. R., Nigussie, E., Virtanen, S., & Isoaho, J. (2014). An elliptic curvebased mutual authentication scheme for RFID implant systems. *Procedia Computer Science*, 32, 198-206.
- MSP430FR5969 (2019). *Texas Instruments*. Retrieved April 4, 2019, from <http://www.ti.com/product/MSP430FR5969/description>.
- Mujahid, U., Najam-ul-Islam, M., & Shami, M. A. (2015). RCIA: a new ultralightweight rfid authentication protocol using recursive hash. *International Journal of Distributed Sensor Networks*, 11(1), Article ID 642180.
- Munilla, J., & Peinado, A. (2007). HB-MP: A further step in the HB-family of lightweight authentication protocols. *Computer Networks*, 51(9), 2262-2267.

- Naderiparizi, S., Parks, A. N., Kapetanovic, Z., Ransford, B., & Smith, J. R. (2015). WISPCam: A battery-free RFID camera. In *2015 IEEE International Conference on RFID (RFID)*, 166-173. IEEE.
- Neumann, P. G., & Weinstein, L. (2006). Risks of RFID. *Communications of the ACM*, 49(5), 136.
- Ning, H., Liu, H., Mao, J., & Zhang, Y. (2011). Scalable and distributed key array authentication protocol in radio frequency identification-based sensor systems. *IET Communications*, 5(12), 1755-1768.
- O'Neill, M. E. (2014). PCG: A family of simple fast space-efficient statistically good algorithms for random number generation. *ACM Transactions on Mathematical Software*.
- Ohkubo, M., Suzuki, K., & Kinoshita, S. (2004). Hash-chain based forward-secure privacy protection scheme for low-cost RFID. In *Proceedings of the SCIS*, 719-724.
- Özcanhan, M. H., Dalkılıç, G., & Utku, S. (2013). Analysis of two protocols using EPC Gen-2 tags for safe inpatient medication. In *2013 IEEE INISTA*, 1-6. IEEE.
- Özcanhan, M. H., Dalkılıç, G., & Utku, S. (2014). Cryptographically supported NFC tags in medication for better inpatient safety. *Journal of Medical Systems*, 38(8), 1-15.
- Özcanhan, M. H., Dalkılıç, G., & Gürle, M. C. (2012). An Ultra-Light PRNG for RFID Tags. In *Computer and Information Sciences III: 27th International Symposium on Computer and Information Sciences*, 231-231. Springer Science & Business Media.
- Pala, Z. (2007). *RFID Teknolojisi ile otomasyon bir uygulama olarak: otopark takibi*. Yüksek Lisans Tezi, Yüzüncü Yıl Üniversitesi Fen Bilimleri Enstitüsü, Van.
- Pang, L., Li, H., He, L., Alramadhan, A., & Wang, Y. (2014). Secure and efficient lightweight RFID authentication protocol based on fast tag indexing. *International Journal of Communication Systems*, 27(11), 3244-3254.

- Panneton, F., L'ecuyer, P., & Matsumoto, M. (2006). Improved long-period generators based on linear recurrences modulo 2. *ACM Transactions on Mathematical Software (TOMS)*, 32(1), 1-16.
- Park, J., Na, J., & Kim, M. (2007). A practical approach for enhancing security of EPCglobal RFID Gen2 tag. In *Future Generation Communication and Networking (FGCN 2007)*, 1, 436-441. IEEE.
- Park, S. K., & Miller, K. W. (1988). Random number generators: good ones are hard to find. *Communications of the ACM*, 31(10), 1192-1202.
- Patel, R. G., Patel, N. P., & Patel, D. R. (2009). Next Generation of Auto-ID: Applying RFID Technology. In *IEEE International Advance Computing Conference-09 (IACC'09)*, 2705-2710.
- Peris-Lopez, P., Hernandez-Castro, J. C., Estevez-Tapiador, J. M., & Ribagorda, A. (2006a). M2AP: A minimalist mutual authentication protocol for low-cost RFID tags. In *Ubiquitous Intelligence and Computing*, 912-923. Berlin: Springer.
- Peris-Lopez, P., Hernandez-Castro, J. C., Estevez-Tapiador, J. M., & Ribagorda, A. (2006b). LMAP: A real lightweight mutual authentication protocol for low-cost RFID tags. In *Workshop on RFID Security*, 12-14.
- Peris-Lopez, P., Hernandez-Castro, J. C., Estevez-Tapiador, J. M., & Ribagorda, A. (2006c). EMAP: An efficient mutual-authentication protocol for low-cost RFID tags. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, 352-361. Berlin: Springer.
- Peris-Lopez, P., Hernandez-Castro, J. C., Estevez-Tapiador, J. M., & Ribagorda, A. (2009a). Cryptanalysis of a novel authentication protocol conforming to EPCC1G2 standard. *Computer Standards & Interfaces*, 31(2), 372-380.
- Peris-Lopez, P., Hernandez-Castro, J. C., Estevez-Tapiador, J. M., & Ribagorda, A. (2009b). LAMED—a PRNG for EPC class-1 generation-2 RFID specification. *Computer Standards & Interfaces*, 31(1), 88-97.

- Peris-Lopez, P., Hernandez-Castro, J. C., Tapiador, J. M., & Ribagorda, A. (2009c). Advances in ultralightweight cryptography for low-cost RFID tags: Gossamer protocol. In *Information Security Applications*, 56-68. Berlin: Springer.
- Peris-Lopez, P., Safkhani, M., Bagheri, N., & Naderi, M. (2013). RFID in eHealth: How to combat medication errors and strengthen patient safety. *Journal of Medical and Biological Engineering*, 33, 363-372.
- Picek, S., Sisejkovic, D., Rozic, V., Yang, B., Jakobovic, D., & Mentens, N. (2016). Evolving cryptographic pseudorandom number generators. In *International Conference on Parallel Problem Solving from Nature*, 613-622. Springer, Cham.
- Potts, K., Shore, D., & Wood, D. (2007). *U.S. Patent Application No. 11/849,689*.
- Qian, Z., Chen, C., You, I., & Lu, S. (2012). ACSP: A novel security protocol against counting attack for UHF RFID systems. *Computers & Mathematics with Applications*, 63(2), 492-500.
- Qingling, C., Yiju, Z., & Yonghua, W. (2008). A minimalist mutual authentication protocol for RFID system & BAN logic analysis. In *2008 ISECS International Colloquium on Computing, Communication, Control, and Management CCCM'08*, 2, 449-453. IEEE.
- Rahman, M. S., Soshi, M., & Miyaji, A. (2009). A secure RFID authentication protocol with low communication cost. In *International Conference on Complex, Intelligent and Software Intensive Systems, 2009. CISIS'09*, 559-564. IEEE.
- Ranasinghe, D. C., Leong, K. S., Ng, M. L., & Cole, P. H. (2006). Small UHF RFID label antenna design and limitations. In *IEEE International Workshop on Antenna Technology: Small Antennas and Novel Metamaterials*.
- Ranasinghe, D., Engels, D., & Cole, P. (2004). Security and privacy: modest proposals for low-cost RFID systems. In *Auto-ID Labs Research Workshop*, Zurich, Switzerland.

- Rathore, R., & Hussain, M. (2015). Simple, secure, efficient, lightweight and token based protocol for mutual authentication in wireless sensor networks. In *Emerging Research in Computing, Information, Communication and Applications*, 451-462. New Delhi: Springer.
- Ren, X., Xu, X., & Li, Y. (2013). An One-way Hash Function Based Lightweight Mutual Authentication RFID Protocol. *Journal of Computers*, 8(9), 2405-2412.
- Repanovici, A., & Cristea, L. (2011). RFID-application in info-documentary systems. In *Designing and Deploying RFID Applications*. InTECH.
- RFID-Internet Archive (2019). Retrieved April 3, 2019, from <https://archive.org/download/rfid-/rfid-.JPG>.
- Rose, G. G. (2018). KISS: A bit too simple. *Cryptography and Communications*, 10(1), 123-137.
- Rostampour, S., Bagheri, N., Hosseinzadeh, M., & Khademzadeh, A. (2016). An authenticated encryption based grouping proof protocol for RFID systems. *Security and Communication Networks*, 9(18), 5581-5590.
- Rostampour, S., Namin, M. E., & Hosseinzadeh, M. (2014). A novel mutual RFID authentication protocol with low complexity and high security. *International Journal of Modern Education and Computer Science*, 6(1), 17.
- Rotenberg, A. (1960). A new pseudo-random number generator. *Journal of the ACM (JACM)*, 7(1), 75-77.
- Salmon, J. K., Moraes, M. A., Dror, R. O., & Shaw, D. E. (2011). Parallel random numbers: as easy as 1, 2, 3. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 16. ACM.
- Sample, A. P., Yeager, D. J., Powledge, P. S., & Smith, J. R. (2007). Design of a passively-powered, programmable sensing platform for UHF RFID systems. In *2007 IEEE International Conference on RFID*, 149-156. IEEE.

- Sample, A. P., Yeager, D. J., Powledge, P. S., Mamishev, A. V., & Smith, J. R. (2008). Design of an RFID-based battery-free programmable sensing platform. *IEEE Transactions on Instrumentation and Measurement*, 57(11), 2608-2615.
- Sanguinetti, B., Martin, A., Zbinden, H., & Gisin, N. (2014). Quantum random number generation on a mobile phone. *Physical Review X*, 4(3), 031056.
- Sarma, S. E., Weis, S. A., & Engels, D. W. (2003). RFID systems and security and privacy implications. In *Cryptographic Hardware and Embedded Systems CHES 2002*, 454-469. Berlin: Springer.
- Savvidy, G. K., & Ter-Arutyunyan-Savvidy, N. G. (1991). On the Monte Carlo simulation of physical systems. *Journal of Computational Physics*, 97(2), 566-572.
- Saxena, N., & Voris, J. (2009). Accelerometer based random number generation on RFID tags. *WISP Summit*. Retrieved April 4, 2019, from <https://pdfs.semanticscholar.org/7b3e/149fe8cd1b38d7ad74624f140daea7850159.pdf>
- Schneier, B., & Sutherland, P. (1995). *Applied cryptography: protocols, algorithms, and source code in C*. New York: John Wiley & Sons.
- Shao-hui, W., Sujuan, L., & Danwei, C. (2012). Efficient Passive Full-disclosure Attack on RFID Light-weight Authentication Protocols LMAP++ and SUAP. *Indonesian Journal of Electrical Engineering and Computer Science*, 10(6), 1458-1464.
- Shen, J., Tan, H., Moh, S., Chung, I., & Wang, J. (2016). An efficient RFID authentication protocol providing strong privacy and security. *Journal of Internet Technology*, 17(3), 443-455.
- Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., & Shirai, T. (2011). Piccolo: an ultra-lightweight blockcipher. In *International Workshop on Cryptographic Hardware and Embedded Systems*, 342-357. Berlin: Springer.

- Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3), 379-423.
- Shoarinejad, K., & Soltan, M. (2009). *U.S. Patent No. 7,492,258*. Washington, DC: U.S. Patent and Trademark Office.
- Singh, K. P., & Kumar, D. (2012). Performance Evaluation of Low Power Encrypted MIPS Crypto Processor based on Cryptography Algorithms. *Performance Evaluation*, 2(3), 1625-1634.
- Singh, R., Singh, E., & Nalwa, H. S. (2017). Inkjet printed nanomaterial based flexible radio frequency identification (RFID) tag sensors for the internet of nano things. *RSC Advances*, 7(77), 48597-48630.
- So, S. C. K., & Liu, J. J. (2006). Securing RFID applications: Issues, methods, and controls. *Information Systems Security*, 15(4), 43-50.
- Song, B., & Mitchell, C. J. (2008). RFID authentication protocol for low-cost tags. In *Proceedings of the First ACM Conference on Wireless Network Security*, 140-147. ACM.
- Srivastava, K., Awasthi, A. K., Kaul, S. D., & Mittal, R. C. (2015). A hash based mutual RFID tag authentication protocol in telecare medicine information system. *Journal of Medical Systems*, 39(1), 153.
- Standard, D. E. (1977). FIPS publication 46.
- Steele, G. L. Jr, Lea, D., & Flood, C. H. (2014). Fast splittable pseudorandom number generators. In *ACM SIGPLAN Notices*, 49(10), 453-472.
- Stępień, R., & Walczak, J. (2012). Application of the DLFSR generators in spread spectrum communication. In *Proceedings of the 19th International Conference Mixed Design of Integrated Circuits and Systems-MIXDES 2012*, 555-558. IEEE.
- Stinson, D. R. (2005). *Cryptography: theory and practice*. Florida: Chapman and Hall/CRC.

- Stockman, H. (1948). Communication by means of reflected power. *Proceedings of the IRE*, 36(10), 1196-1204.
- Sundaresan, S., Doss, R., PIRAMUTHU, S., & Zhou, W. (2014). A robust grouping proof protocol for RFID EPC C1G2 tags. *IEEE Transactions on Information Forensics and Security*, 9(6), 961-975.
- Swedberg, C. (2013). Motorola Sled Reader Adds Low-Cost RFID. *RFID Journal*, 2.
- Sýs, M., & Říha, Z. (2014). Faster randomness testing with the NIST statistical test suite. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, 272-284. Springer, Cham.
- Şahin, Ö. E. (2010). RFID Teknolojisinde İnsan Bilgisayar Etkileşimi. *Ajit-e Bilişim Teknolojileri Akademik Dergisi*, 1(1), 1-7.
- Tan, C. C., Sheng, B., & Li, Q. (2008). Secure and serverless RFID authentication and search protocols. *IEEE Transactions on Wireless Communications*, 7(4), 1400-1407.
- Tausworthe, R. C. (1965). Random Numbers Generated by Linear Recurrence Modulo Two. *Mathematics of Computation*, 19, 201–209.
- Tewari, A., & Gupta, B. B. (2017). Cryptanalysis of a novel ultra-lightweight mutual authentication protocol for IoT devices using RFID tags. *The Journal of Supercomputing*, 73(3), 1085-1102.
- Thakur, J., & Kumar, N. (2011). DES, AES and Blowfish: Symmetric key cryptography algorithms simulation based performance analysis. *International Journal of Emerging Technology and Advanced Engineering*, 1(2), 6-12.
- Thomson, W. E. (1958). A modified congruence method of generating pseudo-random numbers. *The Computer Journal*, 1(2), 83-83.
- Tian, Y., Chen, G., & Li, J. (2012). A new ultralightweight RFID authentication protocol with permutation. *IEEE Communications Letters*, 16(5), 702-705.

- Türkiye Odalar ve Borsalar Birliği [TOBB] (2019). *GSİ Türkiye hakkında*. Retrieved April 4, 2019, from www.gs1tr.org/menu_goster.php?Id=3&MenuId=2.
- Tomassini, M., Sipper, M., Zolla, M., & Perrenoud, M. (1999). Generating high-quality random numbers in parallel by cellular automata. *Future Generation Computer Systems*, 16(2-3), 291-305.
- Tounsi, W., Cuppens-Boulahia, N., Garcia-Alfaro, J., Chevalier, Y., & Cuppens, F. (2014). KEDGEN2: A key establishment and derivation protocol for EPC Gen2 RFID systems. *Journal of Network and Computer Applications*, 39, 152-166.
- Tsudik, G. (2006). YA-TRAP: Yet another trivial RFID authentication protocol. In *Fourth annual IEEE international conference on pervasive computing and communications workshops, PerCom Workshops 2006*. IEEE.
- Tsudik, G. (2007). A family of dunces: Trivial RFID identification and authentication protocols. In *International Workshop on Privacy Enhancing Technologies*, 45-61. Berlin: Springer.
- Turan, M. S., Barker, E., Kelsey, J., McKay, K. A., Baish, M. L., & Boyle, M. (2016). *Recommendation for the entropy sources used for random bit generation. NIST Special Publication 800-90B second draft*. Retrieved April 4, 2019, from https://csrc.nist.gov/csrc/media/publications/sp/800-90b/draft/documents/sp800-90b_second_draft.pdf.
- Tuyls, P., & Batina, L. (2006). RFID-tags for Anti-Counterfeiting. In *Topics in Cryptology-CT-RSA 2006*, 115-131. Berlin: Springer.
- Uçar, M. H., Sondaş, A., & Erdemli, Y. E. (2010). Çevrimiçi RF Veri Toplama Sistemi için UHF-RFID Okuyucu Anten Tasarımı. V. *URSI-Türkiye'2010 Bilimsel Kongresi*.
- Üstündağ, A., & Tanyaş, M. (2009). Radyo Frekanslı Tanıma (RFID) teknolojisinin tedarik zinciri üzerindeki etkileri. *İTÜdergisi/d*, 8(4), 83-94.

- Vaddi, R., Agarwal, R. P., Dasgupta, S., & Kim, T. T. (2011). Design and analysis of double-gate mosfets for ultra-low power radio frequency identification (rfid): Device and circuit co-design. *Journal of Low Power Electronics and Applications*, 1(2), 277-302.
- Van Le, T., Burmester, M., & De Medeiros, B. (2007). Universally composable and forward secure RFID authentication and authenticated key exchange. In *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, 242-252. ACM.
- Vigna, S. (2016). An experimental exploration of Marsaglia's xorshift generators, scrambled. *ACM Transactions on Mathematical Software (TOMS)*, 42(4), 30.
- Vigna, S. (2017). Further scramblings of Marsaglia's xorshift generators. *Journal of Computational and Applied Mathematics*, 315, 175-181.
- Von Neumann, J. (1951). 13. various techniques used in connection with random digits. *Applied Math Series*, 12(36-38), 5.
- Voris, J., Saxena, N., & Halevi, T. (2011). Accelerometers and randomness: Perfect together. In *Proceedings of the fourth ACM conference on Wireless network security*, 115-126. ACM.
- Walker, J. (2008). *ENT: a pseudorandom number sequence test program*. Software and documentation Retrieved April 4, 2019, from <http://www.fourmilab.ch/random/>.
- Weis, S. A. (2007). RFID (radio frequency identification): Principles and applications. *System*, 2(3), 1-23.
- Weis, S. A., Sarma, S. E., Rivest, R. L., & Engels, D. W. (2004). Security and privacy aspects of low-cost radio frequency identification systems. In D. Hutter, G. Müller, W. Stephan, & M. Ullmann (Eds.), *Security in pervasive computing*, (201–212). Berlin, Heidelberg: Springer.

- Wesołowski, K. (2006). Podstawy cyfrowych systemów telekomunikacyjnych. *Wydawnictwa Komunikacji i Łączności*. 348-366, Warszawa.
- Wetzel, C. (1999). *Can you behave randomly*. Retrieved April 4, 2019 from <http://faculty.rhodes.edu/wetzel/random/intro.html>
- Wichmann, B. A., & Hill, I. D. (1982). Algorithm AS 183: An efficient and portable pseudo-random number generator. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(2), 188-190.
- Wisman, R. (2015). *MSP430 timers and PWM*. Retrieved April 4, 2019, from <http://pages.iu.edu/~rwisman/c335/html/msp430timer.htm>
- Wohlgend, J. (2016). *Elliptic curve cryptography: Pre and post quantum*. MIT, Technical Report. Retrieved April 4, 2019, from https://math.mit.edu/~apost/courses/18.204-2016/18.204_Jeremy_Wohlgend_final_paper.pdf
- Wolfram, S. (1983). Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3), 601-644.
- Wu, S., Chen, K., & Zhu, Y. (2012). A secure lightweight RFID binding proof protocol for medication errors and patient safety. *Journal of Medical Systems*, 36(5), 2743-2749.
- Wu, W., & Zhang, L. (2011). LBlock: a lightweight block cipher. In *International Conference on Applied Cryptography and Network Security*, 327-344. Berlin: Springer.
- Xiao, F., Zhou, Y. J., Zhou, J. X., & Niu, X. X. (2013). Provable secure mutual authentication protocol for RFID in the standard model. *Journal on Communications*, 34(4), 82-87.
- Xydis, T. G., & Blake-Wilson, S. (2002). Security Comparison: Bluetooth TM Communications vs. 802.11. *Bluetooth Security Experts Group*, 29, 2004.

- Yeager, D. J., Sample, A. P., & Smith, J. R. (2008). Wisp: A passively powered uhf rfid tag with sensing and computation. *RFID handbook: Applications, technology, security, and privacy*, 261-278.
- Yıldırım, K. S., Aantjes, H., Majid, A. Y., & Pawełczak, P. (2016). *On the synchronization of intermittently powered wireless embedded systems*. arXiv preprint arXiv:1606.01719.
- Yılmaz, H. (2013). *RFID Akıllı Etiketler, Uygulama Alanlarının İncelenmesi ve Öneriler*. Bilgi Teknolojileri ve İletişim Kurumu, Samsun.
- Yamamoto, A., Suzuki, S., Hada, H., Mitsugi, J., Teraoka, F., & Nakamura, O. (2008). A tamper detection method for RFID tag data. In *2008 IEEE international conference on RFID*, 51-57. IEEE.
- Yuan, B., & Liu, J. (2015). A universally composable secure grouping-proof protocol for RFID tags. *Concurrency and computation: practice and experience*, 28(6), 1872-1883.
- Yüksel, M. E., & Zaim, A. H. (2009). RFID'nin Kablosuz İletişim Teknolojileri ile Etkileşimi. *Akademik Bilişim*, 11-13.
- Zavvari, A., & Patel, A. (2012). Critical evaluation of RFID security protocols. *International Journal of Information Security and Privacy (IJISP)*, 6(3), 56-74.
- Zeng, K., Yang, C. H., Wei, D. Y., & Rao, T. R. N. (1991). Pseudorandom bit generators in stream-cipher cryptography. *Computer*, 24(2), 8-17.
- Zhang, W., Liu, S., Wang, S., Yi, B., & Wu, L. (2017). An efficient lightweight RFID authentication protocol with strong trajectory privacy protection. *Wireless Personal Communications*, 96(1), 1215-1228.
- Zhang, W., Qin, S., Wang, S., Wu, L., & Yi, B. (2018). A New Scalable Lightweight Grouping Proof Protocol for RFID systems. *Wireless Personal Communications*, 103(1), 133-143.

- Zhang, W., Wu, L., Liu, S., Huang, T., Guo, Y., & Hsu, C. (2016). A trajectory privacy model for radio-frequency identification system. *Wireless personal communications*, 90(3), 1121-1134.
- Zhang, Z., & Qi, Q. (2014). An efficient RFID authentication protocol to enhance patient medication safety using elliptic curve cryptography. *Journal of medical systems*, 38(5), 47.
- Zhang, Z., Zhou, S., & Luo, Z. (2008). Design and analysis for RFID authentication protocol. In *IEEE International Conference on e-Business Engineering*, 574-577. IEEE.
- Zhao, Z. (2014). A secure RFID authentication protocol for healthcare environments using elliptic curve cryptosystem. *Journal of medical systems*, 38(5), 46.
- Zhou, P., Jiang, T., Wang, C., Zhang, Q. (2014). ZigBee Profile (Chapter 8). In *ZigBee® network protocols and applications*, (Eds.) Wang, C., Jiang, T., Zhang, Q. Florida: Crc Press.
- Zhou, S., Zhang, Z., Luo, Z., & Wong, E. C. (2010). A lightweight anti desynchronization RFID authentication protocol. *Information Systems Frontiers*, 12(5), 521-528.
- Zhou, Z., Liu, P., Liu, Q., & Wang, G. (2017). An ECC-Based Off-line Anonymous Grouping-Proof Protocol. In *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*, 186-200. Springer, Cham.
- Zhou, Z., Liu, P., Liu, Q., & Wang, G. (2018). An Anonymous Offline RFID Grouping-Proof Protocol. *Future Internet*, 10(1), 2.
- Zhu, S., Yang, B., & Zhang, M. (2007). Research on RFID protocols and security. *Information Security and Confidentiality of Communications*, 168 – 170.
- Zhuang, X., Zhu, Y., & Chang, C. C. (2014). A new ultralightweight RFID protocol for low-cost tags: R2AP. *Wireless Personal Communications*, 79(3), 1787-1802.

APPENDICES

APPENDIX 1: C code to sample voltage via ADC on WISP

```
void setup_adc()
{
    // setup adc for external voltage
    ADC12CTL0 &= ~ADC12ENC; // make sure this is off otherwise settings are locked.
    ADC12CTL1 = ADC12SHT1_6 + ADC12DIV_2;
    ADC12CTL0 = CNTL_1 + ADC12ON + ADC12IE0;
}

void readVolt() {
    // setup clock for reading bits from reader. This setup is for 6 MHz
    BCSCTL1 = XT2OFF + ADC12VRSEL3 + ADC12VRSEL2;
    // select internal resistor (still has effect when DCOR=1)
    DCOCTL = DCOFSEL1+DCOFSEL0; // set DCO step. this is 6 MHz set up
    // setup ADC for Voltage Sampling
    setup_adc();
    while (1){
        for (j = 0; j < 16; j++){ // 1st for
            count++;
            max = 0;
            min = 1024;
            diffID = 8000;
            diff = 8000;
            for(i = 0; i < 3; i++){ // 2nd for
                // enter Sampling Mode
                P3DIR |= BIT3;
                P3OUT |= BIT3;
                sample_adc(); // check current voltage
                P3DIR |= BIT3;
                P3OUT &= ~BIT3;
                if (Vbatt < min){
                    min = Vbatt;
                    minID = i;
                }
                else if (Vbatt > max){
                    max = Vbatt;
                    maxID = i;
                }
                for (k = 0; k < 1000; k++); // waste time
            } // End of 2nd for loop
            history[j][0] = count;
            history[j][1] = max;
            history[j][2] = min;
            history[j][3] = max-min;
            history[j][4] = maxID;
            history[j][5] = minID;
            max = 0;
            min = 1024;
            diffID = 8000;
            diff = 8000;
        } // End of 1st for loop
        count = 0;
    } // End of while loop
}

inline void sample_adc(){
    // start conversion
    ADC12CTL0 |= ADC12ENC + ADC12SC; // Sampling and conversion start
    while (ADC12CTL1 & ADC12BUSY); // wait while ADC finished work
    Vbatt = ADC12MEM;
}

#pragma vector=ADC12_VECTOR
__interrupt void ADC10_ISR (void){
    ADC12CTL0 &= ~ADC12ENC; // make sure this is off otherwise settings are locked.
    ADC12CTL0 &= ~(ADC12IFG0 | ADC12IE0);
    LPM4_EXIT;
    return;
}
```

APPENDIX 2: C code to generate random number sampling from temperature sensor on WISP

```

/* @fcn adc12_genRN16 (void)
 * @brief generate a random 16-bit value by sampling b0 of the temperature sensor.
 * Use the 10-bit ADC in repeat-single-sample mode to read the ADC 16 times, keeping
 * each samples' b0 and assembling a full word from them at the end. Use the WISPs recently
 * acquired ADC_clk setting, just cause its available.
 * @return (int16_t) a random 16 bit value.
 */
uint16_t RAND_adcRand16 (void) {
    uint8_t      i;
    uint16_t      returnVal;

    //-----Turn on reference module's VREF and Temp Sense-----//
    REFCTL0 = REFVSEL_1 + REFON; /* turn module on @ 2V with temp sense active */
    // @todo Replace with low power delay
    __delay_cycles(75*16); /* do an ~75us delay for ref to stabilize*/
    //-----Initialize the ADC to known state-----//
    ADC12CTL0 = 0;
    ADC12CTL1 = 0;
    ADC12CTL2 = 0;
    ADC12MCTL0 = 0;
    ADC12IER0 = 0;
    //-----Setup the ADC for Repeat Single-Sample on Temp Sensor ---//

    //-----Clocking Stuff-----//
    // ADC12CTL1 |= ADC12SSEL_2; /* ADC Clk src is SMCLK */
    ADC12CTL1 |= ADC12DIV_0; /* ADC Clk is /1 off source */

    //-----S&H Config-----//
    //ADC12CTL0 |= ADC12MSC; /* setup for repeated samples off first SHI pulse */
    //ADC12CTL1 |= ADC12SHS0; /* S&H source comes directly from ADC12SC */
    ADC12CTL1 |= ADC12SHP; /* enable using the S&H Sampling Timer */
    ADC12CTL0 |= ADC12SHT0_2; /* 16 S&H Cycle cause we need min 13 for the 12-bit res sample */

    //-----Msrmt Source-----//
    ADC12CTL2 |= ADC12RES_2; /* 10 bit sample res to max noise res in msr */
    ADC12MCTL0 |= ADC12VRSEL_1; /* reference is VREF and AVSS*/
    ADC12MCTL0 |= ADC12INCH_10; /* setup to A10 */

    //-----Conversion Sequence-----//
    ADC12CTL1 |= ADC12CONSEQ_0; /* mode is single-sample on single-channel */

    //-----Turn On the ADC-----//
    ADC12CTL0 |= ADC12ON; /* turn that sucker on! */
    while (ADC12CTL1 & ADC12BUSY; /* Wait if ADC12 core is active (i.e. wait until its
    inactive, safety) */

    //-----Acquire 16 measurements!---- //
    ADC12CTL0 |= ADC12ENC+ADC12SC; /* start that conversion! */
    while (ADC12CTL1 & ADC12BUSY); /* Wait for it to complete all 16 samples into MEM0-
15 */
    returnVal = 0;
    for(i=0;i<16;i++) {
        returnVal = (returnVal<<1) | (ADC12MEM0 & BIT0);
        ADC12CTL0 |= ADC12SC;
        while (ADC12CTL1 & ADC12BUSY); /* Wait for it to complete all 16 sampling */
    }
    //--Turn off the ADC-----//
    ADC12CTL0 = 0;
    ADC12CTL1 = 0;
    ADC12CTL2 = 0;
    ADC12MCTL0 = 0;
    ADC12IER0 = 0;
    //---Turn off the REF-----//
    REFCTL0 = 0;
    return returnVal;
}

```

APPENDIX 3: C code to generate true random number by using difference between clock pulses

```
#define RTACCTL TACCTL0
#define RTACCR TACCR0
/*
 * True random number generator using hardware clocks (VLO and DCO)
 */
int slaa338(void) {
    int i, j;
    unsigned _ctl, r = 0;
    /* Save old Timer A control state. */
    _ctl = TACTL;
    RTACCTL = CAP | CM_1 | CCIS_1; /* Capture mode */
    TACTL = TASSEL_2 | MC_2; /* SMCLK (submain), continuous mode */
    /* Fetch all 16 bits individually from hardware clock. */
    for (i=0; i < 16; i++) {
        unsigned char cnt = 0;
        for (j=0; j < 5; j++) {
            /* Wait for interrupt */
            while (!(CCIFG & RTACCTL));
            RTACCTL &= ~CCIFG;
            if (RTACCR & 0x1)
                cnt++;
        }
        /* Shift result and set MSB */
        r >>= 0x1;
        if (cnt > 2)
            r |= 0x8000;
        }
    /* Restore state */
    TACTL = _ctl;
    return r;
}
```

APPENDIX 4: C code to generate true random number by using accelerometer (1st method)

```
void accel_PowerUp (void){
    // Accelerometer power up sequence
    BITCLR(POUT_ACCEL_EN , PIN_ACCEL_EN);
    __delay_cycles(100);
    BITSET(POUT_ACCEL_EN , PIN_ACCEL_EN);

    accelOut.x = 1;
    accelOut.y = 1;
    accelOut.z = 1;
    BITSET(P2SEL1 , PIN_ACCEL_SCLK | PIN_ACCEL_MISO | PIN_ACCEL_MOSI);
    BITCLR(P2SEL0 , PIN_ACCEL_SCLK | PIN_ACCEL_MISO | PIN_ACCEL_MOSI);
    __delay_cycles(5);
    SPI_initialize();
    __delay_cycles(5);
    ACCEL_range();
    __delay_cycles(5);
    ACCEL_initialize();
    __delay_cycles(5);
    ACCEL_readID(&accelOut);
    while(accelOut.x != 0xAD){
        __delay_cycles(5);
        ACCEL_readID(&accelOut);
    }
    __delay_cycles(5);
    ACCEL_readStat(&accelOut);

    while((accelOut.x & 192) != 0x40){
        __delay_cycles(5);
        ACCEL_readStat(&accelOut);
    }
    __delay_cycles(5);
    ACCEL_singleSample(&accelOut);
}

uint16_t random_accel(void)
{
    uint8_t temp1;
    uint8_t temp2;
    uint8_t temp3;
    uint8_t temp4;

    temp1 = 0;
    temp2 = 0;
    temp3 = 0;
    temp4 = 0;

    //Consists out of the x- and y- accelerometer data (Low byte, High byte)
    unsigned short rng_seed[] = {0x0000, 0x0000};

    while(1){
        //First of all we have to initialize the random number generator with a seed.
        //Therefore acceleration sensor is used. Furthermore we sample 5 times, to check if
        int i = 0;
        while(i < 5){
            i++;
            ACCEL_singleSample(&accelOut);
            rng_seed[0] = accelOut.x;
            rng_seed[1] = accelOut.y;
            if(i == 0){
                //Save the initial values
                temp1 = rng_seed[0];
                temp2 = rng_seed[1];
                temp3 = rng_seed[0];
                temp4 = rng_seed[1];
            }
            temp3 += rng_seed[0];
            temp4 += rng_seed[1];
            temp3 /= 2;
        }
    }
}
```

```

    temp4 /= 2;
}
//Now we have to check if the tag was moved during the initialization routine.
temp1 = (signed) ((signed) temp1 - (signed) temp3);
temp2 = (signed) temp1 - (signed) temp4;
if((signed) temp1 < 0 && (signed) temp1 > -25)
    continue;
else if((signed) temp1 >= 0 && (signed) temp1 < 25)
    continue;
else if((signed) temp2 < 0 && (signed) temp2 > -25)
    continue;
else if ((signed) temp2 >= 0 && (signed) temp2 < 25)
    continue;
else
    break;
}
return ((uint16_t)temp1 << 8) | temp2;
}

```



APPENDIX 5: C code to generate true random number by using accelerometer (2nd method)

```

uint8_t i = 0;
uint16_t sayi = 0;
uint16_t counter = 0;
uint32_t randsayi = 0;

WISP_init();
BITSET(PDIR_ACCEL_EN , PIN_ACCEL_EN);
// Accelerometer power up sequence
BITCLR(POUT_ACCEL_EN , PIN_ACCEL_EN);
CSCTL0_H = 0xA5;
CSCTL1 = DCOFSEL_0; //1MHz
CSCTL2 = SELA_VLOCLK + SELS_3 + SELM_3;
CSCTL3 = DIVA_0 + DIVS_0 + DIVM_0;
BITCLR(CSCTL6 , (MODCLKREQEN|SMCLKREQEN|MCLKREQEN));
BITSET(CSCTL6 , ACLKREQEN);
__delay_cycles(100);
BITSET(POUT_ACCEL_EN , PIN_ACCEL_EN);
// Register callback functions with WISP base routines
WISP_registerCallback_ACK(&my_ackCallback);
WISP_registerCallback_READ(&my_readCallback);
WISP_registerCallback_WRITE(&my_writeCallback);
WISP_registerCallback_BLOCKWRITE(&my_blockWriteCallback);

// Get access to EPC, READ, and WRITE data buffers
WISP_getDataBuffers(&wispData);

// Set up operating parameters for WISP comm routines
// Set mode: Tag responds to R/W and obeys the sel cmd
WISP_setMode(MODE_READ | MODE_WRITE | MODE_USES_SEL);

// Set abort conditions: Exits WISP_doRFID() when the following events happen:
WISP_setAbortConditions(CMD_ID_READ | CMD_ID_WRITE | CMD_ID_ACK);

accelOut.x = 1;
accelOut.y = 1;
accelOut.z = 1;
BITSET(P2SEL1 , PIN_ACCEL_SCLK | PIN_ACCEL_MISO | PIN_ACCEL_MOSI);
BITCLR(P2SEL0 , PIN_ACCEL_SCLK | PIN_ACCEL_MISO | PIN_ACCEL_MOSI);
__delay_cycles(5);
SPI_initialize();
__delay_cycles(5);
ACCEL_range();
__delay_cycles(5);
ACCEL_initialize();
__delay_cycles(5);
__delay_cycles(5);

// Set up EPC, copy in sensor data
wispData.epcBuf[0] = 0x0B; // Tag type: Accelerometer
wispData.epcBuf[7] = 0x00; // Unused data field
wispData.epcBuf[8] = 0x00; // Unused data field
wispData.epcBuf[9] = 0x51; // Tag hardware revision (5.1)
wispData.epcBuf[10] = *((uint8_t*)INFO_WISP_TAGID+1) // WISP ID MSB: Pull from INFO seg
wispData.epcBuf[11] = *((uint8_t*)INFO_WISP_TAGID); // WISP ID LSB: Pull from INFO seg

ACCEL_singleSample(&accelOut);

BITSET(TA2CCTL0 , CCIE);
TA2CCR0 = 1000;
BITSET(TA2CTL , (TASSEL_1 | MC_1 | TACLRL));
__bis_SR_register(LPM4_bits | GIE); // Enter LPM4 w/ interrupt
__delay_cycles(1);
BITCLR(TA2CCTL0 , CCIE);
BITCLR(TA2CCR0 , 1000);
BITCLR(TA2CTL , (TASSEL_1 | MC_1 | TACLRL));
ACCEL_readStat(&accelOut);
while((((uint8_t)accelOut.x) & 193) != 0x41){
    ACCEL_readStat(&accelOut);
}

```

```

    __delay_cycles(10);
}
__delay_cycles(10);
ACCEL_singleSample(&accelOut);
wispData.epcBuf[2] = (accelOut.y+128); // Y value LSB
wispData.epcBuf[4] = (accelOut.x+128); // X value LSB
wispData.epcBuf[6] = (accelOut.z+128); // Z value LSB
while (FOREVER) {
    while(1){
        CSCTL0_H = 0xA5;
        CSCTL1 = DCOFSEL_0; //1MHz
        CSCTL2 = SELA__VLOCLK + SELS_3 + SELM_3;
        CSCTL3 = DIVA_0 + DIVS_0 + DIVM_0;
        BITCLR(CSCTL6 , (MODCLKREQEN|SMCLKREQEN|MCLKREQEN));
        BITSET(CSCTL6 , ACLKREQEN);

        ACCEL_readStat(&accelOut);
        if(((uint8_t)accelOut.x) & 193) == 0x41){
            __delay_cycles(20);
            ACCEL_singleSample(&accelOut);
            sayi = sayi | (accelOut.x & 1);
            sayi = sayi << 1;
            i = i + 1;
            if(i==16)
                break;
            sayi = sayi | (accelOut.y & 1);
            sayi = sayi << 1;
            i = i + 1;
            if(i==16)
                break;
            sayi = sayi | (accelOut.z & 1);
            sayi = sayi << 1;
            i = i + 1;
            if(i==16)
                break;
        }
        BITSET(TA2CCTL0 , CCIE);
        TA2CCR0 = 20;
        BITSET(TA2CTL , (TASSEL_1 | MC_1 | TACLR));
        __bis_SR_register(LPM4_bits | GIE); // Enter LPM4 w/ interrupt
        __delay_cycles(1);
        BITCLR(TA2CCTL0 , CCIE);
        BITCLR(TA2CTL , (TASSEL_1 | MC_1 | TACLR));
    } // End of while(1)
    i=0;
    counter++;
    randsayi = counter * counter;
    randsayi = randsayi + sayi ;
    randsayi = randsayi & 65535;
    randsayi = 0;
} //End of while(FOREVER)

```

APPENDIX 6: Scyther code for testing our proposed authentication protocol

```

hashfunction Xor;
usertype SessionKey;
usertype String;
const MAC: String;

protocol GroupProofAuthProtocol(Server, Reader, Tag)
{
    //Role of server
    role Server{
        var Rsr      : String; // Response from Server to Reader;
        var Ntr      : Nonce;  // Nonce value Generated on Tag
        var Nrt      : Nonce;  // Nonce value Generated on Tag

        const IDg    : String;
        const Kg      : SessionKey;
        const IDt    : String;
        const Kt      : SessionKey;

        recv_3(Reader, Server, Xor({ {Xor(Xor(IDg,Nrt ,Ntr), { IDt }Kt)}Kg
            }Kg,Xor(IDg,Nrt,Ntr)), Rsr, MAC);

        send_4(Server, Reader, Rsr, MAC);
        claim_s1(Server, Secret, IDg);
        claim_s2(Server, Secret, Kg);
        claim_s3(Server, Secret, IDt);
        claim_s4(Server, Secret, Kt);
        claim_s5(Server, Niagree);
        claim_s6(Server, Nisynch);
    };

    //RFID Reader Role
    role Reader {
        fresh Nrt    : Nonce; // Nonce value Generated on Reader
        var Ntr      : Nonce; // Nonce value Generated on Tag
        fresh Rsr    : String; // Response from Server to Reader;

        const IDg    : String;
        const Kg      : SessionKey;
        const IDt    : String;
        const Kt      : SessionKey;

        macro M1 = { Xor(IDg, Nrt) }Kg;
        // M1 = E(IDg ^ Nrt,Kg); // Encryption

        send_1(Reader, Tag, M1, Nrt, MAC);
        recv_2(Tag, Reader, {Xor(Xor(IDg,Nrt ,Ntr), { IDt }Kt)}Kg,Nrt,MAC);

        macro M2r = Xor(IDg, Nrt,Ntr);
        // M2r = IDg ^ Nrt ^ Ntr; // Concatenate

        macro M5 = { {Xor(Xor(IDg,Nrt ,Ntr), { IDt }Kt)}Kg }Kg;
        // M5 = D (M4, Kg); // Decryption

        macro M3r = Xor(M5, M2r);
        // M3r = M5 ^ M2r

        send_3(Reader, Server, M3r, Rsr, MAC);
        recv_4(Server, Reader, Rsr, MAC);

        claim_r1(Reader, Secret, IDg);
        claim_r2(Reader, Secret, Kg);
        claim_r3(Reader, Secret, IDt);
        claim_r4(Reader, Secret, Kt);
        claim_r5(Reader, Niagree);
        claim_r6(Reader, Nisynch);
    };

    // RFID Tag Role

```

```

role Tag {
  var Nrt      : Nonce;
  fresh Ntr    : Nonce;
  const IDg: String;
  const Kg: SessionKey;
  const IDt: String;
  const Kt: SessionKey;

  recv_1(Reader, Tag, M1, Nrt, MAC);
  match( {IDg}Kg, M1);
  macro M2 = Xor(IDg,Nrt ,Ntr);      // M2 = IDg ^ Nrt ^ Ntr;
  macro M3 = { IDt }Kt;              // M3 = E(IDt,Kt);
  macro M4 = {Xor(M2, M3)}Kg;        // M4 = E( (M2^M3 ),Kg );
  send_2(Tag, Reader, M4, Ntr, MAC);

  claim_t1(Tag, Secret, Kt);
  claim_t2(Tag, Secret, IDt);
  claim_t3(Tag, Secret, Kg);
  claim_t4(Tag, Secret, IDg);
  claim_t5(Tag, Niagree);
  claim_t6(Tag, Nisynch);
};
}

```

