

DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

RANDOM NUMBER GENERATION IN UHF RFID
TAGS

by
Mesut Can GÜRLE

April, 2015
İZMİR

RANDOM NUMBER GENERATION IN UHF RFID TAGS

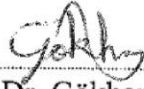
**A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Computer Engineering, Computer Engineering Program**

**by
Mesut Can GÜRLE**

**April, 2015
İZMİR**

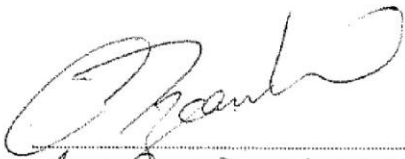
M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled “RANDOM NUMBER GENERATION IN UHF RFID TAGS” completed by MESUT CAN GÜRLE under supervision of ASST. 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 Master of Science.



Asst. Prof. Dr. Gökhan DALKILIÇ

Supervisor



Asst. Prof. Dr. M. Halil Özcanhan


(Jury Member)



Assoc. Prof. Dr. Mehmet

(Jury Member)

S. Ünlütürk



Prof. Dr. Ayşe OKUR
Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

I would like to thank to my thesis advisor Asst. Prof. Dr. Gökhan DALKILIÇ for his continuous suggestions, guidance and support during all phases of this research.

I would like to also thank to Asst. Prof. Dr. Mehmet Hilal ÖZCANHAN for his continuous help and support during all phases of this research.

Mesut Can GÜRLE

RANDOM NUMBER GENERATION IN UHF RFID TAGS

ABSTRACT

Authentication and encryption are frequently used by secure applications. Random number generators are frequently used by authentication and encryption algorithms. Algorithms that use low resource are needed because of the need to high quality random number generation in low capacity devices. Such generators can be used by all applications which need hard predicted and not frequently repeated numbers.

During this research existing random number generators are examined and a new random number generator which can work in low capacity devices has been developed. The developed generator is tested with statistical test suites which are generally used in literature.

Keywords: Authentication, encryption, random number generation, statistical randomness test, low capacity devices

UHF RADYO FREKANSI İLE TANIMLAMA ETİKETLERİNDE RASTGELE SAYI ÜRETİMİ

ÖZ

Güvenli uygulamalarda kimlik doğrulama ve şifreleme sıkça kullanılmaktadır. Kimlik doğrulama ve şifreleme algoritmalarında sıkça rasgele sayı üreteçleri kullanılmaktadır. Düşük kapasiteli cihazlarda kaliteli rasgele sayı üretme işlemini yapabilmek için daha az kaynağa ihtiyaç duyan algoritmalar gerekmektedir. Bu üreteçler tekrar etme ve tahmin edilebilme olasılığı düşük sayı ihtiyacı olan bütün uygulamalarda kullanılabilir.

Bu çalışma kapsamında varolan rasgele sayı üreteçleri incelendi, düşük kaynak ile çalışabilen bir rasgele sayı üretici geliştirildi. Geliştirilen bu üreteç, literatürde genellikle kullanılan test araçları ile test edildi.

Anahtar kelimeler : Kimlik doğrulama, şifreleme, Rastgele sayı üretimi, İstatistiksel rasgelelik testi, Düşük kapasiteli cihazlar

CONTENTS

	Page
THESIS EXAMINATION RESULT FORM	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZ	v
LIST OF FIGURES	viii
LIST OF TABLES	ix
CHAPTER ONE – INTRODUCTION	10
CHAPTER TWO – RFID TECHNOLOGY	12
2.1 RFID	12
2.1.1 RFID Tag Types and Standards.....	13
2.2 RFID Readers	14
2.3 Signaling.....	15
2.4 RFID Usage Area	17
CHAPTER THREE – RFID SECURITY	18
3.1 Known Attacks	18
3.2 Dealing with Security and Privacy issues	20
3.2.1 Tag Data Protection	20
3.2.2 RFID Reader Integrity	21
3.2.3 Personal Privacy	21
3.3 Authentication in RFID	22
3.3.1 Random Numbers	24
3.3.1.1 Pseudo Random Number Generators.....	24
3.3.1.2 True Random Number Generators.....	25
3.3.2 Random Number Generation.....	26
3.3.3 Random Number Generators	28

CHAPTER FOUR – PROPOSED SCHEME	29
4.1 The Inspired PRNG: Mersenne Twister	29
CHAPTER FIVE – PERFORMANCE, TESTING, EVALUATION RESULTS	34
5.1 Performance Results	34
5.2 Testing Results	37
CHAPTER SIX – CONCLUSION AND FUTURE WORK	41
REFERENCES	42
APPENDICES	48

LIST OF FIGURES

	Page
Figure 2.1 RFID antenna.....	12
Figure 2.2 RFID tags.....	14
Figure 2.3 RFID reader	15
Figure 3.1 A typical RFID set up	22
Figure 3.2 Sample RFID authentication.....	23
Figure 4.1 The proposed scheme	32

LIST OF TABLES

	Page
Table 2.1 EPC Fields.	16
Table 5.1 Comparison of operation types used in different PRNGs.....	35
Table 5.2 Total gate equivalent of our proposal	36
Table 5.3 Comparison of performance results.	36
Table 5.4 Gate equivalent of our proposal.	38
Table 5.5 Comparison of diehard and NIST tests.	39

CHAPTER ONE

INTRODUCTION

Lately, usage of mobile devices is increasing rapidly. Mobile devices are very popular among other devices, also they have various usage areas. Security and privacy issues in mobile devices are becoming more critical due to the increase in their usage. Mobile devices must support confidentiality, integrity and availability for privacy and security concerns. A lot of threats exist for mobile devices.

Radio Frequency Identification (RFID) is used for automated identification of objects and people. Radio Frequency Identification is a promising mobile technology. Data transmission takes place in wireless media. Data transmission is between RFID tag and RFID reader. RFID tag is a small microchip which can be attached to an antenna. A tag can be as small as 0.4 mm^2 and it is generally like a sticker (Takaragi, Usami, Imura, Itsuki & Satoh, 2001). RFID gives us opportunity to uniquely identify objects and easily do some automation. RFID usage has a lot of advantages over using barcode. For instance we can uniquely identify items using RFID and barcodes must be optically scanned by the readers. Barcodes need to be in a special position and careful contact with the reader. RFID does not require this kind of care. RFID reader can scan hundreds of tags per second with high accuracy and does not require the tag to be in a certain position with respect to the reader like the barcode scanner does.

RFID usage is increasing nowadays. Recent increase in usage of RFID made it more popular in respect to the security perspective of Organization for Economic Co-operation and Development (OECD, 2008). RFID has different constraints according to other technologies such as low energy consumption and low capacity. These constraints do not allow us to use high capacity required solutions. Scope of this work includes proposing a new pseudo random number generator that uses little die area and clock cycle. This scheme is also verified using well known statistical randomness tests. Test results are included in the work.

The thesis is organized in six chapters. The second chapter is about mobile devices and ubiquitous systems. Chapter three describes RFID security. Chapter four introduces the proposed scheme. Chapter five describes performance, testing and evaluation of results. Chapter six concludes the work and discusses the future work.

CHAPTER TWO

RFID TECHNOLOGY

2.1 RFID

Radio Frequency Identification is a wireless technology which helps us to transfer data, identify and track the tags. Tag contains identification information. Tags can be powered, some of them use local power resource like a battery and some of them can collect energy from radio waves.

Tags can have a factory assigned read only ID when the stage of production. This ID can be used like a key to match the value in the database. Also tags can have memory which can be read and written. System user is authorized to write data into the tag. Some programmable tags can be written once and can be read multiple times. Blank tags can be written by user.

RFID tags contain two parts. One of them is the integrated circuit. It is used to store and the process the data, collect power from reader signal and do some other specific functions (OECD, 2008). The other part is the antenna. Antenna is used to receive and transmit the signal. RFID tags can include a chip or a programmable processor for processing the data.

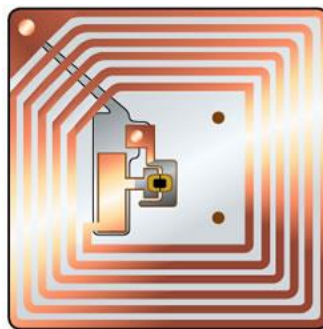


Figure 2.1 RFID antenna

The reader transmits an encoded signal to activate a tag. The tag receives the signal and responds with identification information and the other information if

exists. This data can be a unique tag serial number, a product related information, a batch number, etc.

2.1.1 RFID Tag Types and Standards

RFID tags can be grouped into three categories which are active, passive and semi passive. Active tags hold a battery. They also transmit signals periodically. Semi passive tags also hold a small battery on board. They are activated by a reader. When no reader exists, they stay passive. Passive tags are small and they do not hold a battery. So, they are cheaper than other tags and they are mostly preferred by applications. Passive tags are activated by the reader. They stand passive because of no battery.

RFID operates on different frequency bands. These are low frequency, high frequency and ultra-high frequency. RFID systems which operate in low frequency have shorter read range and slower data read rate. But, they operate well on metal and liquid surfaces.

Low frequency (LF) RFID systems operate between 30 KHz and 300 KHz. LF provides 10 cm read range. Standards for LF are ISO 14223 and ISO/IEC 18000-2. High frequency (HF) operates between 3 and 30 MHz Read range for HF varies from 10 cm to 1m. There are several standards for HF RFID systems such as ISO 15693 for tracking objects; ECMA-340 and ISO/IEC 18092 for NFC; ISO/IEC 14443 A and ISO/IEC 14443 for MIFARE and JIS X 6319-4 for FeliCa. UHF frequency band ranges are from 300 MHz to 3 GHz. UHF Gen2 standards for RFID operates between 860 and 960 MHz band. Read range of UHF tags are 12 m. EPCglobal Gen2 (ISO 18000-6C) UHF standard is the only standard which regulates UHF.



Figure 2.2 RFID tags

RFID tags are categorized into different classes. Class 0 and Class I tags represents the basic functionality. They are read only and passive tags. Class II tags are passive tags with additional functionalities such as memory and encryption. Class III tags are semi passive tags. They holds a battery. When they run out of energy, they are activated like passive tags. Class IV tags are active tags. They are capable of peer to peer communication with other active tags. Class V tags are known as readers. They can communicate with other tags and readers. In addition to communicating, they can power Class I, II and III tags (Sarma & Engels, 2003).

2.2 RFID Readers

RFID readers are also grouped into similar categories like tags. Some readers can either be passive or active. Passive readers can only receive the radio signals of battery powered active tags. Active readers transmit signals and also receive the authentication replies.

Readers can either be fixed or mobile. Fixed readers define a reading area which tags can go in or out. Mobile readers can be hand-held or mounted on a vehicle. Tag type and reader type define the RFID system. For example, using an active reader and passive tags on an RFID system is called as Active Reader Passive Tag (ARPT) system. This can vary like Active Reader Active Tag (ARAT) and Passive Reader Active Tag (PRAT) (Bachu, Saram & Sharma, 2013).



Figure 2.3 RFID reader

Most of UHF readers cost from \$500 to \$2000 depending on the features of the device. It depends on the type of the reader. Tags, antennas and cable are needed to build a complete system. Low and high frequency readers are range in price depending on different factors. A low frequency reader can be under \$100, while a fully functional standalone ones can be \$750. High frequency readers are typically between \$200 and \$300. A standalone reader can be about \$500 (RFID Journal, 2015).

According to functions in RFID System, two types of antennas can be used. These are tag antenna and reader antenna. Tag antenna both transfers the information and catch the wave for energy. Eroded or printed antennas are widely used and the dipole is the typical tag antenna structure. Some circularly polarized antennas for the tag may be preferred for special applications. Patch and spiral antennas are typical reader antennas. Linearly polarized antennas can be preferred for some special applications (Zhang, Jiao, Zhang & Wang, 2009).

2.3 Signaling

Low frequency tags operate in the 124-135 kHz range and have half a meter read range. High frequency tags operate at 13.56 Mhz and have up to a meter or more read range. Ultra high frequency tags operate in 860-960 Mhz and have up to ten meters read range (Juels, 2006).

Electronic Product Code (EPC) is a common type of data stored in the tag. RFID printer can write 96 bit data into the tag. First 8 bits are the header information which identifies the version of the protocol. Next 28 bits are organization information. The organization number is assigned by EPCGlobal consortium. Next 24 bits are object class. This identifies the type of the product. The last 36 bits are unique serial number for a specific tag. Last two fields are assigned by the organization. The total code can be used as a unique key into global database.

Table 2.1 EPC Fields

	Header	Organization Information	Object Class	Unique Serial Number
Number of Bits	8	28	24	36
Total Numbers		268,435,455	16,777,215	68,719,476,735

Generally more than one tag respond the reader. For example many products are hold in a basket. When it is time to check out, lots of products are being read at the same time. Some different protocols exist in collision detection. As one approach the reader can send the initialization command and a pseudo random delay for each individual tag. As adaptive binary tree approach the reader sends initialization symbol and then transmits one bit of identification information. Only tags match the identification information respond (Glover & Bhatt, 2006). Ideally only one tag would respond at a time. Both methods have advantages and disadvantages when multiple readers exist or used with multiple tags.

Three key factors play a significant role in RFID usage. Decreased cost of tags, increased performance of reliability and stable international standards. These standards are driven by the EPCGlobal.

2.4 RFID Usage Area

Many of us already use RFID applications. Some applications include proximity cards for building access, ignition keys for automobiles, some payment tokens. Some American Express and Mastercard credit cards uses RFID (Juels, 2006). Item level inventory tracking operation is very suitable for RFID usage. Items can be tracked during all steps of gathering the raw material, manufacturing process, supply chain and the individual end user. Some intelligent applications like fridge can use the RFID information. Some other smart systems can be built on the top of RFID technology.

As a result of the increase in the RFID usage, RFID technology can be used for new areas. New smart appliances and interactive objects can use RFID tags. RFID usage has lots of advantages and some limitations. These features can be used for new areas in the future.

CHAPTER THREE

RFID SECURITY

Because of the increase in development and use of RFID, makes security a big concern. RFID usage is not limited to inventory and stock tracking. Nowadays usage of RFID is extended to electronic passports and RFID embedded credit cards. The increase in usage of RFID raises concerns about security and privacy issues. RFID tags are dumb devices. They only listen and respond without any information about the sender. This issue reveals the problems with unauthorized access and tag data modification. Some attacks such as eavesdropping, denial of service, traffic analysis and spoofing are major threats to RFID tags.

3.1 Known Attacks

As a security concern, all applications are expected to provide data confidentiality, data integrity and privacy. Following known attacks can be considered as potential risks in RFID security.

Eavesdropping: Radio signals transmitted between tag and reader. This communication can be seen by other receivers. An unauthorized user can gain access to confidential data during the transmission. Researchers in the US has demonstrated a eavesdropping attack on an RFID credit card such as the cardholder's name and account information, could be skimmed if not properly encrypted (Heydt-Benjamin, Bailey, Fu, Juels & O'Hare, 2007).

Man-In-The-Middle Attack: Attacker intercepts the communication between two parties. The communication between RFID reader and tag can be monitored by attacker. Attacker can alter or inject messages into communication. A secure channel must be built for attacker not to mediate the traffic between parties.

Replay Attack: Attacker can replay a previously recorded message. Therefore third party can trigger an action using a valid message. To prevent from replay

attack, constant values must not be used. Using random numbers is a common way for prevention.

Location Tracing: Attacker can find a way to identify a tag. This identification information can be combined with location information. Therefore, tracing of a tag becomes a risk.

Forward Security: Once a secret information is captured, the data used in the communication can be revealed. Generation of secret keys is a big concern for forward security. Disclosure of a session key must not allow an attacker to reveal the future communication between parties.

Backward Security: Like forward security, the captured secret information can be used to reveal the past communication.

Synchronization Attack: A dishonest tag can be created using the captured secrets. The identity value in the tag is the same.

Physical Attack: This concern is mostly about hardware security issues. This is in another category.

Spoofing: Communication records can be used to perform tag spoofing. A software called 'RFDump' can be used to read and write on a not properly protected tag. By using this method, an RFID system can be fooled and the attacker can gain unauthorized access. For example, an RFID tag can be spoofed to show lower price. When it is used with eavesdropping, a replay attack can be a threat for an RFID tag. For a replay attack, attacker queries a tag and receives the information, this information can be retransmitted over and over later (Infosec, 2008).

Denial of Service Attack: This kind of attack can occur for large volumes of RFID data. A denial of service attack can be used to corrupt large volumes of tags. For example a 'kill' command can be transmitted to large amount of tags at a certain time

or a high power ratio frequency transmitter can be used to jam the RFID frequencies (Infosec, 2008).

Privacy Issues: Along with RFID is being used widely, people concern more about privacy issues. This concern includes storage and usage of their data. If a person can be identified by an RFID tag, individuals can be profiled and tracked.

3.2 Dealing with Security and Privacy issues

Solutions surrounding to RFID security can be categorized into three areas. They are tag data protection, reader integrity and personal privacy.

3.2.1 Tag Data Protection

Tag data protection is about the data RFID stores. For example a password can be set to protect the tag data. This approach can prevent being read without knowing the password. Only readers which know the password can gain access to the tag data. If all tags own a unique password, it means millions of passwords. Having such a high number of confidential information can contain storage risks. For a low capacity device it can be hard to store lots of data and check the correct password.

Physical locking of tag memory can be used to produce a read only chip and the information is embedded during the manufacturing process. In this method tag data cannot be overwritten.

Reader authentication in tag memory method can be implemented as follows. The author encrypts the data with its own private key, the data and meta-data is stored in the tag memory. When a reader wants to verify the information, tag sends the reader the data and reader verifies it. Although this solution can seem nice, it requires a key management system.

3.2.2 RFID Reader Integrity

For a solution method reader protection can be approved as reader and tag can agree on a response time or signal power level. Reader can change the frequency, so the communication is hard to eavesdrop. An authentication mechanism is needed to be implemented between reader and back end application.

RFID architecture can contain special devices to detect unauthorized behaviors such as unauthorized update on a tag.

3.2.3 Personal Privacy

Most privacy threads arise from RFID tags which hold unique identifier. These tags can easily associated with a person and the tracking of the person becomes easy. When a customer purchases an item, customer's identity can be matched to item's serial number. So this customer can be monitored and location of the customer can be revealed.

Special kill command is used to deactivate a tag and the tag cannot be re-activated again (Lu, Chen, Qiu & Jan, 2011). This command can disconnect the antenna, so the tag cannot be recognized anymore. So personal information cannot be viewed. But, stores generally do not prefer deactivating a tag because of returning of the product.

A tag can be covered with a special material to block certain frequencies. This approach makes tag detection very hard. But, it is not possible to cover all tags in all environments.

Active jamming can be used to block the any nearby readers. However, using such devices can be illegal. If too much power is used for operation, all nearby RFID systems would be affected.

A blocker tag can be used to deal with the unwanted RFID readers so the other tags can be kept secure from being read or any kind of attack. The blocker tag can have more than one antenna to communicate with multiple readers at the same time.

To unlock a locked tag, a key or pin is needed. The tag can be locked for a time and can be unlocked soon.

3.3 Authentication in RFID

Every consumer good in a supply chain has an identification sticker, which is used to keep track of it, during its journey from manufacture to the basket of a consumer. The stickers that used to be barcode papers are being replaced by RFID tags (Robert, 2006). According to a report RFID is a booming technology, acting as a part of ubiquitous systems (Das & Havrop, 2010). RFID rests upon a wireless technology where a tiny tag with an integrated circuit is energized through an antenna coil by a reader, as shown in Figure 3.1.



Figure 3.1 A typical RFID set up

The unique and sensitive ID information programmed on the tag is read and matched to a database on a remote server. The issue of security arises when the tag and the reader try to authenticate each other over an insecure, wireless medium. Generated pseudo random numbers (PRNs) and the sensitive tag identification number (ID) are transmitted, sometimes in obscured messages to avoid their capture. The low cost passive-UHF tags are reportedly the most popular (Das & Havrop, 2010). The main advantage of the RFID compared to other identification systems is that "it is capable of identifying items of different types and also distinguishing

between items of the same type without mistake, while not requiring physical or visual contact" (Lopez, Castro, Tapiador & Ribagorda, 2009a). Low-cost is the reason of the popularity of passive tags. However, low cost limits the computational capacity and power electronics of the tag; resulting in little resources to be spared for security, hence causing vulnerabilities for attacks (Lopez, Castro, Tapiador & Ribagorda, 2009a; Chien, 2007). Low cost passive tags should not be confused with the high cost tags used in e-passports.

After long efforts, the properties supported on passive tags have been ratified in ISO-18000-6 (ISO/IEC, 2010) and the EPCglobal Class-1 Generation-2 (Gen-2) standards (Gen-2 Standart Version 1.2.0, 2008). PRNs which are used in many security algorithms are supported in the Gen-2 standard. Efficient pseudo random number generators (PRNGs) are needed, in low cost tags which dedicate only a few thousand gates for security (Lopez, Castro, Tapiador & Ribagorda, 2009b; Martin, Millan, San, Entrena, Lopez & Castro, 2011). This is the motivation behind our proposal detailed later in thesis.

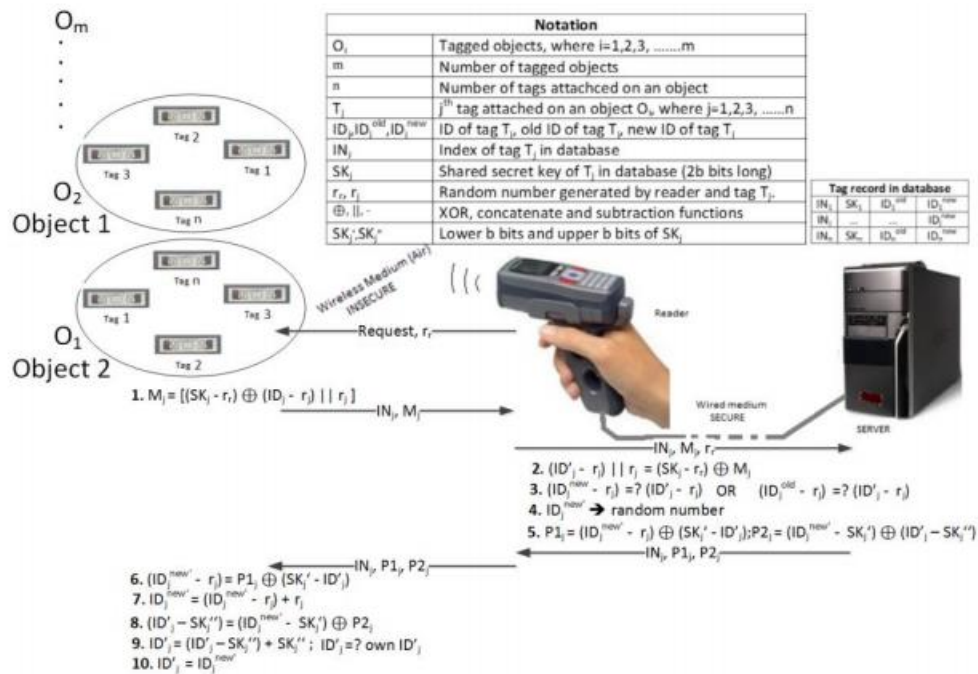


Figure 3.2 Sample RFID authentication

3.3.1 Random Numbers

Random numbers are the numbers that could not be predicted before they are generated. If a number is generated in a range of 1 to N, all the numbers within the range will have same probability to be generated. Random number generation is mostly important in mathematics, cryptography and computer games such as gambling. Random numbers can be generated by a physical device or computationally. Basically randomness exists in daily life including dice and coin flipping. However, it is hard to get a computer do something by chance. Computers follow the instructions blindly, so they are predictable. Necessary properties of random numbers are uniformly distribution and unpredictability. Pseudo Random Number Generators and True Random Number Generators are two main approaches to generating random numbers.

3.3.1.1 Pseudo Random Number Generators

Pseudo Random Number Generator (PRNG) is about building a mathematical model which aims to generate random numbers. PRNG is a deterministic algorithm. Number sequences which are generated by PRNG look random (DNRG, 2014). PRNG needs a seed value to initialize the model. Each output is used as seed value for the next iteration. Therefore, PRNG can generate numbers which show a good statistical behavior. After a period is reached, the generator starts to repeat the numbers generated before. As a result of determinism, PNRG generates the same sequence for the same seed. This deterministic nature can be desirable or undesirable according to the context. PNRG can produce many numbers in short time.

Researchers have invented lots of solutions in this domain. A PNRG which passes the next bit test is called Cryptographically Secure Pseudo Random Number Generator (CRPNRG) (PNRG, 2015). Passing next bit test means that it is computationally infeasible to predict the next bit. One of the algorithms used in PRNG is linear congruential generator. This algorithm is easy to implement and very

fast. A common way to implement a CRPNRG is using a secure block cipher in counter mode.

Some of PNRGs: Blum Blum Shub, Wichmann-Hill, Complementary-multiply-with-carry, Inversive congruential generator, ISAAC (cipher), Lagged Fibonacci generator, Linear congruential generator, Linear feedback shift register, Maximal periodic reciprocals, Mersenne twister, Multiply-with-carry, Naor-Reingold Pseudorandom Function, Park–Miller random number generator, RC4 PRGA, Well Equidistributed Long-period Linear, Xorshift, Rule 30 (Generators, 2015).

3.3.1.2 True Random Number Generators

True Random Number Generator (TRNG) extracts randomness from physical phenomena. This can be as simple as someone's mouse movements or keystrokes (Rock, 2005). However radioactive source is a really good physical phenomena and it can easily be detected. The Hotbits (Hotbits, 2015) service at Fourmilab in Switzerland is an excellent random number generator that uses this technique. Another suitable phenomenon is atmospheric noise. A normal radio can be used to easily pick it up. Random.org (Random, 2014) uses this approach. TRNGs are suitable for lotteries, sampling, games and generation of data encryption keys.

Generating a true random number mostly needs a hardware and because of the Input/Output waits, number generation takes a while (RNG, 2012). TRNGs are not deterministic. They don't need to be seeded. Physical devices are vulnerable to wear over time and generate biased outputs. To overcome the bias, most TRNGs have postprocessors (Sunar, Martin & Stinson, 2007).

Lately Intel has developed an instruction to generate true random numbers. RDRAND instruction reads the hardware generated random value from SP800-90A compliant random bit generator and stores it into the register (DNRG, 2014). The size of the random value is related to size of the register.

Some of TRNGs: Araneus Alea, ComScire, Entropy Key, Fox-IT Fox RandomCard, ID Quantique, Intel 810/815/840/845G chipsets, Intel RdRand instruction, LETech, QuintessenceLabs, TectroLabs, TRNG98, true-random.com, VIA Padlock engine (Generators, 2015).

3.3.2 Random Number Generation

Random number generation is a process of generating numbers that lack of a pattern. Some applications require uniformly distributed or unpredictable numbers. Best way to produce random number is measuring physical phenomena. Radioactive decay, sound samples in noisy environment, thermal noise and images of a lava lamp can be measured to generate unpredictable numbers. However special hardware is required to measure these metrics. Randomness is simulated by deterministic algorithms.

The work on generating random numbers in RFID tags can be divided into three. The first is the work on True Random Number Generators [TRNGs], where they use a physical characteristic of the tag to produce random numbers. The second is the work on Pseudo Random Number Generators [PRNGs], which a deterministic algorithm is used to produce random numbers. The third class is a blend of the first and second categories, where an output obtained from a TRNG is used as a seed to a PRNG. TRNGs mostly fail to provide good quality random numbers or tend to output the same numbers, if physical conditions are reproduced. In PRNGs on the other hand, if the initialization value or seed is guessed the generated RN can be guessed, as the whole process is mathematically deterministic. Therefore, a PRNG by itself is declared insecure without good sources for seeding (Jun & Kocher, 1999; Menenez, Oorschot & Vanstone, 1996). Since it is impossible to create true randomness from within a deterministic system, a source of true randomness is required for seeding (Jun & Kocher, 1999). For these reasons, TRNG outputs are used as inputs to PRNG algorithms in order to get good quality RNs (Holcomb, Burleson & Fu, 2009; Segui, Alfaro & Joancomarti, 2010). Random numbers produced by some tags are shown to look alike or repeat themselves, or can be

guessed (Segui, Alfaro & Joancomarti, 2010; Merhi, Castro & Lopez , 2011); the reason been limited resources of low-cost tags; in terms of memory, computational capability, power consumption and die area. Most schemes which produce good quality random numbers and pass certain tests; e.g. hashing and encryption algorithms, overwhelm the limited computational capabilities of resource stricken RFID (Lopez, Castro, Tapiador & Ribagorda, 2009b; Alomair, Lazos & Poovendran, 2007).

Two PRNG works support their schemes with widely used randomness tests (Lopez, Castro, Tapiador & Ribagorda, 2009b; Martin, Millan, San, Entrena, Lopez & Castro, 2011). These works also have detailed design and performance information, which can be compared with our work. Two previous works which fall into the third category do not provide the same information but concentrate on physical characteristics (Che, Deng, Tan & Wang, 2008; Segui, Alfaro & Joancomarti, 2010). In these works, the much criticized LFSR function is used as a PRNG scheme, which takes only one bit TRN as input. Many attacks on LFSRs have been announced, but they are outside our scope.

Our work comes in at this point, which also falls into the third category that uses a TRNG extracting method, supported by a PRNG algorithm. It has been discovered that the SRAM memory of a tag can be a source of TRNs (Holcomb, Burleson & Fu, 2009). Some bits are shown to settle randomly to an either low or high voltage level. The obtained TRNs have low entropy and fail the randomness tests. For this reason, the authors feed the TRNs to a hash function, which tests show that good quality PRNs are obtained. For now assuming hash algorithms as not suitable for low cost tags, we attempt to replace the hashing algorithm with an ultra-light scheme. The definition of ultra-light tags is in (Lopez, Castro, Tapiador & Ribagorda, 2009a).

3.3.3 Random Number Generators

Linear Congruential Generators is widely used pseudo random number generation technique. Numbers are calculated with a discontinuous linear equation. The theory is simple and fast. Blum Blum Shub generator is cryptographically secure pseudorandom bit generator. It passes the next-bit test.

Using a symmetric block cipher is a popular approach as the hearth of PRNG mechanism. CTR mode and OFB mode are widely used to build a PNRG. A standardized block cipher is a good candidate for a secure PRNG.

Using a stream cipher is an alternative way to generate pseudorandom number generator. If you are dealing with a block of data stream ciphers may be more appropriate.

CHAPTER FOUR

PROPOSED SCHEME

4.1 The Inspired PRNG: Mersenne Twister

Our proposal is based on the Mersenne Twister [MT] (Matsumoto & Nishimura, 1998); which has properties that are feasible in low cost tags. MT is a well-known algorithm proposed in 1997 by Matsumoto and Nishimura that is classified as a good PRNG (Matsumoto, Nishimura, Hagita & Saito, 2005). The name MT comes from the fact that a Mersenne prime is used as its period. MT achieves fast generation of pseudorandom numbers with a long period. Many variants of it have been introduced for better speed and cryptographic security (Matsumoto, Nishimura, Hagita & Saito, 2005; Panneton, Ecuyer & Matsumoto, 2006). MT has better equidistribution and “bit-mixing” properties than its predecessors, for equivalent period length and speed (Panneton, Ecuyer & Matsumoto, 2006). MT is based on linear recurrences in F_2 (finite field with two elements, 0 and 1), where arithmetic operations are arithmetic modulo 2^m (Panneton, Ecuyer & Matsumoto, 2006). Binary recurrences are suitable for implementing bitwise operations in low-cost tags. MT is a special Twisted Generalized Feedback Shift Register [TGFSR] that takes an incomplete array to realize a Mersenne prime as its period. It uses an inversive-decimation method for testing the primitivity of a characteristic polynomial of linear recurrence with a computational complexity $O(p^2)$, where p is the degree of the polynomial. In general, the linear generators over F_2 are represented by the matrix linear recurrence (Panneton, Ecuyer & Matsumoto, 2006):

$$x_i = Ax_{i-1}, \quad (4.1)$$

$$y_i = Bx_i, \quad (4.2)$$

$$u_i = \sum_{l=1}^w y_{i,l-1} 2^{-l} = y_{i,0}, y_{i,1}, y_{i,2} \dots \quad (4.3)$$

$x_i = (x_{i,0}, \dots, x_{i,k-1})^T \in F_2^k$ and $y_i = (y_{i,0}, \dots, y_{i,w-1})^T \in F_2^w$ are the k -bit state and the w -bit output vector at step i . With elements also in F_2 , A and B are a $k \times k$ transition matrix and a $w \times k$ output transformation matrix; where k and w are

positive integers and $u_i \in [0, 1)$ is the output at step i . All operations in (1) - (3) are performed in F_2 and each element of F_2 is represented as one bit. The goal is to find matrices A and B whose recurrence can be implemented efficiently, the produced sets have good uniformity and the overall generator passes empirical statistical tests. Further mathematical argument shows that MT is a special case of Well Equidistributed Long-period Linear [WELL] generators (Panneton, Ecuyer & Matsumoto, 2006). The details of the argument is beyond the scope of this paper but it is shown that MT has a long period of $2^{19937}-1$, a 623 dimensional equidistribution up to 32-bit accuracy, and generates output free from long-term correlations.

The steps of the MT; however, is of interest because of its suitability for the simple bit-wise operations a tag can accommodate. MT works in two parts; a recurring and a method called tempering part. In the first three steps an array is initialized and concatenated. In Step 4, each generated array x during the recursive steps is multiplied by a transformation matrix T , to obtain a tempered matrix $z = xT$. Some of the transformation matrices of MT are equal to each other, which makes MT a special case of WELL generators (Panneton, Ecuyer & Matsumoto, 2006). The simplified steps and the tempering stage of the MT algorithm are given below:

- Step 0: Create bitmask for upper and lower bits,
- Step 1: Initialize the $x[i]$ array with seeds of nonzero values,
- Step 2. Concatenate the upper bits of the previous array $x[i]$ with the lower bits of the iterated array $x[i + 1]$,
- Step 3. Calculate the next state array $x[i]$,
- Step 4. Carry out tempering part:
- Step 5. Increment i by 1
- Step 6. Repeat the process and go to step 2, until i equals n .

The parameters u, s, t, l are tempering bit shifts and b and c are tempering bit masks. These parameters are experimentally tested values for maximally-equidistributed generators (Panneton, Ecuyer & Matsumoto, 2006). Tempering is carried out in Step 4, in order to improve the distribution of the sequences generated

from the recursions because MT has some weaknesses. Firstly, the initial state of MT has too many zeros therefore the generated sequences also contain many zeros for more than 10000 generations. Our work removes this weakness by supplying non-zero, random number initial inputs and removes the matrix recurrences completely. Secondly, for seeds chosen systematically as 0, 20, 30 the output sequences are correlated. Finally, MT is not preferred for cryptographic purposes because it is easy to predict the next state given the present outputs in MT. To fix the problems many variations of MT have been proposed. One of the suggestions is to have the outputs of MT go through a hash function. This is why work (Holcomb, Burleson & Fu, 2009) feeds the obtained TRNGs into a hash function. It is clear that if the generator is initialized with uniform random bits, the probability of getting many zero bits or correlated output is quite small. Only seeding the algorithm with TRNs and the tempering can improve the distribution of the final pseudo random numbers generated (Panneton, Ecuyer & Matsumoto, 2006).

Our work is inspired from Mersenne Twister (MT) which is a PRNG that satisfies all the requirements to be certified as a good PRNG (Matsumoto & Nishimura, 1998). MT achieves fast generation of pseudo-random numbers with a long period. Many variants of MT have been introduced for better speed and cryptographic security (Matsumoto, Nishimura, Hagita & Saito, 2005; Tiny Mersenne Twister, 2011).

Our work is based on the strategy of using extracted TRNs from hardware as a seed for the MT. We aim to remove the deterministic, iterative, pseudo part of the MT and replace it with TRNG seeding. Our proposed scheme is shown in Figure 4.1. It consists of three simple bit-wise operations; XOR, AND and rotation (circular shift). Thus, the hash function is replaced by a scheme that consumes less die area and clock cycles.


```

x := TRN ; initialization0
x := ROTl (x,x) ; for improving the seed
y := x ⊕ ROTr (x,u) ; MT tempering 1
y := y ⊕ (ROTr (y,s) AND b) ; MT tempering 2
y := y ⊕ (ROTr (y,t) AND c) ; MT tempering 3
y := y ⊕ ROTr (y,l) ; MT tempering 4
z := y ⊕ ROTl (y AND b,p) ; additional rotation
u = 7Fh, s = 07h, t = 1Fh, l = 3FFFFh, p = 7Fh.
b = 9D2C5680H.
c = EFC60000H.

```

Figure 4.1 The proposed scheme.

MT is a special case of Wells function (Panneton, Ecuyer & Matsumoto, 2006). In short, it is an algorithm which treats an input as a matrix and twists it right and left with corrective temperings in between to produce good PRNs. For a complete mathematical analysis the reader is referred to the original documents (Matsumoto & Nishimura, 1998). A more compact version of MT has also recently been released as TinyMT (Tiny Mersenne Twister, 2011). All versions of MT are known to pass the randomness tests.

The matrix iterations of the original MT are eliminated in our scheme. The tag is not capable of performing the matrix operations. To compensate, our scheme improves the shift operations with rotation operations of the temperings. ROTr(x,y) is a simple bit-wise shift operation to the right, where the least significant bit (LSB) is wrapped around to most significant bit [MSB]. The value x is the number to be rotated and y is the hamming weight (number of ones in) of y. The operation is simple because y is loaded into a control register and examined bit by bit, while it is shifted. If the tested bit is a one, x is rotated once; if not x remains un-rotated.

Overall, the rotation operation executes permutation and XOR-AND operations provide substitution, on the operands. Thus, the input goes through a sequence of permutations and substitutions, as in modern hashing and encryption algorithms. The direction of rotation operations and coefficients of MT are carefully designed and well-defended in (Matsumoto & Nishimura, 1998; Panneton, Ecuyer & Matsumoto, 2006). In our scheme, an extra rotation is necessary to compensate for the lost iteration, which brings little computational cost.

In our work, different number of rotation operations, different directions against different number of rotations and AND coefficients have been tested, until the scheme with the best randomness results was obtained. As it will be revealed, the original directions and masking coefficients yield the best results. The scheme can be executed as a sequential algorithm, where the coefficients are given as immediate constant operands.

CHAPTER FIVE

PERFORMANCE, TESTING AND EVALUATION OF RESULTS

Our proposal uses a 16-bit tag architecture to obtain a 32-bit PRN, because we do not believe that the latest 32 or 64 bit, state of the art technologies used for microprocessor production, can be widely used in low cost tag production. We assume that 16-bit PRNs can be obtained from our 32 PRNs by either taking the lower 16 bits, or by XORing the higher 16 bits with the lower 16 bits, as in previous work (Lopez, Castro, Tapiador & Ribagorda, 2009a; Martin, Millan, San, Entrena, Lopez & Castro, 2011). But we claim that 32 bit, good quality PRNGs that pass the popular randomness tests are possible in tags, as shown later in Section 4.1. Moreover, only 32 bit randomness tests are accepted, by the community. According to Gen-2 specifications, the tag is expected to provide only a 16-bit PRNG but the period of a 16 bit generator is much shorter than that of a 32 bit generator. Not only the short period hence the reduced randomness of a 16 bit PRN but the overall security supported by the Gen -2 standard is considered highly inadequate, as also underlined in other work (Lopez, Castro, Tapiador & Ribagorda, 2009b; Martin, Millan, San, Entrena, Lopez & Castro, 2011). The maximum number of gates and clock cycles allocated for security in a tag are a few thousands gates and 1800 clocks (Sarma, Weis & Engels, 2002; Feldhofer, Dominikus, Wolkerstorfer, 2004). These cannot be used for only generating a random number, because space and time must be left for other processes such as future stronger authentication steps etc. Using the above guidelines, first performance results and then randomness test results are compared.

5.1 Performance Results

Estimation for the die area of an integrated circuit can be obtained by using the gate equivalents (GE) (Moradi & Poschmann, 2010). The GE of each logic gate and the total GE for each operator are known (Paar, Poschmann & Robshaw, 2009). The same GE metrics are used in the other work that we aim to compare our results. The common timing metric, on the other hand is the clock cycle used. A 16-bit ALU

requires two clock cycles to finish a 32 bit AND or XOR operation; but n clocks for an n number of circular shifts. Using the results of the above metrics, the area-delay product; i.e. complexity, of a tag can be determined.

Our scheme of Figure 4.1 requires only XOR, AND and circular shift operators. Table 5.1 shows the operation types used in the previous schemes against ours. Obviously, the multiplication and the finite state machine requirement of the Akari-x hint an overwhelming load on the tag. The Lamed scheme uses three simple operations like ours but requires input, control and rotation units for iterative work. While Akari-x schemes require memory for an initialization vector, Lamed requires two 32-bits space and ours requires one 32-bits and four extra 16-bits space (Martin, Millan, San, Entrena, Lopez & Castro, 2011; Lopez, Castro, Tapiador & Ribagorda, 2009b).

Table 5.1 Comparison of operation types used in different PRNGs.

	Operation types used	Iteration	Memory
Akari1A	SUM, OR, MULTIPLY, SHIFT	For loop, FSM	Initialization vector
Akari1B	SUM, OR, MULTIPLY, SHIFT	For loop, FSM	Initialization vector
Akari2A	SUM, OR, MULTIPLY, SHIFT, XOR	For loop, FSM	Initialization vector
Akari2B	SUM, OR, MULTIPLY, SHIFT, XOR	For loop, FSM	Initialization vector
Akari2C	SUM, OR, MULTIPLY, SHIFT, XOR	For loop, FSM	Initialization vector
Lamed	SUM, XOR, SHIFT	Control Units	Initialization vector, key
Ours	XOR, AND, SHIFT	Sequential	u, s, t, l, p

The total GE required for our scheme is calculated to be 416 gates, as shown in Table 5.2. Every 1000 GE adds \$0.01 to the cost and power consumption is

proportional to the number of gates (Lopez, Lim & Li, 2008). Hence our scheme is well in the low cost category (Sarma, Weis & Engels, 2002). Later in Table 5.3, this result is compared to previous work together with clock cycles.

Table 5.2 Total gate equivalent of our proposal

Operator	# Used	Logic	GE	16-Bit Total
Register	2	Flip Flop	5.33	170.56
Shifter	1	Flip Flop	5.33	85.28
AND	1	Gates	1.33	21.28
XOR	1	Gates	2.67	42.72
Total				319.84
Control	1	Gates	30%	95.95
Grand Total				415.79

The number of clock cycles spent while obtaining a random number in our scheme is obtained from Figure 4.1 as the total of five XOR, three AND operations and the sum of x, u, s, t, l, p number of rotations. The TRN is read into register x at tag energizing time and is not in the RN generation phase. The XOR and AND operations consume 16 clocks. Although the sum of u, s, t, l and p are constant, total cycles consumed during rotation operations is dependent on the hamming weight of the value x, read from the SRAM. Assuming an average of 16 ones in the initial seed x, the total of rotation number is 56. When checked against the declared limit 1800, the proposed 72 clock cycle scheme is definitely in the ultra-light category (Sarma, Weis & Engels, 2002). This value is shown in Table 5.3, together with the performance values of the previous works.

Table 5.3 Comparison of performance results.

Scheme	Area GE	Die Area μm^2	Power Consum.	Delay Cycles	Through put (Kbps)	Complexity (GE\timesDelay)	Op. Types & Control
Akari1A	76	1494	47.35	644	24.24	306,544	Complex

Table 5.3 Comparison of performance results. (continue)

Akari1B	524	1643	54.61	644	3.55	337,456	Complex
Akari2A	824	2582	57.38	466	31.37	383,984	Complex
Akari2B	891	2794	76.95	466	5.50	415,206	Complex
Akari2C	903	2831	72.33	466	3.01	420,798	Complex
Lamed	1566	4916	157.19	220	71.35	344,520	Complex
Ours	416	1306	41.75	72	18.95	29,952	Simple

The delay clock cycles of our scheme are the smallest, by a far margin. The GE equivalent of our work is also smaller than the others. The area-delay product is a measure of complexity (Feldhofer & Wo0lkerstorfer, 2009). Table 5.3 shows that our scheme has the lowest complexity, enough for leaving space for other security functions as well. The die area, power consumption and throughput values have been calculated with the same metrics used in Akari-x, to base the comparisons on equal ground. Having the smallest GE, our proposal has a smaller die area and consumes less power.

The area-delay product for hashing functions is very high, even for 32-bit architectures (Feldhofer, Dominikus, Wolkerstorfer, 2004; Feldhofer & Wolkerstorfer, 2009). Their complexity values indicate clearly that encryption and hashing schemes are not in the ultra-light category. Therefore, the work of (Holcomb, Burleson & Fu, 2009) has not been included in the comparisons.

5.2 Testing Results

In our randomness tests, we used two sets of inputs to reach the best scheme. At first, an input set from <http://random.org> was used as a preliminary test to differentiate the failing schemes. Then a second set with low entropy (0.00) was used, similar to the entropy of the set in (Holcomb, Burleson & Fu, 2009). Many of those schemes that performed well with RN inputs faltered with low entropy inputs. We selected and improved only those schemes that passed the randomness tests with

low entropy inputs, finally reaching the best solution. It should not be missed that LAMED and Akari-X use random.org inputs for obtaining a RN from a RN, but RN seeds are not available in tags.

In testing PRNGs the ENT (Walker, 1998), Diehard test suits (versions 1, 2) (Marsaglia, 2003) and NIST (Rukhin, Soto, Nechvatal, Smid, Barker, Leigh, et al, 2001) randomness test are used. We performed all four tests to compare with the previous work. First version of Diehard has 15 different tests, which output a "p-value" for each test. The p-values are in the range 0.0-1.0, where a result close to these either extreme values is considered to be an unsatisfactory result. The second version of Diehard gives an overall p-value, where a value less than 0.1 is considered to be a fail. The NIST test also gives p-values, which are expected to be distributed between 0.1 and 0.9. The NIST test also outputs "proportion" values, which should be above 0.95. Any undesirable result is marked with a "*" next to the proportion value.

Table 5.4 compares the ENT results of our work against Akari-x and Lamed results. ENT is relatively a more relaxed test suite, than the Diehard and NIST tests. All schemes pass the tests with satisfactory results. To expose the difference in testing results, we move on to the more strict tests.

Table 5.4 Gate equivalent of our proposal.

ENT Test Results	Lamed	Akari 1/2	Ours
Entropy (bits/byte)	7.999999	8.000000/8.000000	7.999999
Compression Rate	0%	0% / 0%	0%
X2 Statistics	256.90 (50%)	259.09 (41.70%)/250.99 (55.93%)	212.60 (97.52%)
Arithmetic Mean	127.5024	127.4976/127.5031	127.5002
Monte Carlo Estimation	3.141474228	3.141447036/3.141512474	3.141529759
Serial Correlation Coefficient	-0.000023	-0.000026/0.000013	0.000009

For the first Diehard test we used the evaluation criteria used in (Alani, 2010). Because of the detailed output of the Diehard test, the test results and inputs are posted at appendix section. According to the evaluation criteria a score is given for each of the test result and their sum is calculated. We calculated the scores of the previous work by examining their declared results and summarized the overall sums in Table 5.5. For 15 tests, the maximum score is 15. It is worth to mention here that most of our preliminary schemes which scored high with inputs from random.org, scored very low with low entropy inputs. The scheme of Figure 4.1 scored (10.94) very close to original MT (score 11.84), in Diehard version 1 results. Therefore, it was further compared with the previous works, even though the inputs of previous works are RNs and not low entropy values.

Table 5.5 Comparison of diehard and NIST tests.

	Diehard1 Score	Diehard2 Score	NIST
Akari1A	12.4	0.353	Pass
Akari1B	12.4	0.353	Pass
Akari2A	7.5	0.082	Pass
Akari2B	7.5	0.082	Pass
Akari2C	7.5	0.082	Pass
Lamed	13.0	0.778	Pass
Ours	10.94	0.224	Pass*

The 32-bit version of Lamed performs the best, while its 16-bit version performance is poor. But because it is 16-bit, it is not included in these comparisons. Akari1 versions are also satisfactory, while the Akari2 versions perform well below our proposed scheme. Our scheme's result of 10.94 out of a 15 is far from being unsatisfactory, after taking its very low complexity value into consideration. Diehard version 2 scores are all in the satisfactory range, except the Akari2 family. Akari2 results are below the accepted limit of 0.1 and should not be considered to pass the Diehard tests.

The NIST test results are also very detailed reports. Therefore our results are posted in detail at appendices section. The previous work results are also posted on their referenced pages. It is acceptable for a scheme to fail a few tests out of a 188; i.e. a scheme failing two individual tests cannot be considered to not pass the overall strict NIST test (Kohlbrenner & Gaj, 2004). The results of previous work are declared to have passed the tests. Our test results however, show that our proposed scheme fails the Rank and Universal tests, as pointed out by the "*" mark on their right. Nevertheless, the other test results of the NIST test are favorable and our proposed can be evaluated as it has passed the NIST test.

According to testing output, the scheme is simple and sequential, yet its performance equals similar previous works. With equal performance, our proposal uses less die area and clock cycles, proving more suitable for low-cost tags. The proposed generator takes low-entropy seeds extracted from a physical characteristic of the tag and produces output that passes popular randomness tests.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

This work outlines a new random number generator that is feasible in low-cost RFID tags. The obtained low complexity, power consumption and die area results are an indication that the proposed scheme does not violate the resource limits of the ultralight tags. Our scheme takes nonrandom numbers as seeds and produces random numbers. Previous work use random number inputs, which are not available in RFID tags. The randomness test results of our proposed scheme are satisfactory, considering the nonrandom inputs used for seeding. Our scheme is available until a suitable hashing or encryption algorithm that is feasible in low cost tags.

Future work involves the detailed design and implementation of the proposed scheme. Also some work can be carried out until the scheme is further improved to pass all of the NIST tests. Efforts of encryption designs suitable for producing random numbers in RFID tags are intensified, but our scheme is available until such a low cost solution is found.

REFERENCES

- Alani, M. M., (2010). Testing randomness in ciphertext of block-ciphers using DieHard tests. *International Journal of Computer Science and Network Security*, 10/4 (8), 53-57.
- Alomair, B., Lazos, L. & Poovendran, R., (2007). Passive attacks on a class of authentication protocols for RFID. *International Conference on Information Security and Cryptology – ICISC'07*, 102-115.
- Bachu, Vinay Kumar, Saram, Sunil, Sharma, N.V.S.Shravan Kumar, (2013). A review of RFID technology. *International Journal of Engineering Sciences & Research Technology*, 2760-2762.
- Che, W., Deng, H., Tan, X. & Wang, J., (2008). A random number generator for application in RFID tags. *Networked RFID Systems and Lightweight Cryptography, Springer-Verlag*, 16, 279-287.
- Chien, H. Y., (2007). SASI: a new ultralightweight RFID authentication protocol providing strong authentication and strong integrity. *Transactions on Dependable and Secure Computing, IEEE*, 4, 337–340.
- Das, R. & Havrop, P., (2010). *RFID Forecasts, players and opportunities 2011-2021*. Retrieved March 27, 2014 from http://www.idtechex.com/research/reports/rfid_forecasts_players_and_opportunities_2011_2021_000250.asp
- DNRG (2014). *Intel Digital Random Number Generator (DNRG)*. Retrieved March 1, 2015 from: https://software.intel.com/sites/default/files/managed/4d/91/DRNG_Software_Implementation_Guide_2.0.pdf

- Feldhofer, M., Dominikus, S. & Wolkerstorfer, J., (2004). Strong authentication for RFID systems using the AES algorithm. *Lecture Notes in Computer Science, 3156*, Springer, 357-370.
- Feldhofer, M. & Wolkerstorfer, J., (2009). Hardware implementation of symmetric algorithms for RFID security. *RFID Security: Techniques, Protocols and System-on-Chip Design, III*, Springer, 373-415.
- Generators, (2015). *List of random number generators*. Retrieved Feb 28, 2015 from: http://en.wikipedia.org/wiki/List_of_random_number_generators.
- Gen-2 Standart Version 1.2.0, (2008). *Class-1 Generation 2 UHF Air Interface Protocol Standard "Gen-2", Version 1.2.0*. Retrieved March 22, 2014 from <http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2>.
- Glover, B. & Bhatt, H, (2006). *RFID Essentials*, 88-89.
- Heydt-Benjamin, T. S., Bailey, D.V., Fu, K., Juels, A. & O'Hare, T. (2007). Vulnerabilities in first-generation RFID-enabled credit cards. *Proceedings of the 11th International Conference on Financial Cryptography and 1st International Conference on Usable Security*, 2-14.
- Holcomb, D. E., Burleson, W. P. & Fu, K., (2009). Power-up SRAM state as an identifying fin-gerprint and source of true random numbers. *Transactions on Computers, IEEE*, 58(9), 1198-1210.
- Hotbits, (2015). *Genuine Random Numbers: generated by radioactive decay*. Retrieved Feb 28, 2015 from: <https://www.fourmilab.ch/hotbits/>.
- Infosec, (2008). *RFID security*. Retrieved May 19, 2014 from: <http://www.infosec.gov.hk/english/technical/files/rfid.pdf>
- ISO/IEC, (2010). Retrieved April 3, 2014 from http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=46149.

- Juels, A., (2006). RFID Security and privacy: a research survey. *IEEE Journal on Selected Areas in Communication* 2006, 24, 381-394.
- Jun, B. & Kocher, P., (1999). *The intel® random number generator*. Cryptography Research, White Paper.
- Kohlbrenner, P. & Gaj, K., (2004). An embedded true random number generator for fpgas. *Proceedings of the 12th International Symposium on Field Programmable Gate Arrays, ACM/SIGDA*, 71-78.
- Lopez, P. P., Lim, P. T. & Li, T., (2008). Providing stronger authentication at a low-cost to RFID tags operating under the EPCglobal framework. *Embedded and Ubiquitous Computing Conference, IEEE/IFIP International*, 159-167.
- Lopez, P. P., Castro, J. C. H., Tapiador, J. E. & Ribagorda, A., (2009a). An ultra light authentication protocol resistant to passive attacks under the Gen-2 specification. *Journal of Information Science and Engineering*, 25, 33-57.
- Lopez, P. P., Castro, J. C. H., Tapiador, J. E. & Ribagorda, A., (2009b). LAMED a PRNG for EPC class-1 generation-2 RFID specification. *Computer Standards & Interfaces*, 31/1, 88-97.
- Lu, J., Chen, Y., Qiu, Z. & Jan, J., (2011). A secure RFID deactivation/activation mechanism for supporting customer service and consumer shopping. *International Conference on Broadband, Wireless Computing, Communication and Applications 2011*, 405-410.
- Marsaglia, (2003). *The marsaglia random number CDROM including the DIEHARD battery of tests of randomness*, Diehard ver1:<http://stat.fsu.edu/pub/diehard>
- Martin, H., Millan, E. S., San, M., Entrena, L., Lopez, P. P. & Castro, J. C. H., (2011). AKARI-x: a pseudorandom number generator for secure lightweight systems. *17th International On-Line Testing Symposium (IOLTS), IEEE*, 228-233.

- Matsumoto, M. & Nishimura, T., (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* - Special issue on uniform random number generation, 8(1), 3-30.
- Matsumoto, M., Nishimura, T., Hagita, M. & Saito, M., (2005). Cryptographic Mersenne Twister and Fubuki stream/block cipher, *International Association for Cryptographic Research Cryptology ePrint Archive*, 165-165.
- Menenez, A. J., Oorschot, P. C. & Vanstone, S. A., (1996). *Handbook of Applied Cryptography*, 5, 169-190.
- Merhi, M., Castro, J. C. H. & Lopez, P. P., (2011). Studying the pseudo random number generator of a low-cost RFID tag. *International Conference on RFID-Technologies and Applications, IEEE*, 381-385.
- Moradi, A. & Poschmann, A., (2010). Lightweight cryptography and DPA countermeasures: a survey. *Lecture Notes in Computer Science, 6054*, Springer, 68-79.
- Organisation for Economic Co-operation and Development. (OECD), (2008). *OECD ministerial meeting on the future of the internet economy*. Retrieved May 16, 2014 from: <http://www.oecd.org/internet/ieconomy/40892347.pdf>
- Paar, C., Poschmann, A. & Robshaw, M. J. B., (2009). New designs in lightweight symmetric en-cryption. *RFID Security: Techniques, Protocols and System-on-Chip Design*, 3, Springer, 349-371.
- Panneton, F., L'Ecuyer, P. & Matsumoto, M., (2006). Improved long-period generators based on linear recurrences modulo 2. *ACM Transactions on Mathematical Software*, 32/1, 1-16.
- PNRG, (2015). *Pseudo Random Number Generators*. Retrieved Feb 27, 2015 from: <http://sqlity.net/en/2209/pseudo-random/>.

- Random, (2015). *Introduction to Randomness and Random Numbers*. Retrieved Feb 28, 2015 from: <https://www.random.org/randomness/>.
- RFID Journal, (2015). *RFID Frequently Asked Questions*. Retrieved March 7, 2015 from: www.rfidjournal.com/faq/show?86.
- RNG, (2012). *Random Number Generation: Types and Techniques*. Retrieved March 1, 2015 from: <http://digitalcommons.liberty.edu/cgi/viewcontent.cgi?article=1311&context=honors>.
- Robert, C. M., (2006). Radio frequency identification. *Computers and Security, Elsevier*, 25, 18–26.
- Rock, Andrea (2005). *Pseudorandom number generators for cryptographic applications*. Retrieved March 1, 2015 from: <https://www.rocq.inria.fr/secret/Andrea.Roeck/pdfs/dipl.pdf>.
- Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., et al., (2001). *A statistical test suite for random and pseudorandom number generators for cryptographic applications*. Retrieved May 21, 2014 from <http://csrc.nist.gov/rng/>, Technical Report.
- Sarma, S. & Engels, D. W., (2003). On the future of RFID tags and protocols. Technical report, Auto-ID center, Massachusetts Institute of Technology, 2003.
- Sarma, S. E., Weis, S. A. & Engels, D. W., (2002). RFID systems and security and privacy implications. *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science*, 2523, 454-470.
- Segui, J. M., Alfaro, J. G. & Joancomarti, J. H., (2010). Analysis and improvement of a pseudo-random number generator for EPC gen2 tags. *Financial Cryptography and Data Security 2010 Workshops, Lecture Notes in Computer Science, Springer-Verlag*, 34-46.

- Segui, J. M., Alfaro, J. G. & Joancomarti, J. H., (2011). A practical implementation attack on weak pseudorandom number generator designs for EPC gen2 tags. *International Journal of Wireless Personal Communications*, 59/1, 27-42.
- Sunar, B., Martin, W. & Stinson, D., (2007). A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks. *IEEE Transactions on Computers*, 56/1, 109-119.
- Takaragi, K., Usami, M., Imura, R., Itsuki, R., & Satoh, T. (2001). An ultra small individual recognition security chip. *IEEE Micro*, 43–49.
- Tiny mersenne twister*, (2011). Retrieved June 2, 2014 from <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/TINYMT/index.html>.
- Walker, J., (1998). *Randomness battery*. Retrieved June 18, 2014 from <http://www.fourmilab.ch/random/>.
- Zhang, M, Jiao, Y, Zhang, F & Wang, W, (2009). *Development and Implementation of RFID Technology*, 1, 554.

APPENDICES

PRNG TESTING RESULTS

A. SAMPLE INPUT DATA

Sample input data can be found below. The numbers are in hexadecimal format. This sample data is used for testing Diehard Test Suite. The exact input data is 10 Mb and can be found at <http://srg.cs.deu.edu.tr/publications/2012/prng/>.

A1 06 6C 7C AD 94 F3 38 8E 10 76 49 8B D4 3B 3A
42 E6 00 5F 57 15 58 F0 17 95 5A F0 C7 86 04 C8
6C A7 CB 49 F7 12 EF BB 2A 50 3B 5B 02 2F 25 2F
32 CC ED 4C 2C 2C FA 69 88 6C B7 7B AE 81 A7 D3
3A 59 2A E7 04 CC CD DD 13 30 9B 79 51 5C AE 82
B1 32 13 39 61 D2 1E 47 3F D4 40 E4 29 49 9A BA
4B 51 5D 7E 81 9E 40 B3 CC CC 06 84 CE A2 85 23
E2 A2 7F 3B 24 F2 72 0B A4 F2 66 83 CC 49 D2 EC
89 E5 EF 0E 37 A3 A1 95 AB C5 C0 5B 8D 7F F6 98
10 C6 B7 CC F7 B9 EE 11 E6 7A 72 FD D5 63 3F 85
3D A0 D9 5B C3 B7 01 FE E9 25 F4 0E A1 5F 09 FB
50 DE 06 82 E7 96 FD 89 9F D9 DB A5 E5 BE EE 8C

B. DIEHARD TEST SUITE RESULTS

Table B.1 Results according to DieHard test suite

Test	Extra Rotation Eklenen	Input
birthday spacings,	1.00	0.00
overlapping permutations,	0.25	0.00
ranks of 31x31 and 32x32 matrices	0.00	0.00

ranks of 6x8 matrices	1.00	0.00
monkey tests on 20-bit Words (bitstream)	0.60	0.00
monkey tests OPSO, OQSO, DNA	0.62	0.00
count the 1's in a stream of bytes	0.50	0.00
count the 1's in specific bytes	0.72	0.00
parking lot	1.00	0.00
minimum distance	1.00	0.00
random spheres	1.00	0.00
Squeeze	1.00	0.00
overlapping sums	0.50	0.00
Runs	0.75	0.00
craps.	1.00	0.00
SUM	10.94	0.00

C. DIEHARD2 TEST SUITE RESULTS

Table C.1 Results according to DieHard2 test suite.

NEW DIEHARD2 TEST DETAILS	p-value
Birthday spacings	0.721339
Tough Birthday spacings	0.547000
GCD	0.732103

Gorilla	0.119000
Overlapping Permutations	0.4208, 0.6911, 0.6770, 0.6521, 0.6798
Ranks of 31×31 and 32×32 Matrices	1
Ranks of 6×8 Matrices	0.909593
Monkey tests on 20-bit words	0.51857165 (Average)
Monkey Test OPSO	0.549708522 (Average)
Monkey Test OQSO	0.454735393 (Average)
Monkey Test DNA	0.603586871 (Average)
Count the 1's in a stream of bytes	0.249728
Count the 1's in specific bytes	0.44275264 (Average)
Parking lot test	0.682237
Minimum distance test	0.682303
Random spheres test	0.870077
The squeeze test	0.217116
Overlapping sums test	0.773786
Runs up and down test	0.673000
The craps test	0.785901
Craps Test With Different Dice	0.945402
Overall KS p-value	0.223461

D. NIST TEST SUITE RESULTS

C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
9	6	17	9	7	0.202268	1.0000	Frequency
9	5	14	13	12	0.366918	0.9900	BlockFrequency
14	9	7	12	9	0.224821	1.0000	CumulativeSums
13	12	8	11	9	0.455937	1.0000	CumulativeSums
12	7	7	8	10	0.437274	1.0000	Runs
15	10	15	5	8	0.075719	0.9800	LongestRun
0	0	0	0	0	0.000000	0.0000	* Rank
15	11	11	8	9	0.514124	0.9900	FFT
13	14	8	13	12	0.289667	0.9900	NonOverlappingTemplate
12	16	15	10	11	0.153763	1.0000	NonOverlappingTemplate
8	9	19	9	7	0.319084	0.9700	NonOverlappingTemplate
12	5	16	11	11	0.514124	0.9800	NonOverlappingTemplate
8	10	12	6	9	0.202268	0.9700	NonOverlappingTemplate
8	7	11	9	7	0.262249	0.9800	NonOverlappingTemplate
7	14	7	5	12	0.474986	0.9700	NonOverlappingTemplate
10	5	7	11	7	0.122325	1.0000	NonOverlappingTemplate
12	5	12	9	11	0.699313	0.9900	NonOverlappingTemplate
13	5	4	9	14	0.224821	1.0000	NonOverlappingTemplate

8	8	6	13	8	0.455937	0.9900	NonOverlappingTemplate
10	10	7	10	8	0.759756	1.0000	NonOverlappingTemplate
10	11	11	10	8	0.964295	0.9900	NonOverlappingTemplate
6	14	11	9	5	0.657933	0.9800	NonOverlappingTemplate
13	3	9	14	9	0.202268	0.9700	NonOverlappingTemplate
8	9	13	7	6	0.514124	0.9900	NonOverlappingTemplate
8	11	8	8	9	0.236810	0.9600	* NonOverlappingTemplate
11	10	14	13	6	0.419021	0.9900	NonOverlappingTemplate
12	9	12	9	7	0.719747	1.0000	NonOverlappingTemplate
9	12	10	13	11	0.935716	0.9800	NonOverlappingTemplate
10	10	16	12	9	0.699313	0.9900	NonOverlappingTemplate
10	6	16	8	6	0.062821	1.0000	NonOverlappingTemplate
8	7	12	13	6	0.437274	0.9600	* NonOverlappingTemplate
10	10	9	6	14	0.494392	1.0000	NonOverlappingTemplate
8	12	15	9	11	0.455937	1.0000	NonOverlappingTemplate

10	12	15	7	13	0.249284	1.0000	NonOverlappingTemplate
10	6	9	12	10	0.867692	0.9900	NonOverlappingTemplate
6	10	8	14	12	0.474986	1.0000	NonOverlappingTemplate
12	9	12	7	8	0.883171	0.9800	NonOverlappingTemplate

12	8	7	13	7	0.816537	1.0000	NonOverlappingTemplate
10	7	10	10	10	0.798139	0.9900	NonOverlappingTemplate
13	9	12	11	5	0.350485	0.9900	NonOverlappingTemplate
8	13	11	9	11	0.401199	0.9900	NonOverlappingTemplate
10	7	8	8	14	0.534146	0.9900	NonOverlappingTemplate
12	13	5	7	12	0.319084	1.0000	NonOverlappingTemplate
13	8	6	11	9	0.678686	0.9900	NonOverlappingTemplate
9	14	11	6	9	0.834308	0.9800	NonOverlappingTemplate
7	12	16	16	6	0.162606	0.9900	NonOverlappingTemplate
12	10	14	11	6	0.779188	0.9900	NonOverlappingTemplate
9	9	14	10	9	0.514124	0.9900	NonOverlappingTemplate
10	8	11	8	6	0.090936	0.9900	NonOverlappingTemplate
7	11	13	7	6	0.595549	0.9900	NonOverlappingTemplate
9	11	9	19	7	0.191687	1.0000	NonOverlappingTemplate
5	8	7	12	11	0.514124	0.9900	NonOverlappingTemplate
10	9	8	12	17	0.319084	0.9800	NonOverlappingTemplate
9	11	12	9	11	0.759756	0.9800	NonOverlappingTemplate
9	11	9	8	16	0.834308	1.0000	NonOverlappingTemplate
11	14	9	9	12	0.657933	0.9800	NonOverlappingTemplate
9	11	12	6	10	0.739918	0.9900	NonOverlappingTemplate
9	7	12	15	6	0.334538	0.9800	NonOverlappingTemplate

6	11	16	13	5	0.066882	0.9700	NonOverlappingTemplate
2	11	12	9	13	0.090936	0.9900	NonOverlappingTemplate
12	7	13	11	12	0.554420	0.9700	NonOverlappingTemplate
20	5	13	5	11	0.019188	1.0000	NonOverlappingTemplate
15	8	13	10	5	0.455937	0.9900	NonOverlappingTemplate
12	7	15	6	8	0.455937	0.9800	NonOverlappingTemplate
13	11	12	11	9	0.514124	1.0000	NonOverlappingTemplate
6	13	10	11	16	0.401199	0.9800	NonOverlappingTemplate
6	7	8	13	10	0.595549	1.0000	NonOverlappingTemplate
6	16	9	8	8	0.102526	0.9900	NonOverlappingTemplate
11	13	12	11	6	0.739918	0.9900	NonOverlappingTemplate
5	13	8	13	5	0.514124	1.0000	NonOverlappingTemplate
9	9	9	10	16	0.867692	0.9800	NonOverlappingTemplate
11	9	13	8	16	0.474986	0.9900	NonOverlappingTemplate
15	10	6	10	10	0.437274	0.9900	NonOverlappingTemplate
6	6	13	13	11	0.637119	1.0000	NonOverlappingTemplate
11	8	5	15	14	0.171867	0.9800	NonOverlappingTemplate
13	6	9	10	10	0.494392	0.9700	NonOverlappingTemplate
10	14	8	5	11	0.514124	1.0000	NonOverlappingTemplate
9	10	10	7	6	0.249284	0.9800	NonOverlappingTemplate

13	10	3	11	11	0.262249	0.9900	NonOverlappingTemplate
6	8	7	14	11	0.191687	0.9700	NonOverlappingTemplate
8	8	10	13	8	0.759756	1.0000	NonOverlappingTemplate
9	4	11	12	13	0.350485	0.9900	NonOverlappingTemplate
13	14	8	13	12	0.304126	0.9900	NonOverlappingTemplate
15	6	8	3	15	0.066882	1.0000	NonOverlappingTemplate
13	8	6	17	12	0.153763	1.0000	NonOverlappingTemplate
6	10	12	8	11	0.834308	1.0000	NonOverlappingTemplate
9	11	15	13	6	0.437274	0.9900	NonOverlappingTemplate
5	5	12	14	11	0.090936	0.9800	NonOverlappingTemplate
9	8	12	9	10	0.455937	0.9900	NonOverlappingTemplate
13	12	6	5	12	0.595549	1.0000	NonOverlappingTemplate
6	9	16	15	6	0.102526	0.9900	NonOverlappingTemplate
10	7	11	14	13	0.595549	0.9900	NonOverlappingTemplate
6	10	8	9	12	0.739918	0.9900	NonOverlappingTemplate
7	7	8	8	13	0.867692	0.9900	NonOverlappingTemplate
11	15	5	17	6	0.000474	0.9900	NonOverlappingTemplate
8	18	10	7	7	0.236810	0.9900	NonOverlappingTemplate
9	11	10	12	10	0.834308	1.0000	NonOverlappingTemplate
6	11	17	7	8	0.437274	1.0000	NonOverlappingTemplate
9	9	7	8	14	0.595549	0.9800	NonOverlappingTemplate

11	10	12	7	8	0.383827	1.0000	NonOverlappingTemplate
9	6	11	5	8	0.455937	0.9700	NonOverlappingTemplate
11	7	8	9	7	0.366918	0.9800	NonOverlappingTemplate
7	16	11	11	14	0.145326	1.0000	NonOverlappingTemplate
10	9	13	9	7	0.798139	1.0000	NonOverlappingTemplate
11	9	9	12	11	0.455937	0.9900	NonOverlappingTemplate
10	9	9	6	6	0.058984	1.0000	NonOverlappingTemplate
9	9	11	10	14	0.678686	0.9700	NonOverlappingTemplate
8	15	8	12	6	0.289667	0.9700	NonOverlappingTemplate
12	9	12	6	10	0.474986	0.9900	NonOverlappingTemplate
6	7	7	7	16	0.085587	0.9800	NonOverlappingTemplate
11	6	8	15	17	0.162606	1.0000	NonOverlappingTemplate
9	9	8	11	9	0.935716	0.9900	NonOverlappingTemplate
11	8	13	13	16	0.350485	0.9800	NonOverlappingTemplate
8	13	12	9	3	0.455937	1.0000	NonOverlappingTemplate
13	7	9	5	13	0.145326	0.9900	NonOverlappingTemplate
12	13	10	12	9	0.699313	0.9900	NonOverlappingTemplate
4	7	12	11	12	0.080519	0.9800	NonOverlappingTemplate
8	12	12	13	9	0.401199	1.0000	NonOverlappingTemplate
9	11	15	6	6	0.657933	1.0000	NonOverlappingTemplate
11	6	12	12	11	0.867692	0.9800	NonOverlappingTemplate

5	9	11	9	7	0.289667	0.9900	NonOverlappingTemplate
7	5	13	13	10	0.657933	0.9800	NonOverlappingTemplate
7	8	5	17	9	0.080519	1.0000	NonOverlappingTemplate

11	12	6	6	14	0.162606	0.9900	NonOverlappingTemplate
15	16	8	10	6	0.350485	1.0000	NonOverlappingTemplate
16	17	8	9	6	0.137282	0.9800	NonOverlappingTemplate
11	7	8	10	12	0.883171	0.9900	NonOverlappingTemplate
7	7	10	13	6	0.401199	0.9900	NonOverlappingTemplate
7	13	8	13	9	0.249284	0.9900	NonOverlappingTemplate
8	11	11	9	15	0.678686	1.0000	NonOverlappingTemplate
17	12	10	7	9	0.514124	1.0000	NonOverlappingTemplate
7	10	13	15	5	0.350485	0.9900	NonOverlappingTemplate
13	17	8	8	12	0.262249	1.0000	NonOverlappingTemplate
13	11	8	10	10	0.883171	1.0000	NonOverlappingTemplate
7	9	15	9	9	0.834308	1.0000	NonOverlappingTemplate
9	12	13	11	8	0.897763	1.0000	NonOverlappingTemplate
16	7	10	8	15	0.249284	0.9900	NonOverlappingTemplate
11	10	12	6	10	0.759756	0.9700	NonOverlappingTemplate
9	8	16	8	4	0.275709	0.9800	NonOverlappingTemplate
8	17	9	12	11	0.474986	0.9800	NonOverlappingTemplate

13	10	12	8	12	0.616305	0.9800	NonOverlappingTemplate
8	9	8	9	9	0.719747	0.9900	NonOverlappingTemplate
8	11	12	14	6	0.275709	1.0000	NonOverlappingTemplate
12	7	14	9	5	0.514124	0.9900	NonOverlappingTemplate
13	11	6	16	9	0.401199	0.9700	NonOverlappingTemplate
10	8	8	14	13	0.883171	0.9700	NonOverlappingTemplate
8	10	11	13	13	0.719747	0.9900	NonOverlappingTemplate
9	14	12	10	9	0.699313	1.0000	NonOverlappingTemplate
8	8	8	11	9	0.514124	0.9700	NonOverlappingTemplate
12	14	13	6	12	0.616305	0.9800	NonOverlappingTemplate
8	9	6	12	9	0.897763	1.0000	NonOverlappingTemplate
6	12	11	10	15	0.779188	1.0000	NonOverlappingTemplate
11	5	13	9	15	0.616305	0.9900	NonOverlappingTemplate
11	8	13	8	10	0.816537	0.9900	NonOverlappingTemplate
9	10	8	5	11	0.816537	1.0000	NonOverlappingTemplate
9	4	11	12	13	0.249284	0.9900	NonOverlappingTemplate
3	16	7	10	6	0.085587	1.0000	OverlappingTemplate
0	0	0	0	0	0.000000*	1.0000	Universal
9	15	7	7	12	0.616305	0.9800	ApproximateEntropy
0	1	1	2	1	0.739918	1.0000	RandomExcursions
0	1	5	1	1	0.035174	1.0000	RandomExcursions

3	0	0	1	1	0.122325	1.0000	RandomExcursions
2	1	3	1	0	0.534146	1.0000	RandomExcursions
0	3	0	1	1	0.213309	1.0000	RandomExcursions
0	3	1	1	1	0.534146	0.9167	RandomExcursions
1	1	1	1	2	0.991468	1.0000	RandomExcursions
0	5	1	1	3	0.002043	1.0000	RandomExcursions
3	2	1	0	0	0.213309	1.0000	RandomExcursionsVariant

2	2	0	2	1	0.739918	1.0000	RandomExcursionsVariant
2	1	0	2	3	0.350485	1.0000	RandomExcursionsVariant
3	1	1	2	2	0.534146	1.0000	RandomExcursionsVariant
0	2	0	3	3	0.122325	1.0000	RandomExcursionsVariant
1	0	1	1	3	0.534146	1.0000	RandomExcursionsVariant
1	0	2	1	2	0.534146	1.0000	RandomExcursionsVariant
2	1	2	1	2	0.911413	1.0000	RandomExcursionsVariant
2	2	3	1	1	0.534146	1.0000	RandomExcursionsVariant
1	4	1	0	2	0.122325	1.0000	RandomExcursionsVariant
1	1	2	3	1	0.739918	1.0000	RandomExcursionsVariant
1	2	3	1	0	0.534146	1.0000	RandomExcursionsVariant
1	0	2	1	1	0.534146	1.0000	RandomExcursionsVariant
3	1	1	0	2	0.534146	1.0000	RandomExcursionsVariant

1	0	2	2	0	0.213309	1.0000	RandomExcursionsVariant
3	0	1	2	0	0.213309	1.0000	RandomExcursionsVariant
1	1	1	0	2	0.739918	1.0000	RandomExcursionsVariant
0	1	1	1	1	0.739918	1.0000	RandomExcursionsVariant
13	7	8	8	10	0.834308	1.0000	Serial
8	12	7	10	9	0.779188	0.9900	Serial
7	4	6	13	9	0.122325	0.9900	LinearComplexity