A Lightweight Authenticated Symmetric Encryption Cipher for RFID Systems

A Dissertation

Presented to the

Graduate Faculty of the

University of Louisiana at Lafayette

In Partial Fulfillment of the

Requirements for the Degree

Doctor of Philosophy

Zahra Jeddi

Summer 2014

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

A Lightweight Authenticated Symmetric Encryption Cipher for RFID Systems

Zahra Jeddi

APPROVED:

Magdy Bayoumi, Chair Professor of Computer Engineering The Center for Advanced Computer Studies	Hong-yi Wu Professor of Computer Science Director of the Center for Advanced Computer Studies
Ashok Kumar Assistant Professor of Computer Science The Center for Advanced Computer Studies	Mohammad R. Madani Associate Professor of Electrical Engineering

Mary Farmer-Kaiser
Interim Dean of the Graduate School

DEDICATION

To my parents and my husband.

ACKNOWLEDGMENTS

This work would not have been possible without the support and assistance of many people. Here, the author wishes to thank all who provided this possibility to complete this dissertation.

First, I would like to express my deepest gratitude to my advisor, Dr. Magdy Bayoumi, for his outstanding supervision and constant support during the past years. His useful comments and guidance helped me to find my way in the path of this research.

Next, I would like to acknowledge my dissertation committee members, Dr. Hong-yi Wu, Dr. Ashok Kumar and Dr. Mohammad Madani for accepting to be on my advisory committee as well as giving me many valuable suggestions and comments that have enriched and improved my presentation and also my dissertation significantly.

I would also like to express my appreciation to Mrs. Masoumeh Abouroshi because I may have never reached to this place without her help.

Last but not least, I am deeply thankful to my family for their persistent care, trust and support. I am deeply obliged to my father who taught me to work hard for earning my success and my mother who gave me confidence with her endless kindness and inspiration. I also would like to express my special gratitude and thanks toward my husband, Esmaeil Amini, for his kind cooperation and encouragement that helped me in completion of this project.

TABLE OF CONTENTS

DEDICATION iv

ACKNOWLEDGMENTS v

LIST OF TABLES ix

LIST OF FIGURES xi

CHAPTER 1: Introduction and Motivation 1
1.1 Introduction 1
1.1.1 Automatic Identification 1
1.1.2 RFID Applications 5
1.1.3 How Passive RFID Tags Work 12
1.2 Risks and Threats in RFID Systems 14
1.2.1 Eavesdropping 14
1.2.2 Tracing and Tracking 15
1.2.3 Snooping 15
1.2.4 Skimming 16
1.2.5 Tag Modification 16
1.2.6 Physical Attacks 16
1.3 Physical Solutions 17
1.3.1 Killing Tags 17
1.3.2 Sleeping Tags 18
1.3.3 Faraday Cage 18
1.3.4 Blocker Tags 19
1.4 Authentication 22
1.5 Motivations 24
1.6 Main Contributions 25
1.7 Thesis Organization 27

CHAPTER 2: Related Works 29
2.1 Cryptography 29
2.2 Lightweight Cryptography 30
2.3 Asymmetric Key Encryption 32
2.3.1 Elliptical Curve Cryptography (ECC) 34
2.4 Symmetric Key Encryption 35

2.4.1 Block Ciphers 37
2.4.2 Advanced Encryption Standard (AES) 39
2.4.3 PRESENT 41
2.4.4 Stream Ciphers 44
2.4.5 Keystream 45
2.4.6 Trivium 48
2.4.7 Grain 50
2.4.8 Hybrid Ciphers 51
2.4.9 HB (Hummingbird) 53
2.5 Conclusion 55

CHAPTER 3: RBS Cryptosystem 57
3.1 Key and Number of Redundant Bits 58
3.1.1 Key Space 59
3.1.2 Flexibility in Security Level 65
3.2 Location of Redundant Bits 66
3.3 Value of Redundant Bits 66
3.3.1 Message Authentication and Data Integrity 68
3.3.2 Message Authentication and Redundant Bits 70
3.4 Plaintext Manipulation 71
3.4.1 Direct Appearance inside the Ciphertext 71
3.4.2 Bitwise Addition with a Constant-Value Keystream 72
3.4.3 Bitwise Addition with Variable-Value Keystream 73
3.5 Implementation 75
3.5.1 MAC Generator 75
3.5.2 Chosen MAC Algorithm for RBS 77
3.5.3 Adapting the Chosen MAC with RBS 81
3.5.4 Encryption 84
3.5.5 Decryption 85
3.5.6 Reception / Transmission 86
3.6 Overall System 88
3.7 Conclusion 89

CHAPTER 4: Security Analysis 92
4.1 Security Model 92
4.2 Mathematic Background 93
4.3 RBS Security 96

4.3.1 Brute Force Attack 96
4.3.2 Known-Plaintext Attack 97
4.3.3 Chosen-Plaintext Attack 98
4.3.4 Chosen-Ciphertext Attack 99
4.3.5 Differential Attack 100
4.3.6 Substitution Attack 103
4.3.7 Related Key Attack 103
4.3.8 Linear Cryptanalysis 106
4.3.9 Algebraic Attack 107
4.3.10 Cube Attack 108
4.3.11 Side Channel Attack 109
4.4 Conclusion 111

CHAPTER 5: Experimental Results 113
5.1 ASIC Implementation of RBS 113
5.2 Comparison Ciphers 116
5.2.1 Area 121
5.2.2 Performance 125
5.2.3 Area-Time Product 133
5.2.4 Hardware Efficiency 134
5.2.5 Power 136
5.2.6 Energy 138
5.2.7 Energy-per-Bit 139
5.2.8 Trade-offs 141
5.2.9 Power-Area-Time Product 143
5.3 Conclusion 145

CHAPTER 6: Conclusions and Future Work 151
6.1 Conclusions 151
6.2 Future Work 153

BIBLIOGRAPHY 155

ABSTRACT 164

BIOGRAPHICAL SKETCH 165

LIST OF TABLES

Table 1-1 Comparison of auto-ID solutions [1] 5

Table 1-2 Tag frequencies and reading distances 13

Table 2-1 Hardware implementation results for ECC 35

Table 2-2 Results of implementation of AES 41

Table 2-3 The PRESENT S-Box [24] 42

Table 2-4 Hardware implementation result for PRESENT at 100 KHz frequency [13] 43

Table 2-5 Implementation results for Trivium 50

Table 2-6 Implementation result for Grain cipher with different key sizes [48] 51

Table 2-7 S-Boxes used in Hummingbird-2 [62] 54

Table 2-8 Hardware implementations of Hummingbird-2 [62] 55

Table 3-1 The number of bits required in ciphertext to have $s=2^{128}$ 63

Table 3-2 Number of required redundant bits for different security level 64

Table 3-3 Summary of MAC algorithms 76

Table 4-1 Time required for breaking key by Brute Force attack 97

Table 4-2 Simulation of RBS outputs when the inputs are different in one bit 102

Table 5-1 Area report for each part of RBS design [GE] 114

Table 5-2 Static power consumption for each part of RBS design [µW] 114

Table 5-3 Dynamic power consumption for each part of RBS design [µW] 115

Table 5-4 Total area and power consumption overhead for RBS designs 115

Table 5-5 The number of clock cycles required for generating the output in RBS 116

Table 5-6 Comparing RBS with other encryption methods 117

Table 5-7 Hardware implementation of MAC 119

Table 5-8 The performance of different hash functions based on PRESENT 120

Table 5-9 Comparison of clock cycles to encrypt 126

Table 5-10 Number of required cycles for encryption 64-bit plaintext plus authentication..... 127

Table 5-11 Bits-per-clock without authentication..... 129

Table 5-12 Bits-per-clock with authentication..... 129

Table 5-13 Maximum clock frequency..... 130

Table 5-14 Maximum throughput..... 132

Table 5-15 Energy required for encryption of a 64-bit plaintext without authentication..... 139

Table 5-16 Energy required for encryption and authentication of a 64-bit plaintext..... 139

Table 5-17 Summary of normalized metrics without authentication..... 146

Table 5-18 Summary of normalized metrics with authentication..... 147

Table 5-19 The size of ciphertext for different input sizes when authentication is provided..... 149

LIST OF FIGURES

Figure 1-1 Automatic identification solutions 3

Figure 1-2 Global RFID market value in 2016 [2] 7

Figure 1-3 RFID system architecture 8

Figure 1-4 Different types of tags 10

Figure 1-5 A typical passive tag 12

Figure 1-6 Inductive coupling 13

Figure 1-7 Eavesdropping attack adapted from [7] 15

Figure 1-8 A faraday cage in an electric field 19

Figure 1-9 Blocker tags blocks reading by broadcasting signals for every reader's query 20

Figure 1-10 Challenge-response technique in symmetric authentication [11] 23

Figure 2-1 Design trade-offs for lightweight cryptography [13] 31

Figure 2-2 Asymmetric key encryption 33

Figure 2-3 Symmetric key encryption 36

Figure 2-4 Block cipher operations on fixed size of blocks 37

Figure 2-5 Four steps in AES [28] 40

Figure 2-6 A top-level algorithmic description of PRESENT algorithm [24] 42

Figure 2-7 Three layers at one round in PRESENT cipher [24] 43

Figure 2-8 One-time pad cipher 44

Figure 2-9 Keystream generator scheme 46

Figure 2-10 Stream cipher operation 47

Figure 2-11 Typical LFSR 48

Figure 2-12 Hardware implementation of Trivuim [47] 49

Figure 2-13 Grain cipher [54] 50

Figure 2-14 Comparing properties of block ciphers and stream ciphers 53

Figure 3-1 Changing the size of key space with the number of redundant bits 61
Figure 3-2 Size of key space when the number of redundant bits is equal with plaintext bits 61
Figure 3-3 Key space growth while plaintext size is fixed (64 bits) 62
Figure 3-4 Redundant data generation algorithm 67
Figure 3-5 MAC algorithm 68
Figure 3-6 Embedding tag inside the ciphertext in different protocols 70
Figure 3-7 Block diagram of encryption and decryption 74
Figure 3-8 The hardware suggested for MAC generation in [56] 78
Figure 3-9 Authentication algorithm 79
Figure 3-10 The bias as it develops for growing sequence lengths adapted from [56] 80
Figure 3-11 Adapted MAC generator for RBS 83
Figure 3-12 Encryption module in transmission part 85
Figure 3-13 Extracting altered plaintext and redundant data from ciphertext 85
Figure 3-14 Cipher plus transmitter and receiver 87
Figure 3-15 Transmission and reception algorithms 88
Figure 3-16 The flowchart of the RBS algorithm of the overall system 91
Figure 4-1 Differential attack 101
Figure 4-2 Error correcting of secret key 106
Figure 4-3 The cryptographic model including side channel attack [78] 109
Figure 4-4 Adding redundant MAC generator to RBS cipher 111
Figure 4-5 RBS cipher with radix-2 111
Figure 5-1 Comparing area for different design in ECC 122
Figure 5-2 Area comparison of symmetric ciphers without providing authentication 123
Figure 5-3 Area comparison of different ciphers with providing authentication 124
Figure 5-4 Throughput when the operating frequency is 10MHz without authentication 131
Figure 5-5 Throughput when the operating frequency is 10MHz with authentication 132

Figure 5-6 Area-Time product when authentication is not provided 133
Figure 5-7 Area-Time product when authentication is provided..... 134
Figure 5-8 Hardware efficiency when frequency is 10MHz and without authentication..... 135
Figure 5-9 Hardware efficiency when frequency is 10MHz and authentication is provided 135
Figure 5-10 Power consumption without authentication..... 137
Figure 5-11 Power consumption for 64-bit plaintext when authentication is provided 138
Figure 5-12 Energy-per-bit without authentication 140
Figure 5-13 Energy-per-bit with authentication 141
Figure 5-14 Energy-per-bit vs. hardware efficiency without authentication..... 142
Figure 5-15 Energy-per-bit vs. hardware efficiency with authentication..... 143
Figure 5-16 Power-Area-Time product when frequency is 10MHz without authentication 144
Figure 5-17 Power-Area-Time product when frequency is 10MHz with authentication 145
Figure 5-18 Size of output for different sizes of plaintext 150

CHAPTER 1: Introduction and Motivation

1.1 Introduction

In the last decade, the desire and need to develop new technologies which support automatic identification procedures for objects and items, has grown up rapidly. This technology offers enormous productivity benefits such as saving time, reducing error and providing abilities like detecting and tracking. Many modern enterprises and big organization such as Wal-Mart and the United States Department of Defense have made great efforts to improve and apply automated oversight in many applications involved with supporting items tracking, logistics management, supply chain management and control access.

Radio-Frequency IDentification or RFID is one of automatic identification techniques which identify objects remotely through a radio frequency channel. In fact, RFID is not a very new technology. During World War II, British military proposed it to recognize friendly aircrafts. Nowadays, thanks to a combination of dropping cost and technology advancement, RFID can be applied in a variety of applications and in new ways.

Despite attention gained by RFID systems, privacy issues for users such as clandestine physical tracking of objects and inventorying of them are becoming a big concern. Enormous research effort has been done in order to solve this problem. However, most methods request heavy or frequent cryptographic operations on RFID tags, which contradict the low cost demand of RFID tags.

1.1.1 Automatic Identification

The Automatic Identification or Auto-ID system is a broad term refers to any technology that can identify and locate physical objects automatically by electronically

exchanging data and without any human interaction. The goal of using Auto-ID systems is to increase efficiency and decrease cost by reducing required human labor at entering data and consequently decreasing number of happening errors.

Due to high reliability provided by Auto-ID systems, utilizing them are getting widespread in applications that require tracking items like supply chain and manufacturing process from the point of producing the products until the point of sale and service them.

There are various Automatic Identification solutions used in industry like Barcodes, chip cards or smart cards, Optical Character Recognition (OCR), voice recognition, Biometric (e.g. print screen) and Radio Frequency Identification (RFID) [1].

Selecting the best Auto-ID solution among all introduced solutions for particular applications depends on the requirements of the application and also the benefits of the chosen solution. Here, each solution will be introduced individually and its strength and weakness is compared with other solutions.

Barcodes are the most common Auto-ID solution in the industry due to their very low cost. A barcode is a small printed image of bars and spaces, attached to items. It is indicating a binary code which identifies the item. To read the data, it is required that the image to be exposed to a scanner. Printing barcodes is easy and cheap which makes the cost of their produce low. Despite the advantages of barcodes like simplicity, universality and low cost, they need a direct contact with scanner to be read which makes speed of reading items low. Also, their readability might be vanished in harsh environment like by dirt or moisture.

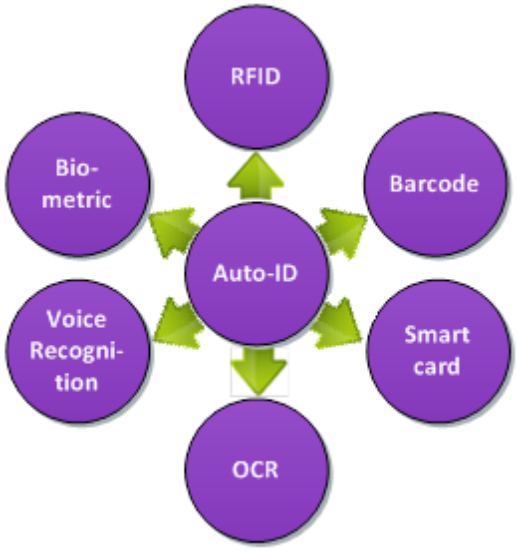


Figure 1-1 Automatic identification solutions

Smart cards are cards with embedded IC which is helpful to provide identification, authentication, data storage and process. Reading data of a smart card is performed through its contact area which makes an electrical connectivity between a reader and the card when the card is inserted into the reader. Smart cards do not have any integrated battery and their required power for communication is provided by the reader. These cards prevent unauthorized reading. However they are vulnerable to harsh environment and they can get affected by dirt. One of the other disadvantages of this solution is cost of maintaining the readers which is very high [1].

In Optical Character Recognition (OCR) any scanned image of text like handwritten or printed text is converted into digital text and processed. The main advantage of this solution lies in handling high density of information. The most important problem in OCR is the cost of readers which is high due to their complexity [1].

In voice recognition, the voice of speaking person is converted into digital data. To recognize the object, this information is compared with the reference patterns recorded

before from all objects. This solution works for just human and utilizing it is not applicable for other items like identifying products [1]. The other disadvantage of this solution is the possibility of forgery by using taped voice.

Biometrics are a type of solution that people are identified by their individual physical attributes like DNA, finger print, palm image and facial image. Voice recognition is a subcategory of biometric solution with this difference that voice recognition consists of audio data while other characteristics are image data. In biometrics solution, direct connection for verifying the identity is required. Also like voice recognition, this solution is applicable just for human.

Radio Frequency Identification solution is closely related to smart cards with this difference that RFIDs can connect to a reader wirelessly when the electromagnetic field is provided by the reader. In this solution, identification is performed using radio signals. Thus, RFID systems do not need physical contact between reader and the card. This way, huge number of items can be identified in a short time with high reliability and low cost which makes this method very attractive for applications like supply chain management, e-health, monitoring objects, electrical tagging, etc. RFID tags can be read in a wide variety of circumstances, where barcodes or other optically read technologies are useless. However, this technology with its all benefits is still costly.

Table 1-1 gives a comparison among different Auto-ID solutions based on different terms. Among all solutions, RFID system gives the better offer compared to other candidates.

Table 1-1 Comparison of auto-ID solutions [1]

	Barcode	OCR	VR	Biometrics	RFID
Data size (Byte)	1-100	1-100	N/A	N/A	16-64 K
Data density	Low	Low	High	High	Very high
Readability by machine	Good	Good	Complex	Complex	Good
Readability by people	Partially	Easy	Easy	Difficult	Impossible
Affected by dirt/ moisture	Strongly	Strongly	N/A	N/A	No influence
Effect of sight distraction	Usage impossible	Usage impossible	N/A	N/A	No influence
Initial costs	Very low	Medium	Very high	Very high	Medium
Unauthorized coping	Easy	Easy	Possible (Tape)	Impossible	Impossible
Reading speed	Slow	Slow	Very slow	Very slow	Fast
Max distance reader/carrier	0.5 cm	Under 1 cm	0.5 cm	Direct contact	0.5 m

1.1.2 RFID Applications

RFID tags bring huge benefits over many systems since they have this ability to be read if they pass near a reader even if it is covered by objects or not visible like when it is in a container or a box. Also, hundreds of tags can be read at a time. These advantages offer new solutions to variety of applications, such as:

- Asset tracking: the location of tagged assets like healthcare facilities or a laptop can be instantly determined anywhere within the help of RFID technology. This application is also very useful in some services like postal services, and monitoring vehicle traffic.
- Animal tracking: this application keeps the track of livestock to help prevent disease outbreaks. It also can be used by pet owners to keep track of their animals when they're lost.

- People tracking: this application is required in hospitals and jails. In a hospital, this technology can help to track special patients who need special or mental care and also for new born babies.
- Supply chain management: By attaching RFID tags to each product, tool, resource and item, manufacturers will be able to get better demand signals from customers and the also market. RFID simply offers the potential to improve product lifecycle management, and quality control with the aim of helping retailers to provide the right product at the right place at the right time and consequently to maximize sales and profits.
- Person identification and access management: this application allows identifying people by injected RFID chips under the skin for use in a variety of settings, including financial and transportation security, military and government security to control accesses to secure areas like residential and commercial buildings.
- Payment applications: these applications are widely organized and began to receive attention as a convenient way for payment like Toll collection.
- Applications for tomorrow: there are some applications which are not applicable now but it might be possible in future such as smart appliances. For example a smart oven which knows how to cook pre-packaged food.

Analyzing the RFID market in many different ways, technical experts expect that retail dominates the market in future (Figure 1-2) [2]. Thus, retail companies are required to move towards RFID system to avoid losing their profits. The supplier of other sectors in this pie will receive the benefits of RFID by providing a secure and enduring support for their

customers, considering anti-counterfeiting RFID for drugs, error-preventing RFID on hospital instruments and anti-terrorism measures in airports.

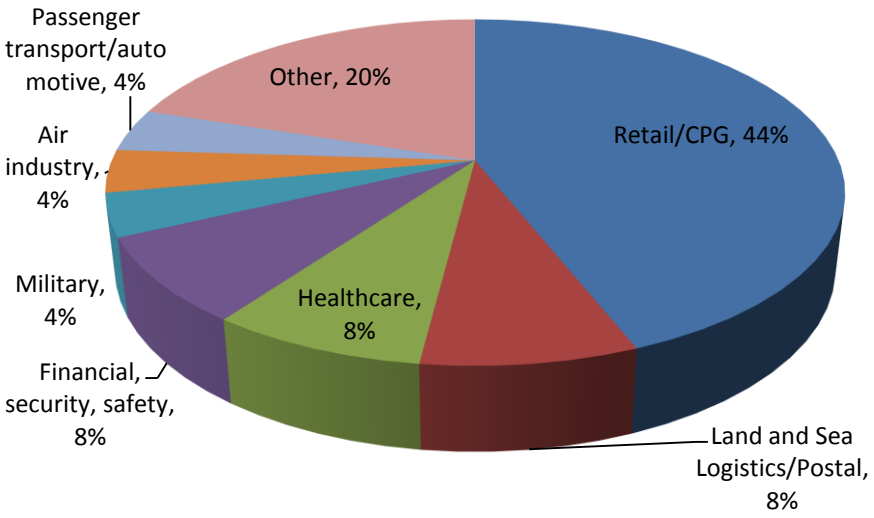


Figure 1-2 Global RFID market value in 2016 [2]

In general, each RFID system consists of three parts: transponder or tag, transceiver or reader, and a back-end server (Figure 1-3). An RFID Tag is composed of an electronic circuit with unique identifiers and one antenna, used for communication. The tag is attached to or embedded in an object to provide unique identification for it. It contains some information associated with the corresponding object. This information can be either as short as few bits or be a collection of data such as identity code for animals, expiration date for groceries and personal medical information for people.

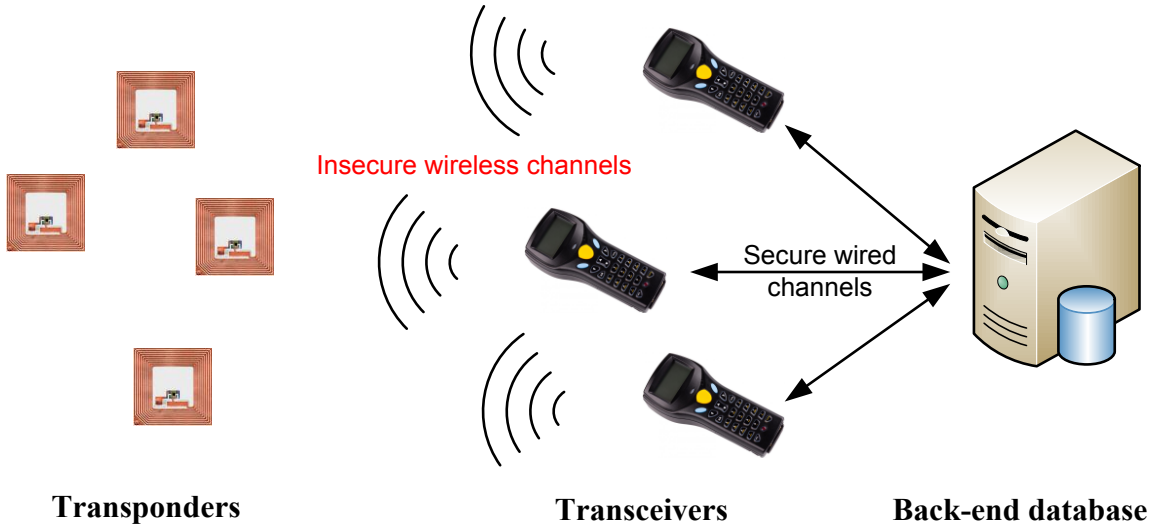


Figure 1-3 RFID system architecture

A Transceiver or reader is a two-way radio transmitter-receiver that both receives and transmits radio waves unlike tags which transmit signals only in response to received signals. The reader has a powerful antenna and a power supply, surrounding itself with an electromagnetic field in order to activate tags and read their data through radio frequency waves.

The collected data from tags by reader is sent to the back-end server. This server contains data base of tags' information. The received data are stored and processed here.

The channels between the reader and the back-end database are wired links that are usually assumed to be secure. On the other hand, both reader and back-end server are powerful enough to apply strong cryptographic protocols. On the contrary, the channels between the tags and the reader are wireless channels. The wireless communication is in danger of eavesdropping by adversaries which make it vulnerable to a variety of attacks. Handling contemporary cryptographic protocols in RFID tags is not possible since they

usually have restricted capabilities in every aspect of computation, communication and storage because of their extremely low production cost.

RFID tags are classified according to their embedded power supply. Generally, there are three types of tags: active, semi-passive and passive tags.

In active tags, a radio signal transceiver is embedded along with a power source, usually in the form of a small battery to power it (Figure 1-4.a). Because of the on-board battery, active RFID tags can initiate communication and activate themselves regardless of the presence of a reader in their vicinity. However to conserve the battery, active tags usually remain in a low power state until they detect the presence of an RF field being sent by a Reader. Whenever the tag leaves this area it comes back to low power state again.

Thanks to the equipped battery, active tags can cover longer ranges compared to other type of tags. So these tags can be read by the reader while they are much farther away. However, their life time is restricted by the capacity of their battery. Even though some of them are built to have up to few years life span, they still have limited life time. Due to these characteristics, active tags are usually utilized in real time location systems to measure environmental parameters like humidity, temperature and pressure. Compared to other types of tags, active tags are more expensive and have more limitations because of the existence of the battery.

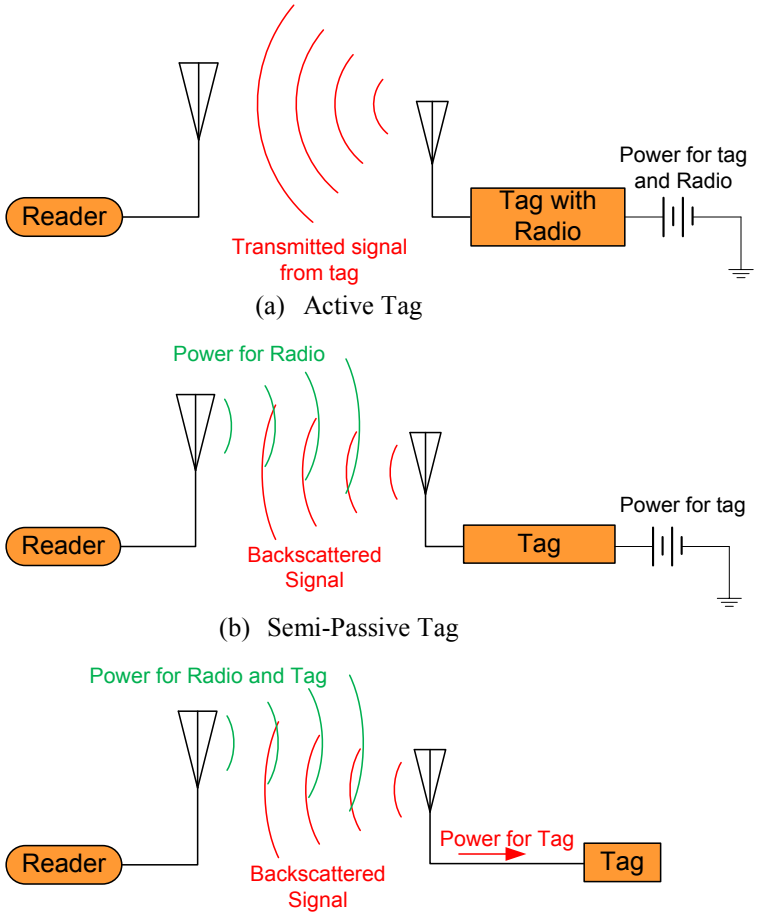


Figure 1-4 Different types of tags

Semi-passive tags have their own power supply that supports the integrated microchip only. When the battery is discharged, these tags cannot transmit signals any more. Unlike active tags semi-passive tags have no active transmitter and to communicate with the reader they use the backscatter technique (Figure 1-4.b). In this technique, radio frequency energy transferred from the reader are gathered and altered to transmit data in a way that the reader can detect. Therefore they cannot initiate communication.

Passive Tags have no internal power source. They draw their power from the electromagnetic field generated by the RFID reader (Figure 1-4.c). They have also no active

transmitter and rely on all power comes from a reader's signal. Passive tags are inactive unless a reader activates them. Compared to other types of tags, passive tags are cheaper and smaller while the covered range is short. Since passive tags do not require having any battery to support their computation and communication, they can stay usable for very long periods of time. Due to these features that make them suitable for a wide range of applications, passive tags are the most common type of tags in the market. Moreover, passive tags can tolerate environmental conditions while these conditions limit the use of tags with on-board batteries. However, in passive tags, the power required for computation and communication is limited by the obtained power from the field. Some solutions have been given to increase the obtained power in tags. One of them is increasing antenna gain of tags which helps to gather more energy from the field. Because of having the limitation on the size of the tag, this solution is impractical. Increasing the power of the field is another solution. However, the maximum strength of sent signals by readers is limited by law. Due to the nature of RFID tags, designers confront many technical limitations which have to deal with them such as:

- Limited power consumption
- Limited area
- Limited execution time
- Limited backward channel
- Limited memory access

Figure 1-5 A typical passive tag

1.1.3 How Passive RFID Tags Work

The communication between a passive tag and a reader performs through transferring energy and data. Energy, provided by the reader is transferred to the tag using coupling via electromagnetic fields [3]. To receive energy, RFID tags can use both the electric field and the magnetic field or one of them. Passive RFID tags do not have any energy for communication until they enter one of these fields. As soon as tags pass through the field they are able to draw enough power from the field to become activated.

Based on the provided field, there are different methods for transferring data from the tag to the reader. One of the contemporary techniques is backscattering which is described before. In this method, the reader transmits a continuous wave of radio frequency signal into the environment. When a tag enters in this area, it receives the reader signal and demodulates it. The transmitted wave consists of commands to inform the tag what operations to perform. In reply, the tag modulates its response and sends it back to the reader.

Inductive coupling is another common method for transferring energy to passive tags (Figure 1-6). This method is based on this fact that when a conductor appears in a magnetic

field, the magnetic field produces a current flow in the conductor [4]. In this method, the antenna of the reader provides the magnetic field and the tag plays as a conductor. When the tag enters the magnetic field, its antenna generates a current into the tag to power it up. Magnetic fields are utilized in low frequency (LF) and high frequency (HF) RFID devices while the distance between the tag and the reader is short.

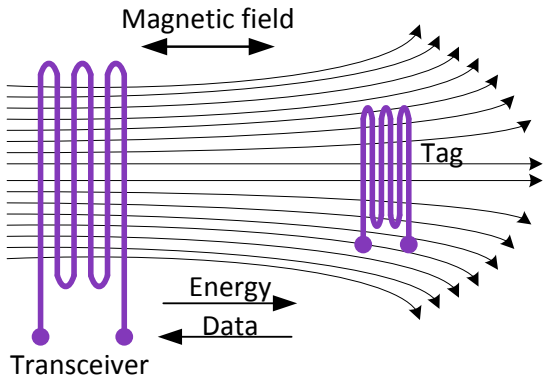


Figure 1-6 Inductive coupling

Electromagnetic coupling method is similar to inductive coupling method with this difference that instead of magnetic field, electromagnetic field is utilized which covers a longer distance for transferring energy to tags. Ultra high frequency (UHF) and microwave tags use this method. Table 1-2 summarizes the using methods of energy transferring in RFID tags based on their operating frequency.

Table 1-2 Tag frequencies and reading distances

Frequency band	Frequency	Distance	Energy Transfer
Low Frequency (LF)	125 KHz	1 - 90 cm, typically around 45 cm	Inductive Coupling
High Frequency (HF)	13.56 MHz	1 - 75 cm, typically around 40 cm	Inductive Coupling
Ultra High Frequency (UHF)	865 - 868 MHz 902 - 928 MHz 433 MHz	Up to 9 m	Electromagnetic Coupling
Microwave (μW)	2.45 GHz 5.8 GHz	Typically 0.3 - 0.9 m	Electromagnetic Coupling

1.2 Risks and Threats in RFID Systems

Like many other technologies, RFID systems might confront with new challenges in providing security and privacy for individuals or organizations against possible threats while they are accomplishing a great productivity gains. Since the communication between the tags and the reader performs through an unsecure wireless channel, the transmitted data is vulnerable to attacks by unauthorized readers.

Attacks can be categorized into two main groups: privacy violation and security violation. In privacy violation, the attacker tries to harvest information from the objects by eavesdropping to the communications between the object and the reader or by tracking them. In security violation, an adversary counterfeit behaviors of a tag or a reader for making undesirable communications. Attacks may happen in the physical layer, the network-transport layer, and the application layer as well as multilayer attacks which affect more than one layer [5]. In this section, some of these security risks and threats are introduced.

1.2.1 Eavesdropping

This threat addresses one of the main privacy concerns over the use of RFID technology. Eavesdropping happens when the channel is overheard secretly by an attacker to retrieve information from it [6]. Since RFID systems working in UHF covers more reading distance than other frequency bands, this threat is more likely to happen in it. Eavesdropping is a feasible threat and hard to be detected since it can be carried out at longer range on the communications between a tag and a valid reader while the adversary is passive and do not send any signal out (Figure 1-7). This threat becomes serious when sensitive information is exchanged on the channel like data of a credit card without any encryption to protect them.

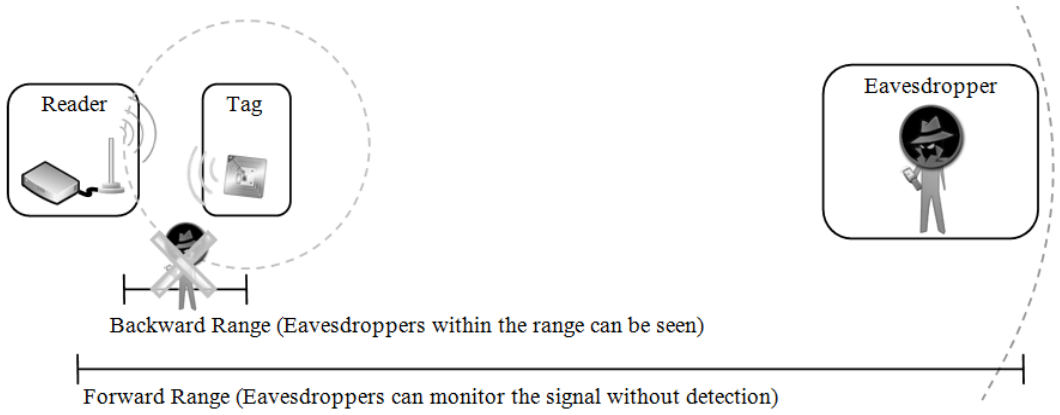


Figure 1-7 Eavesdropping attack adapted from [7]

1.2.2 Tracing and Tracking

These threats violate the concept of location privacy. Illegal tracing and tracking occurs because RFID tags design requires the tag to always respond to the reader's query [6]. By sending queries and obtaining the same response from a tag at various locations it can be determined that where the specific tag is currently and which locations it has visited. Since each RFID tag is affixed to a particular physical item with a unique ID number, this infers that the tag has visited those locations is which object. Encrypting the response by the can prevent having unauthorized access, since the adversary cannot obtain the tag contents without the secret key. However, since the tag always returns a constant response to the queries, the adversary can use this fact to perform illicit tracing and tracking.

1.2.3 Snooping

This attack is defined as illegal reading of a device's identity and data. Snooping is similar to eavesdropping with this difference that in eavesdropping, attacker collect information exchanged between a legitimate tag and legitimate reader. While snooping occurs when the data stored on the RFID tag is read without the owner's knowledge or

agreement by an unauthorized reader interacting the tag. This attack happens because most of the tags transmit their stored data in their memory without requesting any kind of authentication.

1.2.4 Skimming

In this attack, the adversary observes the information exchanged between a legitimate tag and legitimate reader. Via the extracted data, the attacker attempts to make a counterfeit tag which imitates the original RFID tag. To perform this attack, the attacker does not need to have any physical access to the real tag. Skimming attack is precarious when documents like drivers' licenses or passports are authenticated through RFID system. In these situations, attackers observe the interactions between the RFID tag embedded in the document with the reader to make his fake document.

1.2.5 Tag Modification

Since most RFID tags use writable memory, an adversary can take advantage of this feature to modify or delete valuable data from the memory of the tag. This information might be critical like data about patient's health which any inconsistency between data stored on the RFID tag and the corresponding tagged object may results serious problems. In some cases, the reader may not even notice this inconsistency during the communication and thinks that the content of the tag is unaltered.

1.2.6 Physical Attacks

In this attack, an adversary takes advantages of the wireless nature of RFID systems in order to disable tags temporarily or permanently [5]. To permanently disable a tag, he may remove the tag form one item with high price and switch it with a tag of an item with low

price. The other way is sending a kill command to erase the memory of the tag. Removing the antenna or giving a high energy wave to tag will destroy the tag permanently. To disable the tag temporarily, a thief can use a Faraday cage like an aluminum foil-lined bag in order to block electromagnetic waves from it. In other case, he may prevent tags to communicate with readers by generating a signal in the same range as the reader broadcasts, call active jamming.

1.3 Physical Solutions

To protect privacy and security of RFID tags against possible attacks and threats, physical solutions can be helpful. In this section, some of these solutions are introduced and their pros and cons are investigated.

1.3.1 Killing Tags

In this method, RFID tags are "killed" upon purchase of the tagged product by a customer. After killing the tag, it is no longer functional and cannot be re-activated anymore. This approach is performed by sending a special command including a short password [8]. For instance, in a supermarket, the tags of purchased goods would be killed at checkout for protecting the privacy of consumers. Therefore, none of the purchased items would contain live RFID tags.

The advantage of this solution lies in the simplicity and effectiveness of the method. However, since in this method the tag cannot be reused, its lifetime is limited and it cannot be utilized for after-sale purposes while consumers may wish to keep them alive after buying them. For example, a smart fridge which keeps the expiration dates of groceries from their tags. Based on this information, it can also give a report of what is inside it and generate a list

of shopping list. Other examples of RFID-tag applications include theft-protection of belongings, and wireless cash cards. In these applications, the RFID tag is required to be alive when the customer buy it and it cannot be killed.

1.3.2 Sleeping Tags

The "sleeping" mechanism is another type of physical solution [7]. In this approach, the reader sends a "sleep" command including with a password to the tag to make it temporarily inactive. This method is similar to killing method with this difference that the sleeping tag can wake up and be activated as soon as it receives the command from the reader while in killing method, the tag can never be activated.

Sleeping approach offers this advantage to the user to switch the state of the tag between active and inactive. The problem of using this method is existence of this possibility that the password using for controlling tags might be overheard by eavesdropping attack.

1.3.3 Faraday Cage

Faraday cage is an easy way of protecting an RFID tag, inspired by the characteristics of electromagnetic fields, introduced in [9]. A Faraday cage is an enclosure designed made of conducting materials to exclude electromagnetic fields. Since any exterior radio signals cannot penetrate inside the cage, consequently, by keeping an RFID tag in this cage, no reader can have access to the tag to read it.

Figure 1-8 shows how a faraday cage shield enclosed tag from unwanted electromagnetic waves. The electromagnetic field pushes electrons of the cage toward the left. It leaves a negative charge on the left side and a positive charge on the right side of the cage. The result is that the electric field inside the cage is zero.

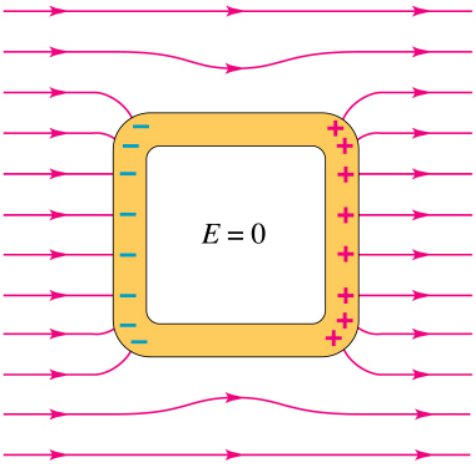


Figure 1-8 A faraday cage in an electric field

Faraday cages are extremely effective at providing consumer privacy against eavesdropping and tracking attacks. However, the main drawback of using this cage is its impracticality. The tag is protected from being read by unauthorized reader only when it is inside the cage. It might be practical for some items like smart cards, while using the cage is not convenient for a variety of objects like for tags injected under the skin or tags attached to a dress when it is being worn. The other problem is preventing being read by the authorized readers unless the tag is out the cage. Besides using a faraday cage for each tag imposes extra cost. These disadvantages put some limitations on using this approach which make this solution suitable for some particular applications.

1.3.4 Blocker Tags

A blocker tag is a physical solution for protecting privacy in RFID systems introduced in [10]. A blocker tag is similar to an RFID tag with this difference that it can block readers from reading identification of those tags that exist in the blocker tag rang.

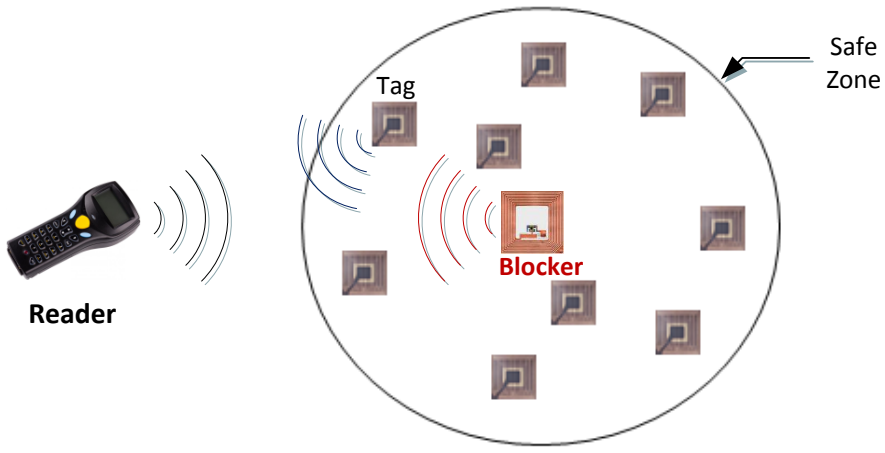


Figure 1-9 Blocker tags blocks reading by broadcasting signals for every reader's query

The operation of blocker tags is based on creating collision for a reader when it is attempting to identify tags in its field. To identify a tag from other tags, a reader sends a query asking its serial number. Since there is a possibility that a multiple of tags exist in the reader rang and respond to this query at the same time, the probability of occurring jam is high. Therefore, to resolve this collision, readers use some algorithms like tree walking. In this algorithm, each time the reader asks that only those tags which their serial number starts with a special number answer. If the reader still receives more than one response, it will continue by limiting the range of serial number until just one tag answers the query. The blocker tag uses this feature and by answering to all queries that reader broadcast, it fabricate a fake collision (Figure 1-9). So the reader is tricked into believing that all tags in its field are in interrogation zone. This way, a blocker tags can establish a safe zone around the tags and all RFID tags that exist in this zone can impede reading their data at the presence of a blocker tag.

One of the practical and attractive applications for blocker tag is using in a supermarket. Before purchasing the goods, their RFID tag can be read inside the supermarket

without any restrictions. When they are placed in the hands of the customer, a blocker tags might be added to the shopping bag to block all further communications. This blocker tags guarantee the customer's privacy against any threats until the items are removed from the shopping bag. Then they the tags of items can operate again like before.

The major advantage of this approach is keeping the functionality of tags. Unlike kill command that the lifetime of the tag was limited by purchasing it, this method allows the tags to be more useful by expanding their lifetime. However, the limitation on the safe is one of drawbacks of this method. The attacker cannot have access to tags just in a defined range and beyond this range, tags are not protected from attacks. Besides, blocker tags are not applicable everywhere. For example, in supply chain, tags are required to be available all the time and they cannot be blocked from being read by readers while the blocker tags imped all readers to have communications with tags even authorized readers.

Considering the limitations and drawbacks of physical solutions for providing security and privacy in RFID applications, these solutions are suitable for particular applications and cannot be applicable for all applications. The solution for this purpose is required to not make any limitation on the life-span of tags like killing method or block authorized readers like faraday cage. It also should not be restricted to a special zone like blocker tags. The suggested solution is using cryptographic algorithm to encrypt messages exchanging between tags and reader. In this solution, an adversary cannot have access to information by overhearing if he does not have the secret key. This solution also brings benefits like providing integrity and authentication which are not possible in physical solutions. However, this solution needs to be compatible with tags which are very resource

limited. In the next chapter, a survey of lightweight cryptosystems considered for RFID systems will be presented.

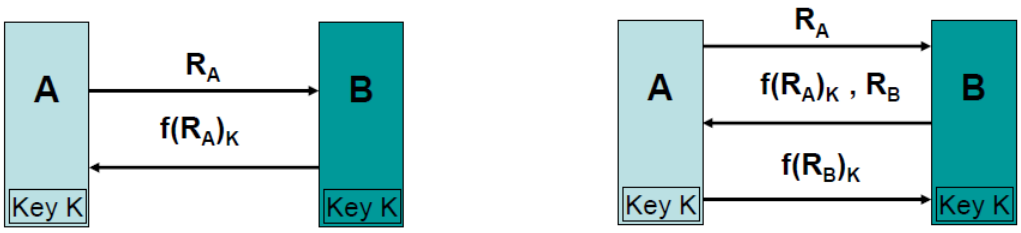
1.4 Authentication

Authentication is a process through which an object proves its claimed identity to other communication party with providing some evidence such as what it knows, what it has, or what it is. This process is applicable through only software solutions and it is not possible by physical solutions. In RFID systems, authentication is required in two phases. First, before beginning any communication, both tag and reader should verify their identity to make sure that they are contacting with the wished partner. Second phase is when data is exchanging between two parties to ensure that the exchanged data is intact.

When a tag passes through an electromagnetic field of a reader, it becomes activated and can detect the reader's signal. To reply to the reader, the tag needs to know if the reader is the legitimate one or not. Otherwise, an unauthorized reader can obtain information about tags which are currently in its field by eavesdropping and keep a tracking of their current locations. Also, an unauthorized reader can have access to the tag's memory to read or even manipulate its data. Therefore, to prevent these threats, a process is required to authenticate the reader to the tag. On the other hand, the reader is required to find out if the tag contacting with is reliable or not. This way, the reader can make sure that it is not communicating with a counterfeit tag. This process is called authenticating tag to the reader. Mutual authentication permits two parties to authenticate each other's identity. This happens when both tag to reader authentication and reader to tag authentication are performed. Conducting mutual authentication between RFID tags and reader should be performed before exchanging any

key and data. This way, all of the former mentioned security problems in the last sections can be solved.

Implementing unilateral and mutual authentication at the beginning of the communication has been interested in many researches. Authors of [11] presented three authentication methods. The first method, password authentication provides a weak level of security. Customized and zero-knowledge authentication is another technique based on mathematical problems which implementing them imposes high cost. Challenge-response is a high secure scheme which is being interested recently. This scheme is categorized into two groups: symmetric and asymmetric. Asymmetric technique is time consuming and its cost for implementation is high. On the contrary, symmetric methods need key exchange and management since they use one shared secret key (Figure 1-10).



(a) Unilateral Authentication (b) Mutual Authentication

Figure 1-10 Challenge-response technique in symmetric authentication [11]

During communication, providing authentication is required since there is a possibility that attackers send the message on behalf of each party or manipulate the message such that they replace their desired message with the real one. This service can be implemented by keyed hash function or Message Authentication Codes (MAC). Using MACs bring this benefit that the integrity of the message can be guaranteed. Authentication is essential when the possibility of existing attackers are high like battle fields or the condition of environment is harsh and may affect the accuracy of the messages. Also, in

applications that the value of data is important like health care, performing this service is vital.

1.5 Motivations

Radio Frequency Identification is a pervasive technology offering more reliable, accurate and faster identification of objects compared to other solutions. Reading data of objects from distance regardless of bad weather or day-light in this technology makes it applicable in a wide range of applications. These advantages bring this hope that in close future, it will become the substitute of its old competitor, barcode. However, to reach this goal there are still some obstacles which need to be considered and investigated.

Spreading RFID technology raises significant concerns in user privacy and security issues especially in secure applications such as authentication and payment systems. Having a secure communication is one of the main obstacles which will be removed by constructing a channel between the reader and tag preserving confidentiality, integrity and authenticity. Confidentiality makes having access to data of objects difficult for attackers without the secret key. Integrity assures the received message is intact and authentication confirms the sender of the message in the channel.

Defending RFID devices against malicious attacks is possible only through a strong protection mechanism. Since physical solutions put limitations on tags and readers, they are suitable only for particular applications while providing privacy may applicable partially. Cryptography is a respectable solution which can provide other services like integrity and authentication besides confidentiality. Contemporary cryptosystems are strong in terms of security. However, they require plenty of resources like power and area for their

implementation while RFID tags are very resource constraint devices. Therefore, introducing new cryptosystems which can provide security with satisfying these limitations, called lightweight encryption is necessary for RFID systems.

Recently some lightweight encryption algorithms have been introduced for this purpose. However, these algorithms are more concerned with confidentiality while authentication is a part of privacy. On the other hand, current hash functions are not suitable for constrained environments. Since they require significant amounts of resources in their designs, they are not hardware friendly at all [12]. In this thesis, a new symmetric encryption algorithm is presented which can provide confidentiality, integrity and authentication all together while the cost of its implantation is suitable for RFID systems.

1.6 Main Contributions

The summary of the key contributions of this thesis are stated in four main points as:

- A lightweight cipher- an important part of this research is proposing a new lightweight symmetric encryption algorithm called RBS stands for Redundant Bit Security. In this algorithm, confidentiality of plaintext is achieved through inserting some redundant bits inside the plaintext bits to change the location of plaintext bits inside the ciphertext. The location of redundant bits and plaintext bits inside the ciphertext is the secret key shared between sender and receiver. Experimental results in CHAPTER 5 shows that implementation of RBS requires less power and area compared to other known symmetric algorithms proposed for RFID systems especially when authentication is required. This saving in area overhead has a direct effect on implementation cost of RFID tags which is also one of the main concerns in

getting acceptance by industry. Regarding other metrics like energy per bit, hardware efficiency, Area-Time product and Power-Area-Time product, RBS offers better results. Since RFID tags are very resource constraint and they have strict limitations on the area and consuming power, RBS algorithm will be a promising candidate to provide confidentiality while in CHAPTER 4 resisting it against strong attacks is proved.

- Providing authentication and integrity- In RBS algorithm, inserted redundant bits in the plaintext are calculated in such a way that they can provide authentication and integrity services along with confidentiality. Offering these services is very important in some applications like health care. Also in environments that the possibility of manipulating transferred data is high by attackers or harsh condition. Implementing keyed hash function or MACs for this purpose requires high cost in area and power, while RBS offer these services with low cost.

- Flexibility in security level- The number of plaintext and redundant bits in the ciphertext are two important factors in defining the security level of RBS cipher. By increasing each of them or both of them, the security level of cipher will be increased. While increasing one of these parameters and decreasing the other one at the same time may leads to different results. Therefore changing these two parameters gives this ability to the designer to change the security level of the cipher to his desired level. The only part of hardware required to be updated with the security level is the MAC generator. If the number of redundant bits is constant, then the security level might be adjusted by the number of plaintext bits while the same MAC generator can

be used without any change in it. For example, if is there are 68 bits for redundant data, by changing the size of plaintext from 32 to 64 bits the key space will grow from 2^{86} to 2^{128}. Therefore different keys can be supported with the same hardware. The only restriction is the size of redundant bits which should be longer than plaintext at least for few bits to avoid collision. However, if the number of redundant bits changes the underling hardware is required to change.

1.7 Thesis Organization

In this chapter, RFID technology, known as one of Auto-ID solutions, was introduced briefly along with describing its components, types of transponder and their limitations in performing communications. Additionally, a number of attacks threatening this system plus some physical solutions for them were presented. Since this kind of solutions is not able to provide security and privacy for their consumers, software solution called cryptography is recommended in order to solve this problem by preventing attackers from having access to tags data without having the secret key. This solution also has this advantage that providing other services such as integrity and authentication will be feasible. After this introduction on RFID system, the reminder of this thesis is organized as follow:

CHAPTER 2 presents the concept of lightweight cryptography designated for resource constraint designs such as RFID systems. A survey of cryptosystems which are compatible with this definition is introduced here. For each of them, possible attacks which make them vulnerable to is investigated. At the end, the results of their hardware implantation at different platforms are given.

The algorithm of RBS which is based on inserting redundant bits is described in CHAPTER 3. Then the level of provided security in this algorithm, the location of redundant bits, the value of redundant bits and the method of appearing plaintext in the ciphertext are defined. This is followed by hardware implementation of the RBS cipher. This part is consisted of two parts. First part is implementing redundant bit generator which is adapted from a MAC generator. Since it is designed for stream ciphers some modification is performed on it to make it compatible with block ciphers. The second part is implementing encryption/ decryption ciphers. This part is integrated with transmission and reception parts of an RFID transponder.

CHAPTER 4 introduces some powerful and common attacks such as known-plaintext, chosen plaintext, related key attacks etc. Then it is shown how RBS algorithm is resisting against these kinds of attacks.

In CHAPTER 5, the results of hardware implementation for RBS cipher is given and its one-dimensional and multi-dimensional performance metrics in ASIC design such as area, power consumption, energy and hardware efficiency are evaluated. Afterward, these results are compared with other lightweight cryptosystems introduced in CHAPTER 2. Since RBS cipher provides authentication for all of messages, this comparison is performed in two categories. First when none of competitor ciphers support the authentication service and second, when all of them do.

CHPATER 6 summarizes the work achieved in this dissertation by providing concluding remarks and presenting further research perspectives.

CHAPTER 2: Related Works

To provide security and privacy in RFID systems, Physical solutions are not suitable because of their limitations and disadvantages. Instead, cryptography seems inevitable way to make RFID technology secure. From a theoretical point of view, standard cryptosystems might be an accurate approach. However, they demand resources far more than those available on many tags in terms of circuit size, power consumption and area. Since low-cost RFID tags are very constrained devices with severe limitations in their budget, lightweight cryptographic techniques appear to be the most appropriate solution for this kind of RFID tags.

In this chapter, first the characteristic of a lightweight cryptosystem will be defined. Then, a selection of well-known and recently published lightweight-cryptography implementations will be presented. This survey covers recent hardware implementations of symmetric as well as asymmetric ciphers.

2.1 Cryptography

Cryptography is studying different techniques concerned with keeping communication between two parties private at presence of third parties. An encryption scheme has five ingredients: plaintext, encryption algorithm, secret key, ciphertext, and decryption algorithm. In these techniques, a message, called plaintext will be converted at the sender party by a secret key and an algorithm or mathematical procedure such that the result, called ciphertext, appears non-sense for all parties. The used algorithm for encryption and decryption is available for all parties while the secret key is shared only between the sender

and the receiver. To protect data and systems against adversaries the following four requirements are essential:

Confidentiality: Only the sender and the intended recipient of a communication can see the content of that communication. This concept is accomplished through encryption.

Data Integrity: It guaranties that the data received at the reception party is original and was received exactly as it was sent by the sender party. If the content of a communication is compromised it must be detectable by either communicating parties. Data integrity can be threatened either by environmental hazards, such as heat, dust, and electrical surges or by attackers.

Authenticity: The sender and the recipient should be able to verify each other's identity. Any impostor needs to be either detected or identified.

Non-repudiation: It means preventing an entity from denying previous actions. In other words, the sender of the message cannot deny having sent the message.

Among these four services, confidentiality is the primary service and all security algorithms are required to provide it, while other services are arbitrary.

2.2 Lightweight Cryptography

Lightweight cryptography is an innovative approach which is concerning solutions to meet the challenge of developing fast and efficient security mechanisms for harsh resource constrained environments. These solutions include new design in cryptographic primitives and protocols in addition to adapting and modifying contemporary cryptosystems [13].

To design a lightweight cryptography, there are three objects which are required to be optimized; security, performance and cost. Security is measured with the number of bits of key. By increasing the size of key, the provided security will be higher. Performance is considered in terms of the total number of clock cycles to complete an operation which is proportional with throughput and energy. Cost like power and area, depends on the utilized architecture. Among these three objects, there is a trade-off which makes optimizing all of them together in one design very difficult (Figure 2-1). For example security is in trade-off with performance and cost. Having high security requires increasing either the number of rounds or cost. Performance and cost are two other vertexes of this triangle. Serialized architecture yields lower power and area while it results lower performance.

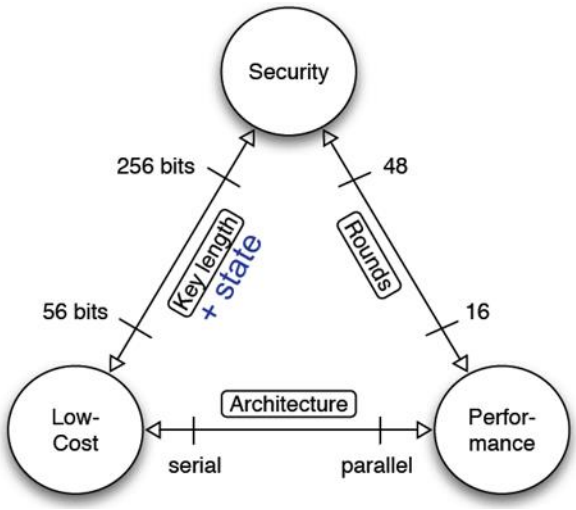


Figure 2-1 Design trade-offs for lightweight cryptography [13]

To have a more precise definition of lightweight cryptography, it is required to define the boundaries of cost and performance. Power consumption of the security implementation has to be reduced to10s of microwatts, and for EEPROM read operation this limitation should not exceed it unless the tag read range requirements cannot be preserved [14]. A

complete RFID tag, including the analog part, might have between 1000 to 10000 GE and for security part; this margin may be kept between 200 to 2000 GE [15]. Performance is mainly limited by user requirements and air interface protocols. However it is recommended to be 10s to 100s clock cycles.

In the following sections, some research towards lightweight cryptography is studied. In this survey, some new lightweight design and some modified contemporary cryptosystem will be investigated separately. At the end, a comparison of these designs will be given.

2.3 Asymmetric Key Encryption

Asymmetric key encryption algorithms, called also public key algorithms are very strong in terms of security. They provide confidentiality, integrity, reliability, availability and non-repudiation altogether. In this cryptography, two different keys are used: public key which is published on the network and private key which is kept secret by user. To encrypt a plaintext, a public key is enough, but to decrypt the ciphertext, a corresponding private key is required. Thus every part can encrypt a message while only the party has the private key can recover the message. Public-key constructions are typically based on some mathematical problem, such as factoring, which is assumed to be a hard problem in a computational sense. For example, in factoring, the private key can consist of two large prime numbers and the corresponding public key is their product. Obtaining the private key from the public is possible in theory, but in practice, a huge resources e.g., time is required to compute it.

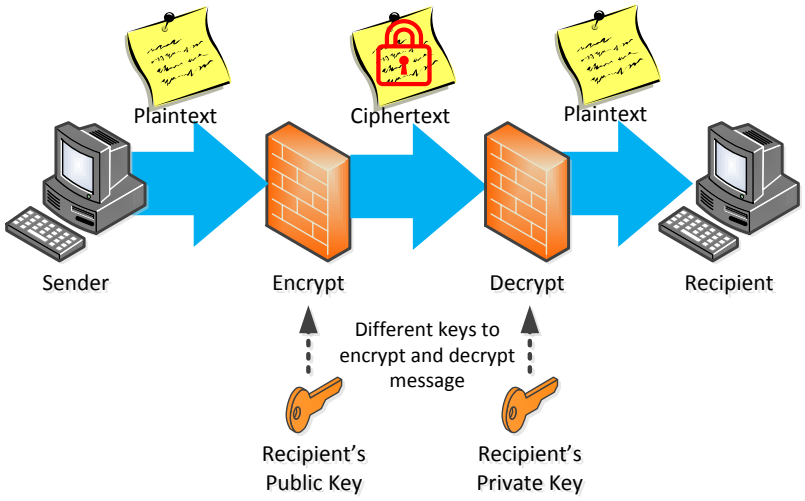


Figure 2-2 Asymmetric key encryption

One of the advantages of Asymmetric key cryptosystems is distributing key among parties. Since it is not required for all parties to keep the encryption key in private, no key is required to be exchanged among involved parties.

Public key algorithms are extremely secure compared to private key algorithms. However, their implementations are much more complex as well. As a result, their computation speed is relatively poor. Although speed up through hardware implementation is possible through parallelism, public key hardware systems use more die space and usually require more power than private key systems. Furthermore, since public key algorithms often rely on complicated mathematical computations, they are generally much more resource hungry compared to private key algorithms. Nevertheless, some researches have been done towards adapting public key algorithms with resource restricted applications. Here, one of the well-known public key algorithms, ECC is studied.

2.3.1 Elliptical Curve Cryptography (ECC)

Elliptical curve cryptography, ECC is a public key encryption technique based on elliptic curve theory over finite fields. ECC-based systems offer similar security for smaller key sizes compared to RSA-based systems [16]. Since the computational and area complexities of hardware implementations for cryptographic algorithms are proportional to their key sizes, ECC-based systems are smaller, faster, and consume less power compared to RSA-based systems.

In ECC cryptosystem, all parties agree on all parameters defining the elliptic curve and a base point on this curve. Each party selects a number as a private key and compute multiplication of the base point with its private key. The result of multiplication will be another point on the curve which is published as a public key. Finding the original point from the result is very difficult even with knowing the base point. This property guarantees the security of ECC algorithm.

To encrypt a message in ECC, the sender will first compute shared secret key by multiplying the receiver's public key with its own private key. Then the message is added to this shared key and sent out.

A lot of research has been done on hardware-efficient ECC implementations. In [17], the authors have tried to adapt ECC algorithm with RFID systems by reducing the number of registers, operations, the operation frequency and also using restructured formulas as much as possible in order to meet the resource limitations of RFID systems. However, their proposed hardware is still far from the boundaries of RFID systems.

Making public key algorithm lighter is another solution. Reducing the flexibility of ECC algorithm by limiting the number of parameters such as using only one special elliptic curve [18], selecting specific field sizes [19] or choosing specific prime numbers [20], [21], [22], [23] are other ways to make ECC lighter. Although applying dedicated hardware with these limitations leads to meet the power limitation, but any change in security parameters imposes replacement of all tags with new ones. The result of hardware implementation of these designs in Table 2-1, indicates that they are still away from the definition of lightweight cryptosystem in terms of area, performance and power despite of all improvements performed in them.

Table 2-1 Hardware implementation results for ECC

Design	**Bits**	**Area [*gates*]**	**Techn. [*μm*]**	**Op. freq. [kHz]**	**Perf. [*ms*]**	**Power [*μW*]**
[17]	226	16.9 K	0.18	1280	N/A	6.6
[19]	131	11969	0.35	13560	18	---
[21]	100	18 720	0.13	500	410.45	< 400
[20]	134	6103	0.13	200	210	<13
[18]	163	12506	0.13	1130	244.08	36.63
[23]	167	30333	0.13	20000	31.9	990
[22]	61	18720	0.13	500	817.7	394.4

2.4 Symmetric Key Encryption

Symmetric key encryption is the oldest and best-known technique to provide security in communications. In this technique, the sender and the receiver both share a secret key, which they have already agreed on. The shared key is used for both encryption and decryption (Figure 2-3). This setting, referred also to as private key cryptography, is considered to be confidential if only eligible parties whose have access to the shared secret key can recover the plaintext from the ciphertext.

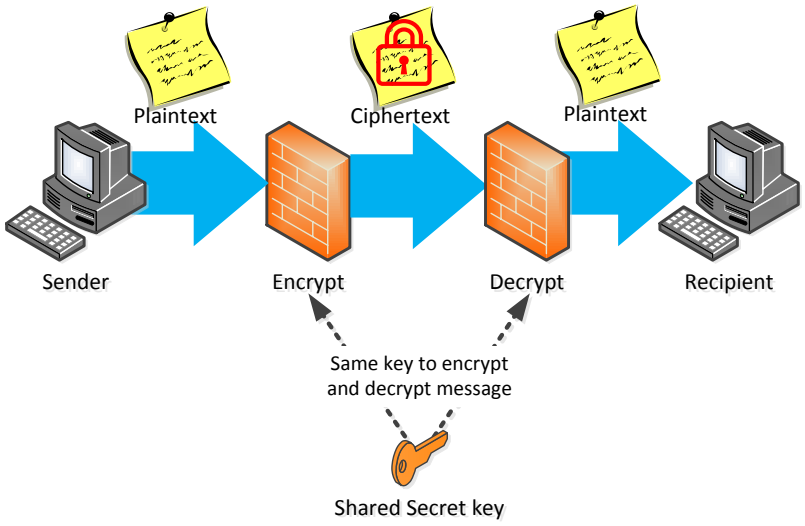


Figure 2-3 Symmetric key encryption

There are several drawbacks which make symmetric key algorithms less interested in some applications. One of the obvious problems is distributing private keys among authorized parties. Moreover, keeping one secret key for each party makes managing keys more difficult by increasing the number of parties. Symmetric encryption algorithms cannot provide integrity and authentication alone. To provide these services, they need other algorithms to be integrated with them. Not supporting non-repudiation service is one of other problems of these cryptosystems. Despite all of these drawbacks, there are efficient software and hardware implementations for private key algorithms which make them suitable for restricted resource applications. Therefore, since Public key algorithms have still significant challenges for RFID systems' implementation, recently researches have been directed towards Private Key schemes.

There are traditionally two classes of symmetric encryption algorithms: block ciphers and stream ciphers. Recently, a new class, called hybrid cipher which is a combination of these two ciphers has been introduced.

2.4.1 Block Ciphers

Block cipher is an encryption function that works on fixed size blocks, typically 32 to 256 bits. For example, AES performs on blocks of 128 bits, while other block ciphers use smaller block sizes such as 64 in PRESENT [24]. Therefore, the size of the ciphertext is fixed independent of the size of the message. In general, in block ciphers, a block of N-bit plaintext is replaced with a block of N-bit ciphertext. Block ciphers like DES, 3DES, AES breaks message into blocks. Then, each of blocks is encrypted while the key is same for every block (Figure 2-4). These ciphers repeat one or more simple operations like substitution and permutation, several times. Encryption process is different from decryption process in block ciphers.

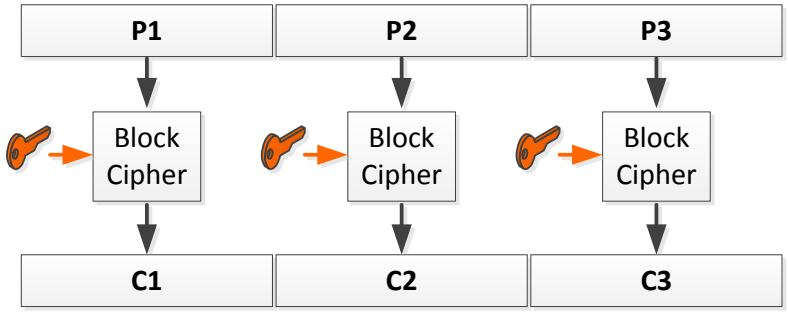


Figure 2-4 Block cipher operations on fixed size of blocks

To provide confidentiality of a communication, ciphers are required to obscure statistical properties of original message completely by providing confusion and diffusion between the message and the key.

Confusion is a way to make the relation between the plaintext and the ciphertext as complex as possible. It can be achieved by using a complex substitution algorithm. Thus, even if an attacker can handle on the statistics of the ciphertext, it is very difficult to assume

the key. Caesar ciphers have poor confusion while Polyalphabetic substitutions have good confusion.

Diffusion is a way to spread the effect of changing the individual plaintext over the value of ciphertext digits as much as possible, like permutation or transposition ciphers. By globalizing the local effects, tracking effects of each plaintext digit on the ciphertext digits will be more complicated for an attacker.

One type of modern block ciphers is substitution-permutation network or SP which is based on the two primitive cryptographic operations: substitution box and permutation box.

Substitution Box or S-Box is a basic component in block ciphers which substitute n-bit data in the input with m-bit data in the output. Usually m and n are equal. S-Boxes are fixed look-up tables (LUT), used to provide high non-linearity and high Boolean function complexity relationship between the plaintext and the ciphertext to satisfy the confusion property in block ciphers. They use big area which makes them expensive in hardware implementation. For example, 8×8 S-Box as found in AES [25] needs 300 GE, 6×4 S-Box in DES [26] requires 120 GE and 4× 4 S-Box as used in PRESENT [24] is implemented by 28 GE. In contrary, S-Boxes are suitable to be implemented by software because they can be replaced by small size of memories in software implementation. For example, for software implementing 8×8, 6×4 and 4×4 S-Boxes, 256, 64 and 16 Byte ROM memories are required respectively. Thus, the selected S-Box is required to be small in hardware implementation to save more cost in area. Since S-Boxes can be implemented in a single LUT, hardware Implementation of S-Boxes in FPGA is easily applicable with saving cost in area.

Permutation Box or P-Box is another helpful tool for encryption in block ciphers. It is a basic component in block ciphers which performs reordering on n-bit input to n-bit output to satisfy diffusion property of block ciphers. It is a reversible function; therefore it can be used to retrain the message by the same hardware. Permutation is very suitable for hardware implementation since no gate is required and it is composed of just wiring. However, it brings complexity in routing in low level design fabrication. Since there is no gate, no transition will occur. Thus, no extra delay and power cost will be imposed. In contrary, this method is difficult in software implementation, since it needs cumbersome bit operations. For example, permutation of 64 bits needs 64 cycles and 64 Byte ROM memory.

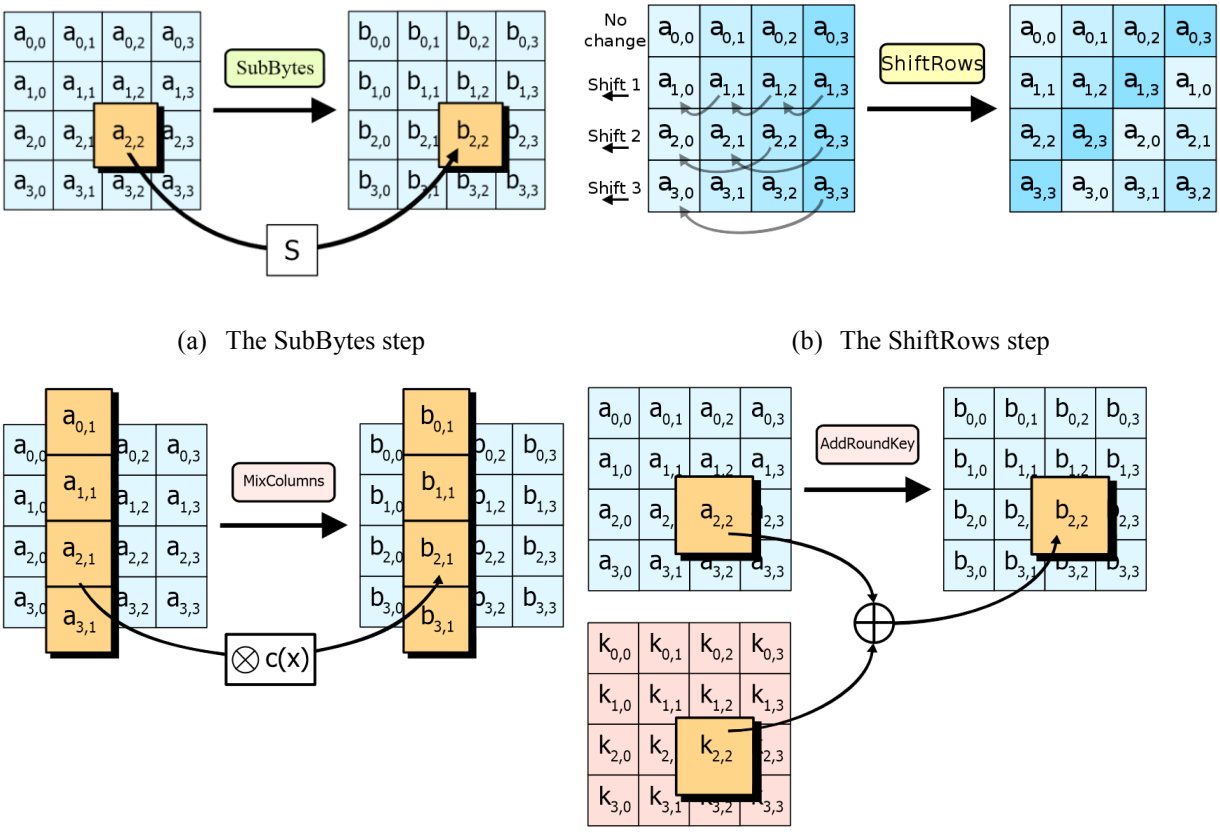
Several mature block ciphers based on S-Box and P-Box are available for limited resource environments like AES and PRESENT which will be discussed in the following sub-sections.

2.4.2 Advanced Encryption Standard (AES)

AES is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST) in December 2001. It is the successor of DES and an example of SP network which operates on fixed 128-bit block of data with supporting 128, 192, or 256-bit key sizes [27]. It is organized in a 4×4 column-major order matrix of bytes, called STATE. The number of rounds in AES depends on the size of the key, e.g. 10 rounds for AES-128. To provide confidentiality, AES uses four types of transformations at each round:

- SubBytes- Each byte in the STATE matrix is replaced with a SubByte using an 8-bit Sbox (Figure 2-5.a).

- ShiftRows- Each row of the state is shifted cyclically a certain number of steps (Figure 2-5.b).
- MixColumns- Four bytes of each column of the state are combined using a linear function (Figure 2-5.c).
- AddRoundKey- Each byte of the STATE is combined with the round key using bitwise addition (Figure 2-5.d).



(a) The SubBytes step

(b) The ShiftRows step

(c) The MixColumns step

(d) The AddRoundKey step

Figure 2-5 Four steps in AES [28]

Among block ciphers, AES is well known block cipher for encryption. Many low-cost implementations of the smallest variant, AES-128 have been published which bring down the size of cipher to only 3100 gate equivalents [29]. An ultra-low power and low

energy AES design is presented in [30]. However, the best known lightweight AES design requires 3100 gate equivalent (GE) for implantation [29], which is still significantly higher than the assumed 2000 GE. So it is not a good candidate for extremely constrained device like RFID systems. This might be due to this fact that the AES has good software implementation properties but it is not designed with hardware-friendly properties. So, the gate count for a hardware implementation of AES is not very likely to further decrease. Therefore, AES is often considered out of the option for developing such technology. Instead, it is considered as a benchmark for comparison of different encryption algorithms.

Table 2-2 Results of implementation of AES

Platform	Area [GE]	Tech. [µm]	Max Freq. [MHz]	Cycles per Block	Throughput [Mbps]	Power [µW/MHz]	Mode
[25]	3400	0.35	80	1032	9.9	45	En/De
[29]	3200	0.13	130	160	104	30	En
[30]	4070	0.13	N/A	534	N/A	47.66	En/De

AES is known to be vulnerable to the following attacks: known-key distinguishing attack [31], chosen-key-relations-in-the-middle attacks [32], key-recovery attacks based on bicliques [33].

2.4.3 PRESENT

PRESENT is a block cipher based on a SP network, inspired by the techniques used in DES and AES and designed for resource constraint systems. It operates on 64-bit block of data, supporting 80-bit and 128-bit keys [24].

generateRoundKeys()
for i= 1 to 31 **do**
    addRoundKey(STATE,K_i)
    sBoxLayer(STATE)
    pLayer(STATE)
end for
addRoundKey(STATE,K_{32})

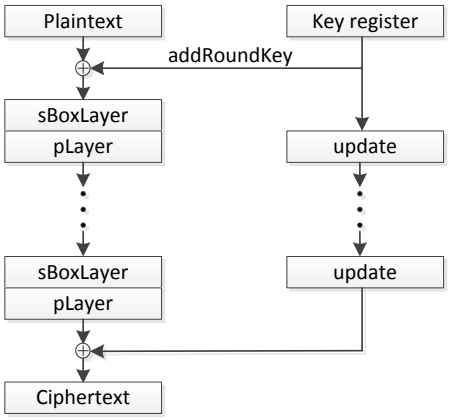


Figure 2-6 A top-level algorithmic description of PRESENT algorithm [24]

The encryption process is completed in 31 rounds. Each round composed of three layers: addRoundKey layer, sBoxLayer and player (Figure 2-6). These three layers are followed one by one at each round. At the beginning of encryption process, the Key register and STATE are initialized with the encryption key and plaintext respectively. At each round, the key register is updated by rotating 61 bits to the left and the round key is loaded with the 64 leftmost bits of the key register.

In the first layer at each round of addRoundkey, the STATE is updated by performing bit wise addition on STATE with the round key. The second layer is sBoxLayer, consisted of 16 copies of a 4-bit to 4-bit S-Box, S0-S15. The current state is divided into sixteen 4-bit words, fed into S-Boxes. The content of the used S-Box in PRESENT is shown in Table 2-3. This is one of the smallest S-Boxes in hardware implementation with 28 GE for each.

Table 2-3 The PRESENT S-Box [24]

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

The third layer is pLayer, performing permutation on bits of STATE. This layer changes the place of bits in the STATE. Figure 2-7 shows one round of PRESNT cipher composed of three layers.

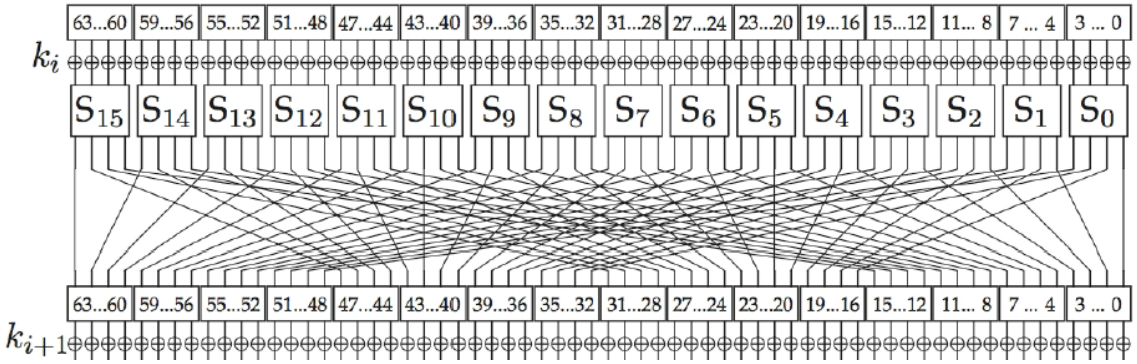


Figure 2-7 Three layers at one round in PRESENT cipher [24]

PRESENT cipher uses S-Box and permutation components which are appropriate for implementing in FPGAs. Implementing PRESENT cipher on FPGA in [34] shows that only 117 LUT slices are need which makes it comparable in size to other ciphers.

Encryption and decryption processes in PRESENT cannot be processed in same hardware because reverse of S-Box is different in decryption process from encryption process. The implementation result of PRESENT encryption cipher in different architectures is shown in Table 2-4 [13]. PRESENT is also has been introduced to provide MAC with different key and output sizes, however, PRESENT is still away from limitations of RFID systems because of high area requirements.

Table 2-4 Hardware implementation result for PRESENT at 100 KHz frequency [13]

Design	Key size	Datapath width	Cycles/ Block	Throughput [Kbps]	Tech. [um]	Area [GE]	Current [uA]
Serialized	80	4	547	11.7	0.18	1075	1.4
Serialized	128	4	559	11.45	0.18	1391	N/A
Round-based	80	64	32	200	0.18	1570	2.78
Round-based	128	64	32	200	0.18	1884	3.67
Parallelized	80	64	1	6400	0.18	27028	38.3

PRESENT cipher has been proved that is secure against following attacks: Statistical Saturation [35], Algebraic-Differential attack, differential attack [36], Linear with weak keys [37], Multidimensional Linear [38], Bit-Pattern Integral [39], Related Key Rectangle [40], Linear Hull [41]. However in [42] it is shown that at the most 30 sub-key bits can be recovered by the attack given in [36] after some modifications in that algorithm. Authors of [43] have proposed improved side channel cube attacks which can reveal 48 bits of key with 211.92 chosen plaintext in PRESENT-80.

2.4.4 Stream Ciphers

Stream cipher is a function that processes the message bit by bit as a stream. They operate with a time-varying transformation on individual plaintext digits, inspired by on one-time pad concept.

One-time pad (OTP), also called Vernam-cipher [44], is a crypto algorithm where the plaintext is encrypted with a secret random key. The encryption is performed by adding the key to the message modulo 2, bit by bit. The decryption is accomplished by the same function (Figure 2-8). If P_i, C_i and K_i, are plaintext, ciphertext and key bits respectively then:

$$\begin{aligned} &\text{Encryption: } C_i = P_i \oplus K_i \quad i=1, 2, 3\ldots \\ &\text{Decryption: } P_i = C_i \oplus K_i \end{aligned} \tag{2.1}$$

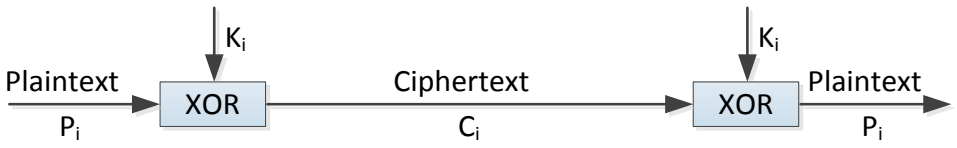


Figure 2-8 One-time pad cipher

It is proved that in OTP cryptosystem, the ciphertext will be impossible to decrypt or break unconditionally under the following conditions [45]:

- The key must be at least as long as the message or data that is being encrypted.
- The key must be truly random, not generated by a simple computer function.
- Key and plaintext are calculated by modulo 10 (digits), modulo 26 (letters) or modulo 2 (binary).
- Each key must be used only once, and both sender and receiver must destroy their key after using it.
- There should only be two copies of the key: one for the sender and one for the receiver.

One-Time Pad has the advantage that it is faster and less complex in hardware than other cryptosystem but the challenge is the key which must be as long as the message. This challenge makes using it impractical for almost all applications. On the other hand, the decrypting party must have access to the same key to encrypt the message and this raises the problem of how to convey the key to the decrypting party safely or how to keep both keys secure because of difficulty in key distribution and management.

2.4.5 Keystream

Stream cipher is a practical scheme which the infinite secret key in one-time pad cipher is replaced with a keystream. Keystream is a pseudorandom digit stream, generated from a secret key of finite length while the keystream is independent of plaintext and ciphertext (Figure 2-9). This scheme is close to one-time pad with this difference that the secret key is a seed to generate a stream for encryption and decryption.

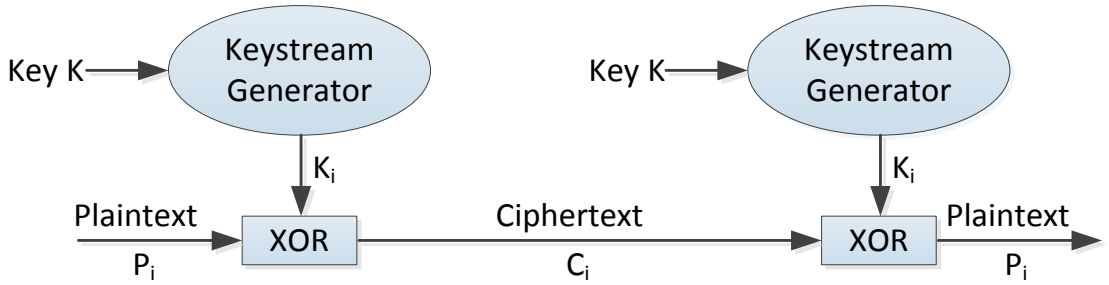


Figure 2-9 Keystream generator scheme

The proposed scheme is never secure theoretically since attacker can always try all possible keys 2^k- brute force attack. Thus the goal is to make it secure computationally. Since the keystream generator is only able to produce 2^k distinct keystreams, if a key is reused with a stream cipher in two different sessions, the exact same keystream will be produced. So the attacker by comparing two different ciphertexts can find the keystream and consequently plaintexts easily. On the other hand, exchanging the key for each session is not practical. To solve this problem, modern stream ciphers utilized initial vector (IV). While the key is secret and constant between sender and receiver, IV is public among all parties and after some sessions, it is renewed and published over the network. These two, key and IV, combined together will generate distinct keystreams for each session. Regarding Figure 2-10, stream cipher operation consists of the following phases:

1. In the initialization phase, the secret key and public IV are loaded into a state register. The state is updated in some clock cycles, without producing any output to blend the key and the IV such that a change in the IV yields a completely different keystream. By setting up the internal state, the cipher will be ready for next phase to generate keystream.

2. In encryption/decryption phase, the keystream is generated by updating the next state. Then the next block of data is encrypted/decrypted by the generated keystream.

3. After several communications, a new session by publishing another IV starts while the secret key is same.

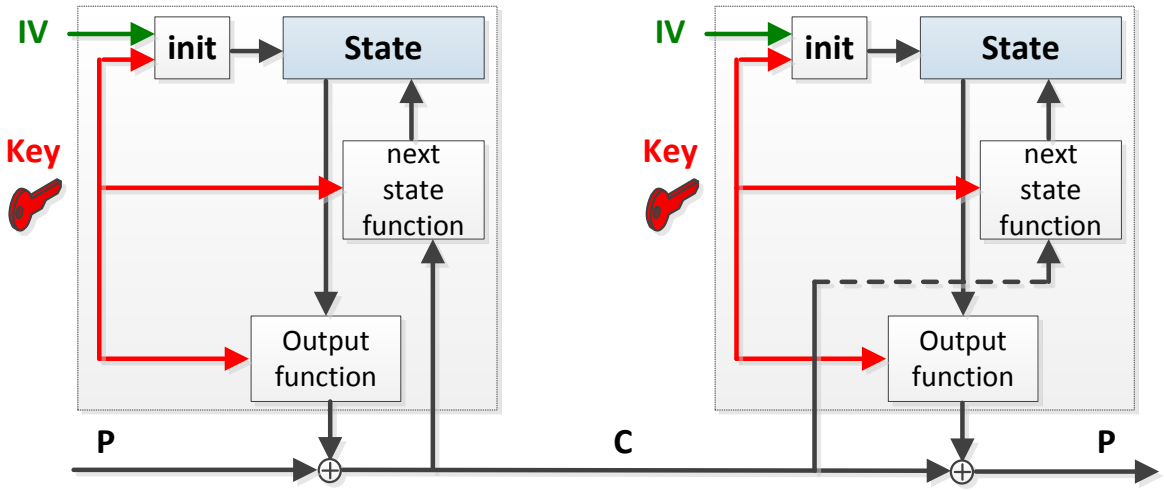


Figure 2-10 Stream cipher operation

The main part of stream cipher is the keystream generator. This part is required to be capable of producing long pseudo random sequence for any key while the security of the cipher does not depend on the IV. To build a keystream generator, there are some basic blocks and mathematical operations suitable for generating random streams such as LFSRs with low complexity and good statistical properties, S-Boxes and Boolean functions to provide nonlinearity and bitwise addition (mod 2n) which helps in making nonlinearity and breaking associativity.

LFSR is a shift register whose present state is a linear function of its previous state. This register will produce a stream which will be repeated after a while. The length of the

stream is dependent of its polynomial function C(x) (Equation 2.2). To have the maximum length, LFSR function is required to be primitive.

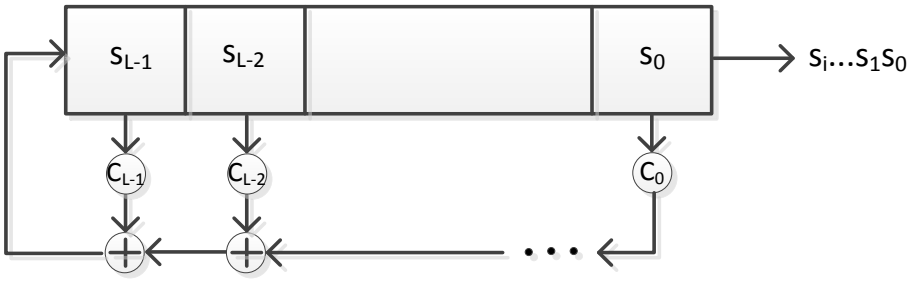


Figure 2-11 Typical LFSR

$$C(x) = c_0 + c_1 x + c_2 x^2 + ... + c_{L-1} x^{L-1} \qquad c_i = 0 \text{ or } 1 \tag{2.2}$$

A maximum length LFSR satisfies pseudorandom assumes but the problem is linear recursion. With 2k output bits, the initial state can be recovered by using algorithms like Berlekamp-Massey [46]. To destroy nonlinearity, different stream ciphers have found different solutions. One of them is applying two or more outputs of LFSR into a non-linear function. The other solution is applying a nonlinear filtering, e.g. a Boolean function and feeding it back to the input of LFSR which is called NFSR. NFSR is a shift register which generates a nonlinear relation of the states. To provide non-linearity, a high linear function and bent function will be used together. This approach is used in Grain and Trivium ciphers.

2.4.6 Trivium

Trivium is a synchronous stream cipher, designed to be compact in area and fast for high throughput applications [47]. Trivium supports 80-bit private key and 80-bit IV. It is composed of three NFSRs with different length, 93, 84 and 111 bits. In each clock cycle

three NFSR registers are updated while bitwise addition of their outputs generates the keystream (Figure 2-12).

For initialization, 80-bit IV is loaded into first NFSR, 80-bit key is loaded into second NFSR while all bits of third NFSR are set with zero except three last bits which are set to one. To start the encryption process, 1158 clock cycles are required before having the first output.

One of the advantages of Trivium is small area. It can be implemented with 228 registers, 3 AND-gates and 7 3-input XOR-gates. The minimum area reported for implementing Trivium is 1294 GE In [48]. To speed up the cipher, it is possible to implement Trivium in parallel with different radix. Table 2-5 shows the results of two different implementations of Trivium with radix one.

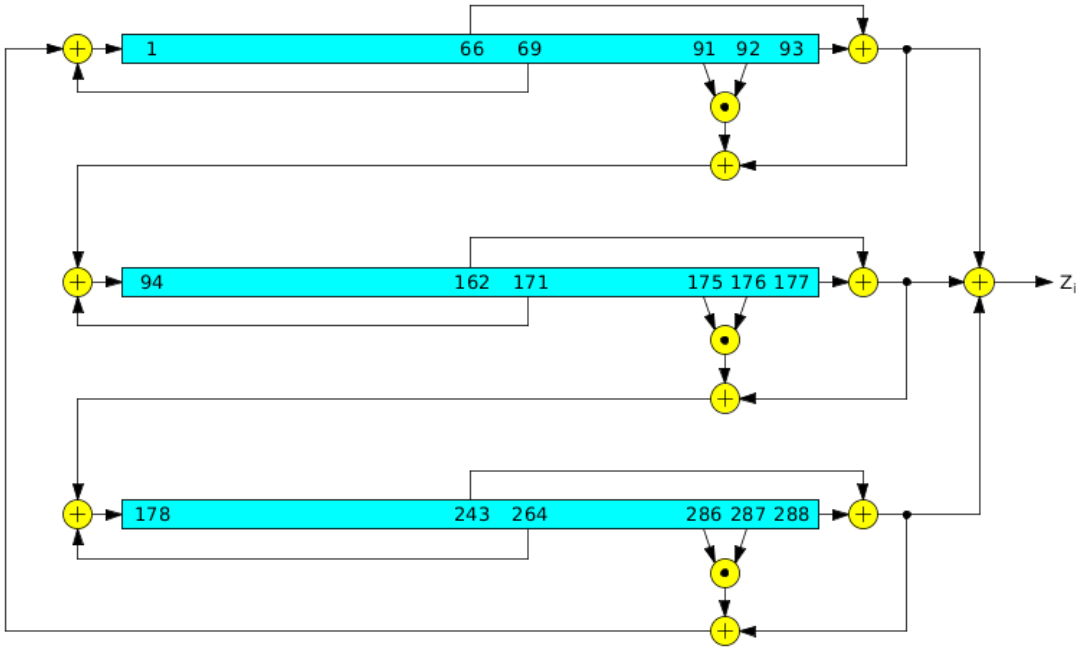


Figure 2-12 Hardware implementation of Trivuim [47]

Until 2010, no cryptanalytic attacks better than brute force attack were known for Trivium. However, several attacks come close to it like cube attack [49], Algebraic IV Differential Attack (AIDA) [50] and also proposed attacks in [51] [52] which made Trivium to increase the length of the key beyond 80 bits.

Table 2-5 Implementation results for Trivium

Design	Cycles Init.	Tech. [µm]	Area [GE]	Power [µw] @ 100KHz
[48]	1333	0.13	2599	5.6
[53]	1607	0.35	1603	1.06

2.4.7 Grain

Grain is a hardware-oriented stream cipher with small area overhead, designed for limited resources environments. The first version of Grain supports 80-bit key and 64-bit IV [54]. The second version supports key size of 128 bits and IV size of 96 bits [55] along with optional authentication. The design of this cipher is very simple and based on two shift registers, one linear and one nonlinear, and three functions f(x), g(x) and h(x) (Figure 2-13). f(x) is a linear function while g(x) and h(x) are non-linear functions.

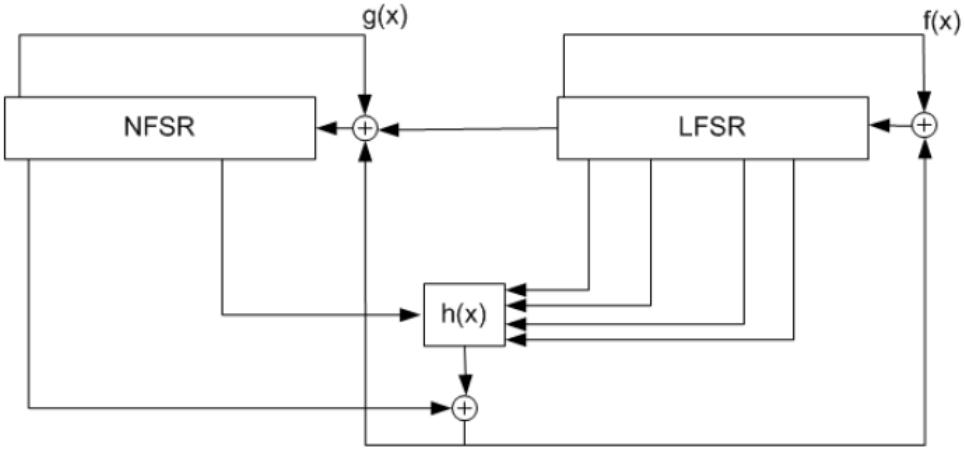


Figure 2-13 Grain cipher [54]

At the beginning of encryption process, LFSR and NFSR are initialized with IV and key respectively. Then the cipher is clocked 160 (first version) or 256 (second version) times without producing any keystream. The generated keystream is the output of h(x). To speed up the Grain cipher, it is possible to implement it in parallel with different radix. Table 2-6 shows the results of two different implementations of Grain with radix one reported in 0.13µm [48].

Table 2-6 Implementation result for Grain cipher with different key sizes [48]

Key [bits]	Cycle Init.	Cycle/ Bits	Max freq. [MHz]	Area [GE]	Leakage Power [µw]	Total Power [µw] @ 10MHz
80	321	1	724.6	1294	2.22	109.45
128	513	1	925.9	1857	2.70	167.73

Grain cipher supports an optional authentication message with at least 32-bit size which will be appended to the end of the ciphertext before transmitting it. Implementation result of Grain cipher in Table 2-6 does not cover the implementation of authentication part.

To prevent substitution attack in Grain, it is required to refresh the authentication key after each communication unless the key will be revealed after two or three communications [56]. The first version of Grain is vulnerable to a related key attack [57] and an algebraic attack with a weak Key-IV [58]. The second version of Grain is found to be immune against dynamic cube attacks and also differential attacks [59]. Until now, no attack is reported to break down Grain-128.

2.4.8 Hybrid Ciphers

Block ciphers and stream ciphers are two main groups of cryptosystems which are popular in light cryptography. Each of this group has its own advantages and disadvantages. Stream ciphers are interesting for two reasons. First, they are faster in software applications.

This main advantage is important when an enormous amount of data is being encrypted like video stream. Another advantage of stream ciphers lies in their design perspective by having a low circuit complexity. However stream ciphers require a considerable amount of time for initialization before generating first output. This disadvantage is not important since initialization happens during the algorithm startup or whenever the key changes.

Compared with block ciphers, which their security is well understood particularly against statistical attacks, security of stream ciphers is still require more research. In block ciphers, several bits of data as a block are encrypted together. Therefore, when the message size is not a multiple of the block size, some bits are required to be padded to the message. If any bits of ciphertext changes during transmission, it might be explored by receiver in encryption process, since this bit may affect the entire message. Despite of block ciphers, in stream ciphers, each bit is encrypted independently. Therefore, the sender and the receiver must be synchronized to confirm that the sender applies the right key sequence to the given bit of the plaintext. If any error happens during transmission, this error will be propagated to all bits of the ciphertext and all decrypted plaintexts will be wrong. Bits deletion or bits insertion is one of effective tools used by attackers to break the cipher. Besides, Initial vectors may also bring new opportunities for attackers.

Authentication in stream cipher is essential because the ciphertext is the result of bitwise addition of plaintext and keystream. Therefore by changing one bit of the ciphertext, the corresponding change in the plaintext will be easily predictable, while in block cipher the corresponding block is altered in a completely unpredictable way. For example, if the plaintext contains an amount of money, in stream ciphers, the attacker might be able to alter this amount by altering some bits of ciphertext.

Stream ciphers have this advantage that they encrypt and decrypt data with the same algorithm since encryption process is same as decryption process. While a block cipher may need to have two different algorithms for encryption and decryption processes which may impose extra area overhead for hardware implementation. On contrary, a block cipher is stateless so the ciphertext is not a function of the time, while a stream cipher has an internal state.

Properties	Block Ciphers	Stream ciphers
Message size	Fixed (+padding)	Variable length
Memory	Stateless	Internal state
Core	Encryption + Decryption	Encryption = Decryption
Equivalent	Random permutation	PRNG
Model	Diffusion + Confusion	One-Time Pad

Figure 2-14 Comparing properties of block ciphers and stream ciphers

Block ciphers and stream ciphers are compared in the figure 2-14. Hybrid ciphers are those ciphers which inherit some of their properties from block ciphers and some other from stream ciphers to receive benefits from both of them. The best example for these ciphers is Hummingbird (HB) cipher.

2.4.9 HB (Hummingbird)

Hummingbird has a hybrid structure of block cipher and stream cipher, providing the designed security with block sizes as small as 16-bit block. It is specially designed for resource-constrained platforms. The first generation of HB was designed to provide 256-bit security and 80-bit internal state [60]. However it is showed that HB-1 is vulnerable to a chosen-IV and chosen-message attack in [61].

The Hummingbird-2 cipher has a 128-bit secret key and a 128-bit internal state which is initialized by a 64-bit Initialization Vector, IV [62]. Used functions in HB-2 are the exclusive-or operation on words, addition modulo 65536 and a nonlinear mixing function f(x). The nonlinear mixing function f(x) consists of four-bit S-Box permutation lookups on each nibble of the word, followed by a linear mix. The fundamental block or round function of HB-2 encryption is defined as:

$$WD16(x, Ka, Kb, Kc, Kd)=f(f(f(f(x+Ka)+Kb)+Kc)+Kd) \quad (2.3)$$

Where x is the input plaintext, intermediate state, Ka, Kb, Kc, and Kd are four 16-bit secret keys and the nonlinear function f is specified as:

$$\begin{aligned} &S(x) = S1(x1) \mid S2(x2) \mid S3(x3) \mid S4(x4);\ \ x = (x1, x2, x3, x4) \\ &L(x) = x + (x <<< 6) + (x <<< 10) \\ &f(x) = L(S(x)) \end{aligned} \quad (2.4)$$

The Hummingbird-2 S-Boxes S1, S2, S3 and S4 are given Table 2-7.

Table 2-7 S-Boxes used in Hummingbird-2 [62]

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S_1(x)$	7	12	14	9	2	1	5	15	11	6	13	0	4	8	10	3
$S_2(x)$	4	10	1	6	8	15	7	12	3	0	14	13	5	9	11	2
$S_3(x)$	2	15	12	1	5	6	10	13	14	8	3	4	0	11	9	7
$S_4(x)$	15	4	5	8	9	7	2	1	10	3	0	14	6	12	13	11

For Initialization intermediate states are loaded with IVs and before starting encryption process, four round procedures in 80 clock cycles will be run. Hummingbird cipher is not reversible therefore for decryption part different hardware is required to be

implemented. The results of hardware implementation of HB-2 with area and performance optimization are shown in Table 2-8.

Table 2-8 Hardware implementations of Hummingbird-2 [62]

Cipher	Tech. [µm]	Cycles / Block	Datapath Width	Area [GE]	Init [Cycles]	Throughput (bits/cycle)	Power [µw] @10MHz
HB2-ee4c	0.13	4	16	3220	16	4	163.1
HB2-e16c	0.13	16	16	2332	80	1	156.8
HB2-e20c	0.13	20	16	2159	80	0.8	149.1

Authentication is provided in HB-2 cipher and it is optional. To generate the authenticated message no extra hardware is required and it can be generated by the same hardware used for encryption. The length of MAC is fixed 64 bits for plaintext with sizes of one word to eight words.

Security of HB-2 has been investigated in [63]. Authors of this paper have proposed an attack based on key recovery and differential sequence analysis (DSA) for HB-2. However, this attack is only of a theoretical interest and it does not affect the security of the Hummingbird-2 in practice. In [64], it has been proved that HB-2 cannot resist related key attack.

2.5 Conclusion

In this chapter, the most recent and well-known symmetric and asymmetric ciphers designed for low-cost RFID implementation have been studied. These ciphers cover new lightweight designs like PRESENT, Gain and HB and also adapted and modified version of contemporary cryptosystems like ECC. Asymmetric ciphers provide key-management advantages and non-repudiation service besides confidentiality. However, these ciphers are computationally far more demanding than symmetric ciphers in terms of performance, power

and area. This huge cost gap between these two types of ciphers makes asymmetric ciphers unsuitable for RFID systems while new designs in cryptography are directed towards symmetric ciphers.

In symmetric algorithms, block ciphers and stream ciphers are both competitive candidates for obtaining the name of lightweight cryptography. Block ciphers are well investigated and understood in security, while stream ciphers are better in cost. The lightweight primitives presented in this chapter are further compared and discussed in CHAPTER 5, together with our proposed cipher presented in CHAPTER 3.

CHAPTER 3: RBS Cryptosystem

RBS (Redundant Bit Security) is a lightweight authenticated symmetric encryption block cipher. The proposed algorithm is light in terms of area and power consumption which makes it suitable for restricted resources applications like RFID systems and sensor networks. Confidentiality of the plaintext in this algorithm is achieved by inserting some redundant bits inside the plaintext to change the location of plaintext bits while the order of them is unchanged in the ciphertext. Besides confidentiality, redundant bits provide authentication and integrity services as well. The location of redundant bits inside the ciphertext is a secret key shared between two parties. The security level of this algorithm is adjustable by the number of redundant bits. The hardware implemented RBS cipher requires less power and area compared to other known symmetric algorithms proposed for RFID systems while they provide just confidentiality service.

Typically, encryption algorithms are based on performing some complicated mathematical operations on the plaintext and ciphertext like multiplications and divisions which take plenty of resources. Unlike these conventional encryption methods, the proposed RBS algorithm in this thesis does not use these complicated mathematics computations for encryption and decryption processes. Instead, the message is intentionally manipulated by inserting redundant bits into original bits. In this algorithm, the location of the original bits changes in the ciphertext while their order will be unaffected. As an example, suppose that the original message is “1010”. Inserting one redundant bit at the third place changes the message to “10110”. Knowing that just one bit of the ciphertext is redundant, the attacker confronts with five possibilities to find the place of the redundant bit. Besides redundant bits’ locations, their values are important in providing confidentiality as well. For instance,

assume that the original bits are all zero. Therefore adding one ‘0’ bit as a redundant bit will not have the same effect as adding a “1” redundant bit in this case. Consequently, i) the number of redundant bits, ii) their locations and iii) their values are all important factors in hiding the plaintext inside the ciphertext. In other words, there is a relation between the security level of RBS algorithms and each of these parameters. In the following section, their effect on the security of the algorithm and the way to calculate them will be investigated. At the end of this chapter, hardware implementation of RBS algorithm will be studied.

3.1 Key and Number of Redundant Bits

The first parameter of RBS algorithm is the required number of redundant bits to provide the desired security level. Based on the necessary security level, at first, the number of redundant bits is calculated and hardware is implemented. Then all of communication between two parties will be encrypted base on it. So the number of redundant bits is a public parameter in this algorithm which is published among all parties while their location inside the key is secret between only sender and receiver.

The relation between the security levels of RBS algorithms is proportional with the number of redundant bits. Because, increasing the number of redundant bits grows the number of possible plaintexts for potential adversary which makes finding them in a ciphertext more complicated. On the other hand, the security level is defined by the size of key space. This definition makes a relation between the number of key space and the number of redundant bits.

3.1.1 Key Space

Security level is defined by answering this question that how long it will take for an attacker to break the algorithm based on what resources he needs in order to have a reasonable chance of succeeding. This cost, usually measured in time and/or money, for breaking the algorithm is required to be higher than the value of the protected asset. For example, if information is needed to be secure for only a few hours, one week effort for breaking of the system might be acceptable. One of the well-known tools for measuring the security level of an algorithm is key space.

Key space determines the set of all possible keys that can be used to initialize a cryptographic algorithm. The security level of an encryption algorithm has a direct relation with its key space. Suppose that n is the number of original bits or plaintext and m is the number of redundant bits. The ciphertext is an $(n+m)$-bit data obtained by insertion of redundant bits among plaintext bits. The location of redundant bits inside the ciphertext defines the secret key. Therefore, the secret key is simply an $(n+m)$-bit string where "1" in this string presents the location of redundant bit and "0" presents the location of plaintext bit in the ciphertext. For example, suppose that "10", "01" and "0110" are plaintext, redundant data and secret key respectively. So the first and last bits of the ciphertext belong to plaintext and other bits of the ciphertext will be replaced by redundant bits. For the sake of simplicity, suppose that the plaintext data appears directly in the ciphertext without any alteration. Under this assumption, the ciphertext would be "1010". However, in RBS algorithm the plaintext bits are altered before inserting them to the ciphertext which will be discussed later.

The number of possible keys, key space, is equal with counting all possibilities to select m distinct elements from a set of $n+m$ elements, called m-combinations. The size of key space (s) or the number of possible locations of redundant bits in the ciphertext, depends on n and m and is expressed by Equation (3.1).

$$s = \binom{n+m}{m} = \frac{(n+m)!}{m!\,n!} = \frac{\prod_{m=1}^{m}(n+m)}{m!} \tag{3.1}$$

In Equation (3.1), m and n are interchangeable. In other words, increasing either the number of redundant bits or the number of plaintext bits has the same effect on the key space size. As a result, fixing one of them, the size of key space can be adjusted to the desired security level by changing the other parameter. However, increasing these two factors while the other factor is decreasing will result in different way.

Figure 3-1 illustrates how the key space grows by changing the number of redundant bits and plaintext when the total number of bits is constant ($m+n$=100). It shows that when these two parameters are far from each other, the key space will reach to its minimum size. The maximum size of key space happens when the number of redundant bits is equal with the size of plaintext. Based on this graph, it can be concluded that high security level for small block of plaintexts is not possible unless with huge number of redundant bits. Also, it is not possible to provide high security level with low number of redundant bits. The best choice is selecting these two parameters close together to obtain high available security.

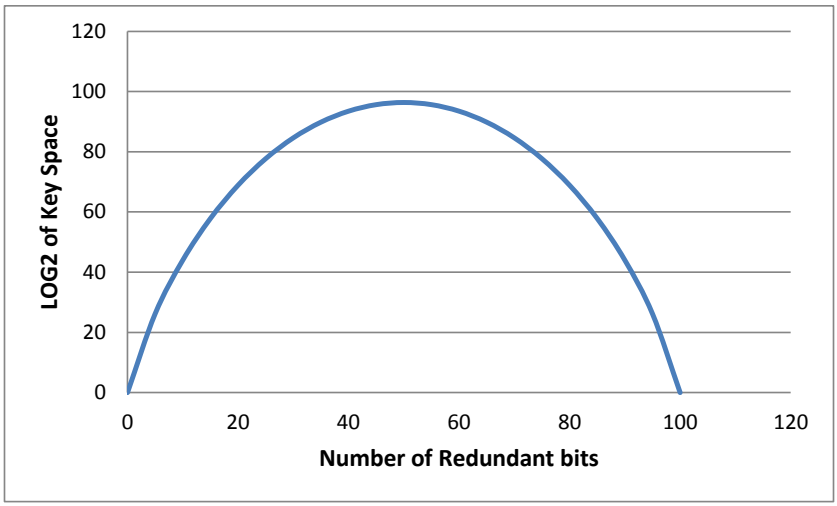


Figure 3-1 Changing the size of key space with the number of redundant bits

To find the optimum number of redundant bits, the size of key space is calculated when the number of redundant bits and the size of plaintext are equal. It is the situation that key space is in its maximum size. Figure 3-2 exhibits how big the key space can be for different size of plaintexts when the same size of redundant bits is merged with it.

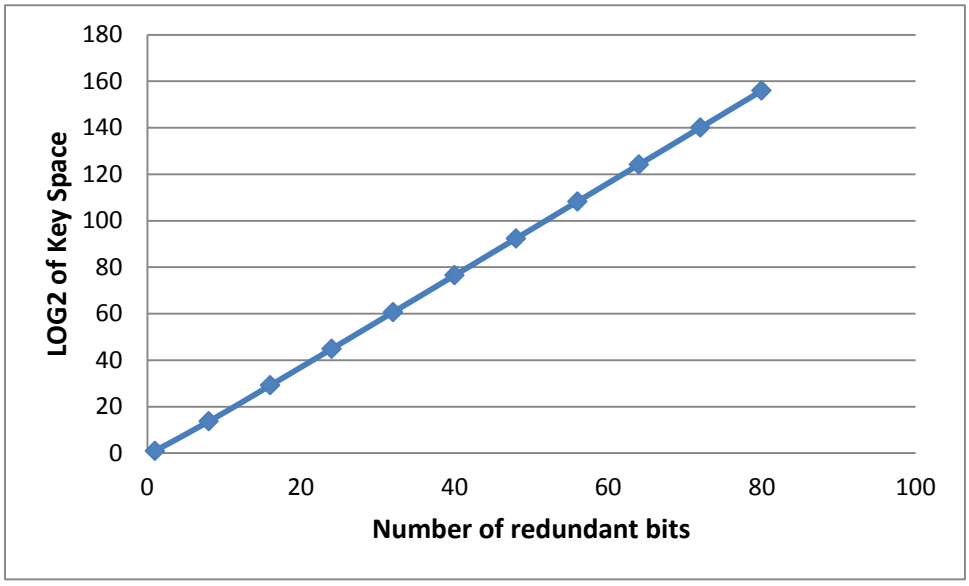


Figure 3-2 Size of key space when the number of redundant bits is equal with plaintext bits

The start point for this study is selected to be 64-bit block plaintext. Figure 3-3 demonstrates the relation between s and m for (n=64). Increasing m from 0 to 128; the key space (s) will exponentially increase from 1 to 2^{172}.

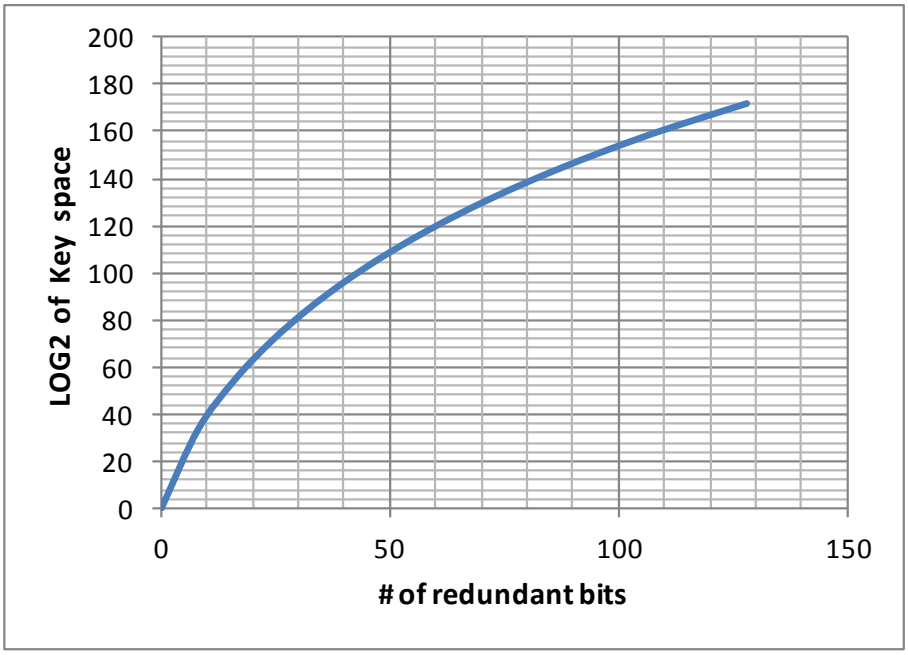


Figure 3-3 Key space growth while plaintext size is fixed (64 bits)

As mentioned before, there is a relation between the size of the key space of an encryption algorithm and its security level against possible attacks. The question is how big the space key should be to guarantee the desired security. The Brute-Force attack has been studied for finding the boundary of the key space for RBS algorithm. In this attack the attacker performs a complete search through all possible keys of the key space to find the right key. The 2^{128} key space size is computationally secure against Brute-Force attack. Applying this number in Equation (3.1), there will be a variety of choices for m and n.

Table 3-1 shows a possible set of m ad n for s=2^{128}. One of limitation factors which makes the span of choices narrow is the size of ciphertext. Considering this fact that the

required energy for transmitting the message increases by the length of ciphertext, *m* and *n* should be chosen such that the number of bits of ciphertext becomes least. Referring to Table 3-1, the minimum size of ciphertext is 132 bits and this happens when $(n,m) \in \{(64,68), (65,67), (66,66), (67,65), (68,64)\}$ which are highlighted in Table 3-1.

Table 3-1 The number of bits required in ciphertext to have $s=2^{128}$

m	50	55	57	60	63	64	65	66	67	68	70	73	78	91
n	91	81	78	73	70	68	67	66	65	64	63	60	57	50
*c**	140	136	135	133	133	132	132	132	132	132	133	133	135	140
c :* *# of bits in the ciphertext*														

For two reasons, the best choice is (64, 68) when the size of plaintext is 64 bits with 68 bits redundant bits. First, data blocks are processed and stored normally in multiples of 8-bits. While redundant bits are used only in this cipher so they do not need to be a multiple of eight. Second, the number of redundant bits is required to be more than plaintext bits for few bits, to prevent collision which will be discussed later in this chapter.

Based on the application requirement, the strength of security may change. The designer can change the number of plaintext and redundant bits to reach his desired security level. Performing the same simulation steps for $s=2^{128}$, the recommended number of redundant bits for different key spaces has been acquired as shown in Table 3-2. Compared to other cryptosystems, RBS algorithm needs 3 to 4 bits more in key size to provide the same security level that they support.

In Table 3-2, the size of plaintext varies between 40 to 64 bits to provide key space between 2^{80} to 2^{128}. However, the designer has this ability to play with the number of plaintext and redundant bits to reach his arbitrary security level. For example, to have 2^{80} key

space when the plaintext size is 32 bits, the designer can either use 56 redundant bits or RBS-83 while the first choice imposes 88 bits in the ciphertext which is 5 bits longer than the ciphertext in RBS-83.

Table 3-2 Number of required redundant bits for different security level

	Size of Key space	Size of plaintext	Number of redundant bits	Size of key & ciphertext
RBS-83	2^{80}	40	43	83
RBS-100	2^{96}	48	52	100
RBS-116	2^{112}	56	60	116
RBS-132	2^{128}	64	68	132
RBS-197	2^{192}	96	101	197
RBS-262	2^{256}	128	133	261

Using plaintexts shorter than 40 bits reduces the security level of RBS cipher sharply unless the number of redundant bits grows dramatically. Using plaintexts longer than 64 bits increases the security level more than it is necessary. In both cases, the length of the ciphertext becomes very long for transmission. On contrary, 2^{128} key space can be obtained by 128-bit plaintext along with 40-bit redundant. Thus, the length of the ciphertext will be shorter than when RBS-132 is utilized for two 64-bit plaintexts. However, this design is not acceptable since the number of redundant bits is less than plaintext and it cannot guarantee exclusive redundant data for each plaintext. On the other hand, the hardware overhead will be significantly high for this design. Thus, it is recommended that the size of plaintext is limited between 40 to 64 bits based on the desired security level. So, for plaintext shorter than 40 bits, RBS-83 is a decent choice and for longer 64 bits, the plaintext will be broke down into suitable sizes.

3.1.2 Flexibility in Security Level

Supporting different key sizes with the same hardware is one of the advantages of RBS algorithm. The number of plaintext and redundant bits in the ciphertext are two important parameters in defining the security level of RBS. Playing with these two parameters gives this ability to the designer to change the security level of the cipher to his desired level. So before hardware implementation, the optimum cipher required for the given key size can be designed off-line while it may not be easy for some other block ciphers which use pre-defined key and data block sizes.

After implementation, the designer has still this ability to change security level of RBS cipher on-line by using different key and data block sizes while the number of redundant bits in the key is same as before. The only part of RBS hardware which is fixed and cannot be updated with the security level is the MAC generator. Therefore, the number of redundant bits is constant in different key sizes and the security level can be adjusted only by the number of plaintext bits while the same MAC generator can be used without any change in it. For example, if there are 68 bits for redundant data, by changing the size of plaintext from 32 to 64 bits the key space can vary from 2^{86} to 2^{128}. Therefore, different tags with different key sizes can be supported with the same hardware. By using this feature, it will not be required to replace tags whenever the security level of the system changes. The only restriction in key flexibility is the size of redundant bits which should be longer than plaintext at least for few bits to avoid collision.

3.2 Location of Redundant Bits

The second significant parameter in providing the security of RBS algorithm is the location of redundant bits inside the ciphertext. This information is required to keep as a secret key among involved parties to have a secure communication. Revealing the location of any redundant bits in the ciphertext will diminish the size of key space.

The distribution of redundant bits inside the ciphertext is also another important factor in providing confidentiality. This distribution should not follow any linear or non-linear mathematic function, otherwise i) the size of the key space will be reduced, ii) a dependency will be constructed among redundant bits and iii) redundant bits will be distributed uniformly among plaintext bits. Therefore, the position of every redundant bit must be independent of other bits' positions. This way, if the location of one of the redundant bits being exposed just the key space will shrink while location of other redundant bits is still secret.

The best solution is utilizing random distribution of redundant bit inside the ciphertext. This type of distribution can be defined by a user or one pseudo random number generator (PRNG) with this condition that the number of one and zero in the secret key are constant in the secret key.

3.3 Value of Redundant Bits

In addition to providing confidentiality of the sent data, the injected redundant bits can carry some additional information about the original data as well. For generating these redundant bits, there are three options:

- Choosing constant values. In this case, the redundant bits are the same for different plaintexts. This way the attacker can easily figure out the location of the redundant bits just by comparing ciphertexts of two or more different plaintexts.
- Choosing random values. In this case there would be several ciphertexts for one plaintext. This way the attacker can again easily figure out the location of the redundant bits by comparing the different generated ciphertexts for the same plaintext.
- Values of redundant bits are injective functions of the plaintext. So, there is an exclusive redundant data per each plaintext. As a result, plaintext and redundant data cannot be distinguished easily in the ciphertext.

Among these three approaches, the third option is suitable as it has potential to provide both attack prevention and authentication. This algorithm can be implemented by splitting the plaintext into small blocks and performing mathematical functions on each of blocks individually. At the end, all blocks are combined while it is encrypted by a secret key. The pseudo program of this algorithm is presented in figure 3-4.

```
Split the plaintext into several small segments Si.
for each Si
        Shift/rotate/add/XOR (Si , a constant number Ni)
Combine all segments Si to a single segment S
Encrypt (S, secret key K) using a symmetric algorithm

//K will be used in the receiver side for authenticating the sender
```

Figure 3-4 Redundant data generation algorithm

One applicable implementation solution for the presented algorithm in figure 3-4 is Message Authentication Code (MAC) algorithm because a very small change in the plaintext

will produce an entire different output. Using MAC algorithm for generating the redundant bits, integrity and authentication will be provided as well as confidentiality which will be discussed in the following subsection.

3.3.1 Message Authentication and Data Integrity

Data integrity is defined as maintaining correctness and consistency of a message. Since the message is sent via wireless network, the integrity of the message is always in danger of being altered in transmission by either an adversary or environmental hazards, such as heat, dust, and electrical surges. Therefore, the receiver should validate the received data.

Message authentication is one of other cryptography services which guarantee that the received message has been sent by an eligible user. It is crucial for a party – tag or reader - which receives a message to make sure who sent it. Message Authentication Code or MAC is a piece of information which is used for both data integrity and authentication purposes. It is generated by a MAC algorithm which has two inputs i) an arbitrary-length message and ii) a shared key between two parties (Figure 3-5). Typical MAC algorithms are strong in terms of security and some of them also guarantee that no collusion will happen in their outputs for different input messages.

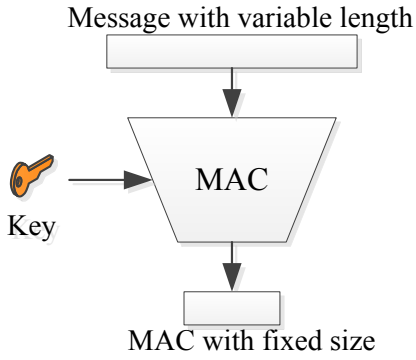
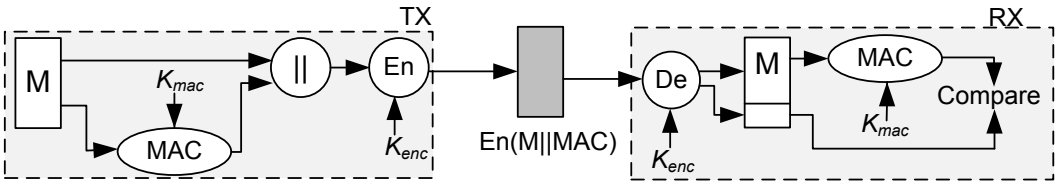


Figure 3-5 MAC algorithm

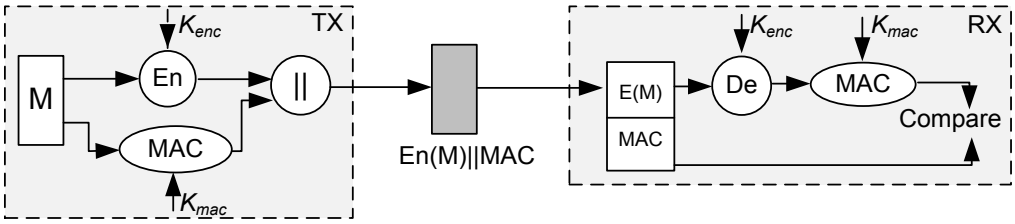
In general, there are three contemporary protocols for embedding the MAC inside the ciphertext (Figure 3-6).

- The first protocol is more common than other protocols in symmetric key encryption algorithms. After generating the MAC by the authentication key, *Kmac*, in this protocol, MAC will be attached to the original message and then the new message is encrypted by encryption key, *Kenc* (Figure 3-6.a) [65]. In the receiver side, after decrypting the received data, the MAC will be regenerated and then compared with the received one. Generating the same MAC means that the message is intact and it is sent by an authorized party. Otherwise the received message will be discarded.
- The generated MAC will be generated and attached to the end of the ciphertext before transmission (Figure 3-6.b). Grain [55] and HB-2 [62] both use this protocol for providing authentication.
- Instead of the plaintext, the MAC of the encrypted plaintext is attached to encrypted message before transmission (Figure 3-6.c). This protocol requires more time than other two protocols since MAC and encryption cannot happen at the same time.

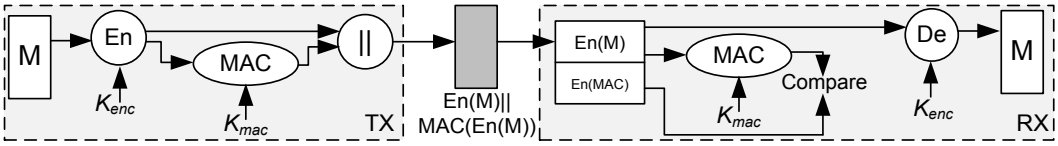
In the second and third protocols, the boundary between the MAC and message is clear. Hence, the MAC algorithms used in these protocols are required to be very secure against the substitution attack. In this attack, the adversary tries to replace the legitimate message with his own plaintext and MAC and assumes that it will be accepted by the receiver. Considering this fact, the first protocol seems more secure against substitution attack because MAC is encrypted along with the plaintext and there is no direct access to it.



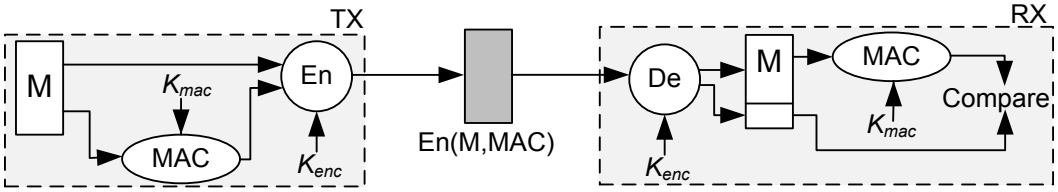
(a) First authentication protocol



(b)Second authentication protocol



(c)Third authentication protocol



(d)Proposed authentication protocol

Figure 3-6 Embedding tag inside the ciphertext in different protocols

3.3.2 Message Authentication and Redundant Bits

As stated earlier, redundant data is generated by MAC algorithm. The second and third MAC generation protocols cannot be used in RBS algorithm as the MAC is attached to the end of the message which is the redundant part in RBS. The method used in RBS is based on the first protocol with a slight modification (Figure 3-6.d). It might be noticed that second and third protocols can be special cases of RBS algorithm when all of redundant bits are

located at the end of the ciphertext. In other words, if *m* is number of redundant bits, the most *m* significant bits of the secret key are one while the rest of the key bits are zero.

In RBS algorithm, the generated MAC as redundant bits is inserted among message bits instead of being appended to the end of the message. In other words, merging the MAC with plaintext is a part of encryption process. The distribution pattern of the MAC part inside the ciphertext is based on the encryption key. At the receiver side, the received data is broken into two parts based on the encryption key: the altered plaintext and redundant bits. Regenerating the MAC of received plaintext at the receiver side and comparing it with the received MAC, the receiver decides to keep the data or discard it.

3.4 Plaintext Manipulation

How the plaintext appears inside the ciphertext is the last significant parameter in RBS algorithm which is directly related to the confidentiality of this algorithm. Three possible scenarios for this issue will be discussed in the following subsections.

3.4.1 Direct Appearance inside the Ciphertext

In this approach, the original plaintext bits will be merged with redundant bits without any change on the plaintext. So in decryption process, the plaintext can easily be extracted from the ciphertext by removing the redundant bits. Despite the simplicity of this method, the key space may shrink sharply which makes the algorithm vulnerable to some attacks like known plaintext attack and chosen plaintext attack. In these attacks, since the attacker knows the plaintext, those bits of ciphertext which have the same value of the plaintext will be potential locations for plaintext bits in the secret key. For example, if the plaintext is all zero, all corresponding zero in the ciphertext might be location of plaintext in the secret key too.

There are some ways to expand the key space size such as increasing the number of redundant bits or having separate encryption keys based on the plaintext pattern. Increasing the number of redundant bits introduces more area overhead on MAC implementation and so more power for transmitting the ciphertext. Generating a new key based on the plaintext pattern and exchanging it are also challenging tasks in symmetric encryption algorithms. Based on the stated reasons, this approach is not interesting to be used in RBS algorithm.

3.4.2 Bitwise Addition with a Constant-Value Keystream

In this approach, regardless of the pattern of the plaintext, some bits of the plaintext will always be altered in the ciphertext. The location of these plaintext bits is fixed. Thus, always some bits of plaintext will appear altered while other bits are not changed. This approach makes the algorithm secure against known plaintext attack because the attacker does not know which bits of plaintext are altered in the ciphertext.

As a solution for this approach, suppose that the secret key, K_{enc} is a binary sequence such that $K_{enc} = \{k_0, k_1 \ldots, k_{n+m}\}$ where n is the size of plaintext and m is the number of redundant bits. This key will be broken into two binary sequences K_{enc1} and K_{enc2} such that:

$$K_{enc1} = \{k_0, k_1 \ldots, k_i\}, K_{enc2} = \{k_{i+1}, k_{i+2}, \ldots, k_{n+m}\} \text{ where } i=(n+m)/2 \quad (3.2)$$

Then, the plaintext will be XOR-ed by $K'_{enc} = K_{enc1}\ XOR\ K_{enc2}$. This way, some bits of the plaintext will be altered depending on the value of K_{enc}. The number of altered bits is varying between zero bit (when $K_{enc1} = K_{enc2}$) to n (when $K_{enc1} = \sim K_{enc2}$). Since the attacker does not know the encryption key, he does not know how many bits of the plaintext and which of them in the ciphertext have been altered. This way, the attacker confronts with 2^m

different possible manipulated plaintexts. However, this solution is still vulnerable to chosen plaintext attack. Suppose that, the attacker changes just one bit of the plaintext, redundant bits (MAC) in the ciphertext will change while all bits of the plaintext, except one will be same. Comparing these two ciphertexts which have almost same plaintexts will shrink the key space dramatically and make it easy for the attacker to find approximate location of the plaintext in the ciphertext. Using this approach is not promising due to its weakness against attacks.

3.4.3 Bitwise Addition with Variable-Value Keystream

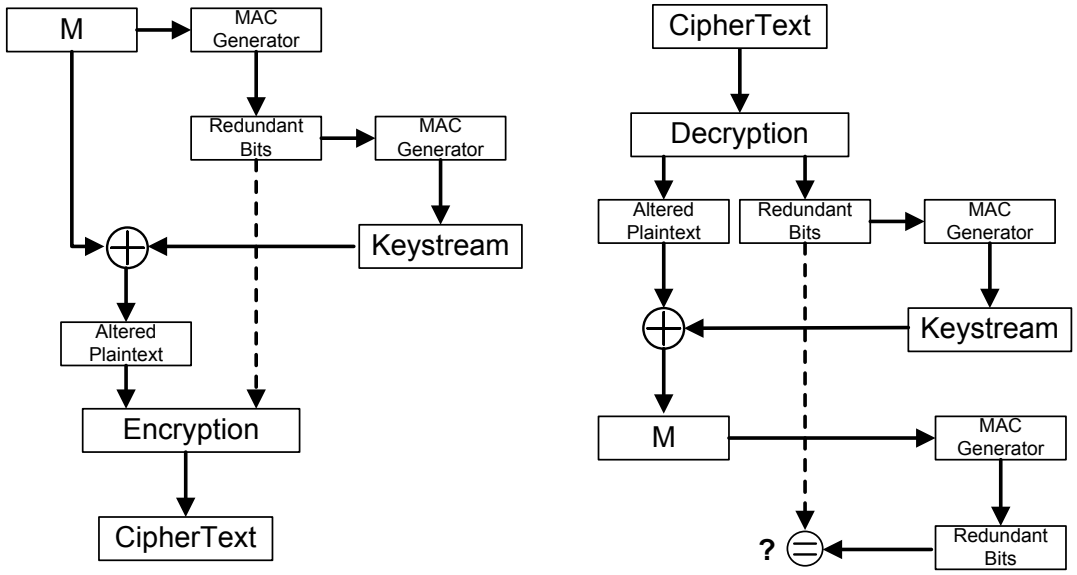
In this approach, the plaintext bits are altered by performing bitwise addition on the plaintext with a variable keystream. This keystream will be different for each plaintext and it is unique for each plaintext so for the same plaintexts the same ciphertexts will be generated. To satisfy this condition, the keystream is required to be a function of the plaintext. This way, if any change happens in the plaintext, different keystreams and consequently different ciphertexts will be produced.

This solution is somehow similar to one-time pad because in both methods the plaintext is altered with a variable keystream. However, it is different since in one-time pad the keystream is a random generated number so for two same plaintexts different keystreams will be resulted and the keystream is independent of plaintext while in this proposed solution there is a dependency between the plaintext and the generated keystream.

Variation in the value of the keystream based on the ciphertext will eliminate the weaknesses caused in the last approaches since for each plaintext the number of altered bits

and their locations will be different from any other plaintext. This feature encourages us to use this approach for manipulating the plaintext in the ciphertext.

One straightforward mean for implementing this approach is through MAC function. For the sake of hardware sharing, the same hardware for generating redundant data can be used for altering the plaintext. However, the MAC of the plaintext has already been used as redundant data, so it cannot be used again as a keystream. Otherwise, it creates dependency between altered plaintext and redundant data which makes the algorithm vulnerable to some attacks such as chosen plaintext attack. Instead of using the MAC(Plaintext) as a keystream, the MAC(Redundant data) or more precisely MAC(MAC(Plaintext)) is used for generating the keystream in RBS algorithm as illustrated in Figure 3-7.a.



(a) RBS Encryption *(b) RBS Decryption*

Figure 3-7 Block diagram of encryption and decryption

As Figure 3-7.a shows, the altered plaintext is obtained by performing bitwise addition on the plaintext with the generated keystream. Eventually, the ciphertext will be

produced by merging the altered plaintext with redundant data based on the secret key. Since the length of keystream is not equal to the length of redundant data – it is three to four bits longer than the length of plaintext data - only the *n* least significant bits of the keystream (*n* is the size of plaintext) will be used to generate the altered plaintext. The decryption process is illustrated in Figure 3-7.b. Since the receiver side extracts the redundant part from the ciphertext with the secret key. Afterwards, the keystream will be generated through MAC of the redundant data which then will be used for recovering the original plaintext.

Having the same process for encryption and decryption is the one of the main advantages of RBS algorithm which makes the same hardware implementation can be used for both processes. This characteristic which has already been studied in stream ciphers will cause significant saving in the area.

3.5 Implementation

The hardware implementation of RBS is composed of three main parts: MAC generator, encryption part, and decryption part which will be discussed in detail in the following subsections.

3.5.1 MAC Generator

The MAC generator circuit is responsible for generating MAC of the plaintext utilized as redundant bits in the ciphertext and also MAC(MAC(P)) or Mac of redundant bits used as the keystream.

The implementation complexity of the MAC generator depends on its underlying MAC algorithm. There are several categories of MAC algorithms. A number of them use

block cipher operations such as OMAC, CBC-MAC, and PMAC. The CBC-MAC comes in different versions varying in details such as padding, length variability and key search strengthening [66].

HMAC is another type of MACs which is based on iterating a hash function. The cryptographic strength of the HMAC depends on the cryptographic strength of the underlying hash function, the size of its hash output, and the size of the key. The size of the output of HMAC is equal with the size of the output of used underlying hash function.

Implementing MACs by universal hash functions has been interested because it is shown to be secure and well adapted for hardware implementation [67]. VMAC is a block cipher-based MAC algorithm which is using a universal hash. VMAC has excellent performance in software implementation. The length of VMAC output is a multiple of 64-bit up to the block size of the block cipher in use. UMAC is a type of MACs based on universal hashing which is calculated by choosing a hash function from a class of hash functions.

Table 3-3 Summary of MAC algorithms

Algorithm	Output size	Comments
CBC-MAC	Varies in versions	Based on block ciphers
HMAC	Fixed depends on used hash function	Based on hash functions
VMAC	Multiple of 64 bits	Based on block cipher and universal hash, Good in software
UMAC	Support different sizes	Based on universal hash and choosing hash function randomly

One of the main factors in selecting a MAC algorithm in RBS, is the length of generated MAC which is usually fixed and is defined as a parameter of the algorithm. For instance, SHA-0 and MD5 generate 160-bit and 128-bit MACs respectively. In other words,

the size of MAC depends on the chosen MAC algorithm. This limitation can be resolved if the chosen MAC algorithm supports variable-length MAC.

3.5.2 Chosen MAC Algorithm for RBS

The MAC algorithm presented in [56] is a family of universal hash functions. This MAC is selected to be utilized in RBS algorithm for two reasons. First, the output of this MAC can be set before implementation to generate arbitrary size MACs. Second, it is a light-weight universal hash algorithm from ϵ-almost XOR universal (ϵ-AXU) family based on Toeplitz matrices. The security of this algorithm is promised by low probability of exact substitution. Besides its resistance against collision attack is high.

Based on the definition, if H is a family set of hash functions mapping from set A to set B by (H,A,B), then (H,A,B) is ϵ-AXU if $\forall$ x, x′ $\in$ A, x $\neq$ x′, y $\in$ B,

$$| \{h \in H : h(x) \oplus h(x') = y\} | \leq \epsilon \cdot |H|. \tag{3.3}$$

Constructing a MAC using an ϵ-AXU family, one part of the key is used to select a function from h ∈ H and the output of this function is XOR-ed with a second part of the key, used as a one-time pad, chosen randomly from B. The MAC algorithm offered in [56] is constructed based on Toeplitz matrices by assuming that k_i, $i = -w, 1 - w, \ldots, L - 2$ is a sequence of randomly chosen key bits. Then if $\boldsymbol{t} = (t_0, \ldots, t_{w-1})$ is a bit-vector of tag or digest with length of w and $\boldsymbol{m} = (m_0, \ldots, m_{L-1})$ a bit-vector of message with length of L, a possible MAC construction will be:

$$\begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_{w-1} \end{bmatrix} = \begin{bmatrix} k_{-1} & \cdots & k_{L-2} \\ k_{-2} & \cdots & k_{L-3} \\ \vdots & \ddots & \vdots \\ k_{-w} & \cdots & k_{1-L-w} \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{L-1} \end{bmatrix} \tag{3.4}$$

From algorithmic interpretation point of view, the output of MAC or tag is initialized $\boldsymbol{t} \leftarrow 0$. Introducing a window of size w to form:

$$\boldsymbol{K}_0 = [k_{-w} \ldots k_{-1}], \ldots, \boldsymbol{K}_{L-1} = [k_{L-1-w}, \ldots k_{L-2}] \tag{3.5}$$

For each bit m_i, if it is zero nothing will be happen and if it is one the tag will be updated by $\boldsymbol{t} \leftarrow \boldsymbol{t} \oplus \boldsymbol{K}_i$. Figure 3-8 shows the block diagram of its implementation which is constructed of three parts: a linear feedback shift register (LFSR), a non-linear shift register (NFSR), and an accumulator to keep the output.

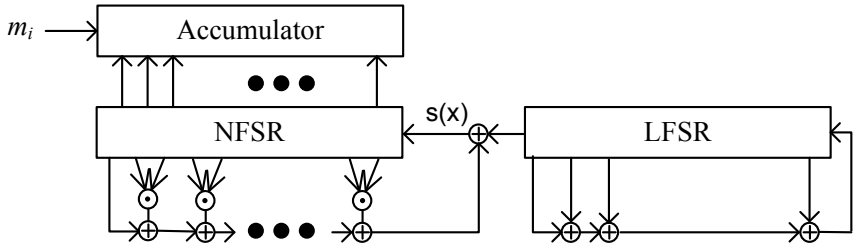


Figure 3-8 The hardware suggested for MAC generation in [56]

The present state of LFSR is a linear function of its previous state. The generated LFSR sequence will be fed into the NFSR. In NFSR, the present state is a non-linear function of its previous state. This non-linear function is composed of some linear function and bent function together. LFSR and NFSR together build up a pseudo random number generator (PRNG). The output of this PRNG, s(x), is the result of performing bitwise addition of LFSR with the result of NFSR function which feeds back into the NFSR as an input. This output is

dependent on the initialized value of these two registers. Therefore by any change at the initialization value, the generated output sequence, s(x) will be different. The value of NFSR updates the accumulator. The accumulator is a register whose bits are XOR-ed by NFSR's value if (m_i = 1). The input m_i, is the input message which is checked by accumulator bit by bit.

At the beginning of the process of MAC generation, LFSR and NFSR are initialized with the authentication key while the accumulator is set to be zero. After initializing registers, the message will be entered bit by bit at each clock cycle. If the input bit is one then the accumulator will be XOR-ed with the content of NFSR. Otherwise, nothing will be happen. Both LFSR and NFSR registers will be updated at each clock cycle. This process repeats until all bits of the message are checked by the accumulator. Therefore, the time required to generate the MAC is dependent on the length of the message and takes *m* clock cycles where *m* is the length of the message. The algorithm is presented in Figure 3-9 in a pseudo code.

Initialize	{NFSR, LFSR} = authentication key Accumulator = 0
Process	for every m_i in the message m repeat if (m_i = 1) Accumulator = Accumulator ^ NFSR
Result	Accumulator contains the MAC code

Figure 3-9 Authentication algorithm

The MAC algorithm in Figure 3-9 has a weakness when the message is a zero string which generates zero as MAC as well. To overcome this flaw, a one-bit pad with value of one is deliberately appended to the end of the message and then this message is applied to the MAC algorithm.

Preventing collision at the MAC is very important in security strength of RBS cipher. If two or more different plaintexts have the same MAC as redundant bits then they will have the same keystream. Thus, attacker can find some locations of altered plaintext by performing bitwise addition on the ciphertexts of their plaintexts. However, the generated keystream is not required to be collision free. Since two plaintexts are not same, by performing the bitwise addition on them with plaintexts, two different altered plaintexts will be resulted.

Hash collision probability for universal hash function is proved to be equal or less than the bias, ϵ, if the key is refreshed after each communication [68]. If L is the length of the input plaintext and w is the length of the NFSR, then bias ϵ is defined for the MAC algorithm in [56] as:

$$\epsilon = L/2^w \tag{3.6}$$

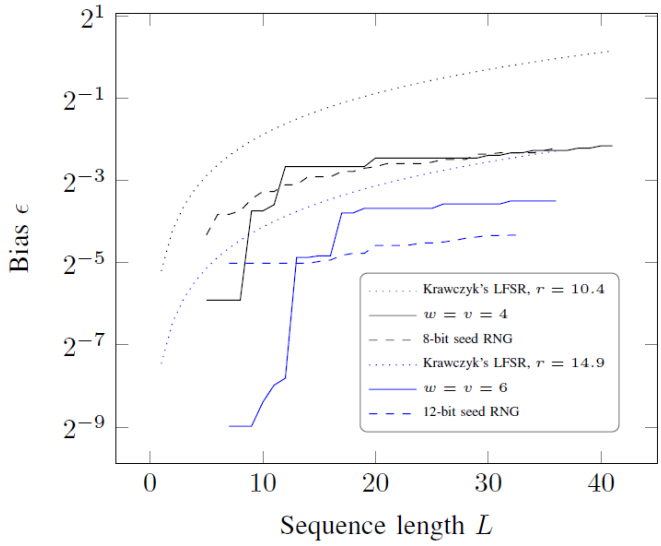


Figure 3-10 The bias as it develops for growing sequence lengths adapted from [56]

However, the experimental results show that the calculated bias is lower than the calculated results by equation 3.6 and this equation gives only an upper bound of the bias. In figure 3-10, two different tag sizes *w* have been studied in [56] and their respective biases are plotted with solid lines. The dotted lines give the biases for the LFSR construction using equal amounts of randomness. Dashed lines show the behavior of random number generators. Lower curves give lower biases which offer lower probability of happening collision. As this figure shows, the obtained biases are lower than expectation calculated by equation 3.6. For example, when w =6 and L=32, the obtained bias is less than 2^{-3} which is far less than the calculated bias, $\epsilon=32/2^6= 2^{-1}$.

Based on the equation 3.6, the probability of collision for RBS-132 is 2^{-62}. However, based on the experimental results for shorter plaintexts, it is expected that this probability will be lower than 2^{-62} which guarantees that happening collision for redundant data will be very low and close to zero.

3.5.3 Adapting the Chosen MAC with RBS

The first step for adapting this authentication algorithm with RBS cipher is defining the size of NFSR, LFSR and accumulator registers and also the authentication key. The size of the accumulator and the NFSR are equal to the length of the MAC. Since the length of redundant data in RBS is *m* bits, two *m*-bit registers are reserved as NFSR and Accumulator. Equations 3.7 to 3.10 presents the proposed NFSR functions for different designs of RBS denoted as *f(x)*.

RBS-83:

$$f(x) = 1 + x^{11} + x^{24} + x^{34} + x^{43} + x^{14}x^{19} + x^{20}x^{42} + x^{26}x^{29} + x^{37}x^{38} \quad (3.7)$$

RBS-100:

$$f(x) = 1 + x^{13} + x^{15} + x^{29} + x^{41} + x^{52} + x^{17}x^{24} + x^{25}x^{50} + x^{26}x^{27} + x^{32}x^{35} + x^{44}x^{45} \quad (3.8)$$

RBS-116:

$$f(x) = 1 + x^{15} + x^{17} + x^{33} + x^{47} + x^{60} + x^{20}x^{27} + x^{28}x^{58} + x^{29}x^{31} + x^{37}x^{41} + x^{52}x^{53} \quad (3.9)$$

RBS-132:

$$f(x) = 1 + x^{8} + x^{25} + x^{38} + x^{64} + x^{68} + x^{5}x^{14} + x^{20}x^{30} + x^{34}x^{41} + x^{46}x^{54} + x^{51}x^{60} \quad (3.10)$$

This MAC algorithm is inherently designed for stream ciphers and LFSR plays a major role in this process because its present state will be referred for refreshing the authentication key in the next communication step. Since RBS is block cipher algorithm and it uses fixed authentication key for each communication step, so keeping LFSR register is not required anymore. However, the LFSR key is required in generating pseudo-random numbers. Therefore, the LFSR key enters to NFSR register bit by bit.

In order to have the same key for both authentication and encryption, the size of LFSR key is defined to be *n* bits which combined with *m*-bit NFSR key forms a *n*+*m*-bit key (Equation 3.11). Before applying the authentication key to the MAC generator, this key will be initialized once when the cipher starts up or the key changes. For initialization the key is loaded to the NFSR while the input message remains zero during the process. The result after 2*m* clock cycles will be ready in the NFSR and it will be kept as a NFSR key.

$$Kauthentication = \{KLFSR, KNFSR\}$$
$$K_{LFSR}=\{K_0,\ldots,K_{n-1}\}, K_{NFSR}=\{K_n,\ldots,K_{n+m-1}\} \quad (3.11)$$

Universal hash functions are guaranteed to be collision free if the key is refreshed after each usage [24]. To refresh the authentication key the authentication key must be unique per each message. In other words, there must be unique K_{NFSR} and K_{LFSR} per each message. In order to refresh these keys in RBS algorithm, they are defined as a function of initial key and plaintext data. One straightforward solution is through performing bitwise addition on plaintext and initial key value for generating authentication key per each plaintext as illustrated in Figure 3-11. The LFSR key is generated by performing bitwise addition on the plaintext bits, m_i and the LFSR key k_i.

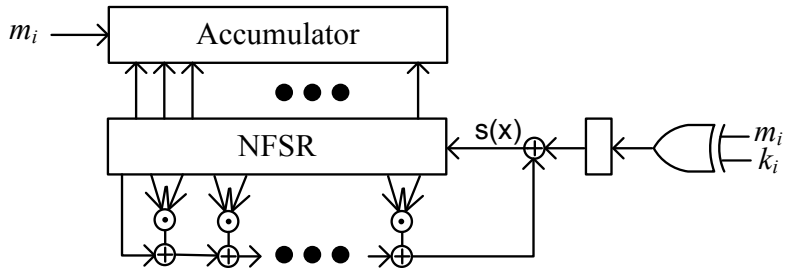


Figure 3-11 Adapted MAC generator for RBS

At this phase of generating the keystream, the size of LFSR key is less than the size of input. Therefore, the LFSR key is repeated from the beginning when it reaches to end to support generating the sequence of s(x).

To prevent having zero as the MAC, the initialized value of both NFSR and LFSR registers are required to be non-zero. To guarantee that their initialized value will not be zero for any message, the register placed after XOR of LFSR key and message is initialized with one. Thus, at the first clock cycle if all bits of NFSR are zero, one bit "1" will be generated as

the first bit of s(x) and entered to NFSR. Therefore, there is at least one bit "1" at NFSR register which prevents generating zero as a MAC.

Generating the redundant bits takes n+1 clock cycles since the size of the input of MAC is n-bit plaintext plus one bit padding. Generating the keystream requires m clock cycles because the length of the redundant bits as an input to MAC is m bits without any padding. To generate the keystream, padding is not necessary since the value of redundant bits is always nonzero.

3.5.4 Encryption

The encryption process completes in two phases. In the first phase, the plaintext is altered through bitwise addition with the keystream. For the sake of area, the MAC generator circuit (Figure 3-11) is used for altering the plaintext as well. This way, NFSR and accumulator are loaded with the keystream and the message respectively while the input m_i is set to be one. The altered plaintext will be generated and stored in accumulator in just one clock cycle.

In the second phase, the altered message is merged with redundant data based on the secret key during data transmission. Figure 3-12 illustrates the process, where the altered plaintext bit (p_i), redundant bit (r_j), and encryption key bit (k_l) entering the cipher bit by bit. Depending on the value of key (k_l) either p_i or r_j will be transmitted.

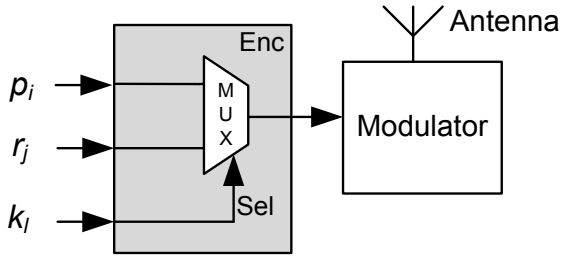


Figure 3-12 Encryption module in transmission part

3.5.5 Decryption

The decryption process completes in three phases. In the first phase, redundant bits and the altered plaintext will be extracted from the received ciphertext (Figure 3-13). Receiving data from antenna and demodulating it, the received bit will be considered as either the altered plaintext bit, $\not{p}_i$, or the redundant bit, $\not{r}_j$, based on the value of the key, k_l. These bits are shifted to their corresponding registers as they are received.

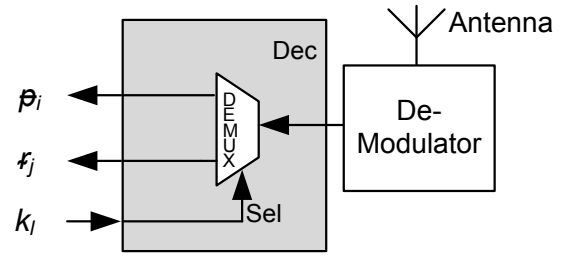


Figure 3-13 Extracting altered plaintext and redundant data from ciphertext

In the second phase, the keystream will be regenerated using extracted redundant bits and the key. Performing bitwise addition on the regenerated keystream and the altered plaintext data, the original plaintext will be recovered as illustrated in Figure 3-7.b.

In the last phase, the redundant data is regenerated by calculating the MAC of the recovered plaintext as illustrated in Figure 3-7.b. Comparing the received redundant data with the regenerated redundant data will authenticate the received message. In case of failure

in the authentication process, the decryption part returns a string of 0's as decrypted message. This way, the algorithm would be secure against chosen ciphertext attack which will be discussed in the next chapter.

3.5.6 Reception / Transmission

Figure 3-14 displays the encryption and decryption parts together along with reception and transmission data. Since the system is half-duplex, reception and transmission will not happen at the same time.

The En/De signal determines which process is being performed now, either encryption or decryption. Reception/transmission part is composed of a counter, multiplexer and two registers. Registers store the sent or received message during transmission or reception. Here there are two registers, one for altered plaintext and the other for redundant data. The counter keeps the number of bits required to be shifted to encryption module or from decryption module which is $n+m$ bits at the beginning of the each process. Since the total number of shifted bits is always fixed and shifting them is controlled with the secret key, there is no need to have separate counters to keep the number of shifted plaintext and redundant bits. The multiplexer is responsible to select which register is being shifted now based on the secret key. This multiplexer is designed such that it is active as long as the counter is working. Sending and receiving messages serially is an essential part of each RFID tag and is not designated for only this algorithm. Therefore this part is not counted in calculating the experimental results like area and power consumption except the ciphers and multiplexer.

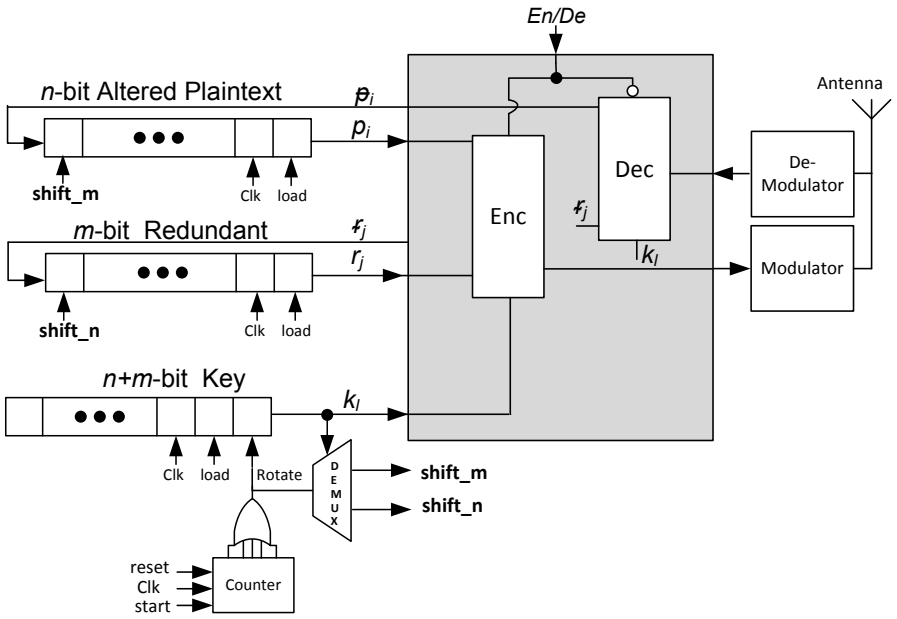


Figure 3-14 Cipher plus transmitter and receiver

To send out the encrypted message, at first, the counter will be initialized with the total number of bits required to send which is $n+m$ in RBS algorithm. Then based on the key, the least significant of the corresponding register will be shifted to encryption module. From this module the corresponding bit will be sent to modulator to be transmitted. This process continues until the counter becomes zero. At this time, all of altered plaintext and redundant bits are sent to air.

To receive the encrypted message, the same process will be followed with this difference that the received bits, based on the secret key, will be shifted to its own corresponding register. At the end of the process, all received bits are separated by decryption module and stored in their registers.

The transmission and reception algorithms are shown with pseudo code in Figure 3-15. Both algorithms are composed of trivial shifting and selection operations which allows the system to encrypt or decrypt the data during sending and receiving data. This capability

makes the RBS algorithm very efficient in terms of timing overhead. The only considerable overhead in encryption and decryption processes is MAC implementation which will be discussed in detail in experimental results chapter.

#Reception Algorithm	**#Transmission Algorithm**

counter = 0
for *i* in range (*n* + *m*)
{
if (key[*i*] = 0)
{
shift right plaintext register
send LSB(plaintext) to Enc module
}
else
{
shift right redundant register
send LSB(redundant) to Enc module
}
shift right key register
}
	counter = 0
	for *i* in range (*n* + *m*)
	{
	if (key[*i*] = 0)
	{
	Send data to MSB(plaintext)
	shift right plaintext register
	}
	else
	{
	send data to MSB(redundant)
	shift right redundant register
	}
	shift right key register
	}

m: plaintext data length *n*: redundant data length	

Figure 3-15 Transmission and reception algorithms

3.6 Overall System

RBS algorithm performs encryption and decryption processes along with authentication with the same hardware. Since the RFID systems are half-duplicated, at one moment either encryption or decryption will be performed while authentication is a part of their process not an optional service.

Figure 3-16 describes the overall system with a flowchart. In encryption mode, first the MAC of plaintext, redundant bits is generated. In this part, first the MAC generator is initialized with the XOR of the plaintext and the secret key. Then at each clock, the plaintext is entered into the MAC generator bit by bit. It repeats for n+1 clock cycles where n is the

size of plaintext. The next step is generating the keystream which is similar to the last step with this difference that it repeats *m* cycles where *m* is the number of redundant bits. Then the result of bitwise addition of the keystream and the plaintext is calculated as the altered plaintext. The last step is transmitting the message. In this step, based on the secret key either the altered plaintext or redundant bits will be sent out.

In decryption mode, first at the receiver side, the altered plaintext is separated from the redundant bits based on the key secret and shifted into their own registers as the receiver is receiving the message bit by bit from demodulator. Then, with having the redundant bits, the keystream is generated by calculating MAC of redundant bits as same as in the encryption mode. Afterward, the plaintext is found by executing bitwise addition on the keystream and the altered plaintext. Eventually, by having the plaintext, the redundant bits are regenerated to be compared with the received redundant bits. If these two are same then the message will be authenticated. Unless, the message will be known as corrupted message and discarded.

3.7 Conclusion

This chapter was dedicated to describing the RBS algorithm which is a new authenticated symmetric encryption method for RFID systems based on inserting redundant bits into original data bits. The proposed method provides authentication, integrity, and confidentiality, all together. The security level of the proposed system can be adjusted without changing the underlying MAC algorithm just by changing the number of redundant bits and the plaintext.

RBS algorithm is counted as a lightweight cryptosystem since the performance, cost overhead and its security strength are acceptable in the definition of these type of encryption algorithms. Compared to other cryptosystems, the only disadvantage of this algorithm is length of the ciphertext which is longer than others. However, if these cryptosystems want to provide the authentication, the length of RBS ciphertext will be comparable with the length of their ciphertexts especially for stream ciphers which the authentication part is recommended.

At the end of this chapter, the hardware implementation of the RBS algorithm in three parts is explained: the MAC generator to produce the redundant bits and the keystream, the encryption cipher embedded in the sender to merge the altered plaintext with the redundant bits and the decryption cipher embedded in the reception to separate redundant bits from the altered plaintext. The main part of this hardware is the MAC generator which takes more resources than other two parts. The presented MAC generator for RBS cipher is adapted from [56]. To make this MAC compatible with the proposed RBS cipher, some modification has been done on initialization phase and also LFSR part. These modifications made the considerable resource saving in term of area and power. However it imposed some extra cycles which makes the performance of RBS cipher disgraced slightly.

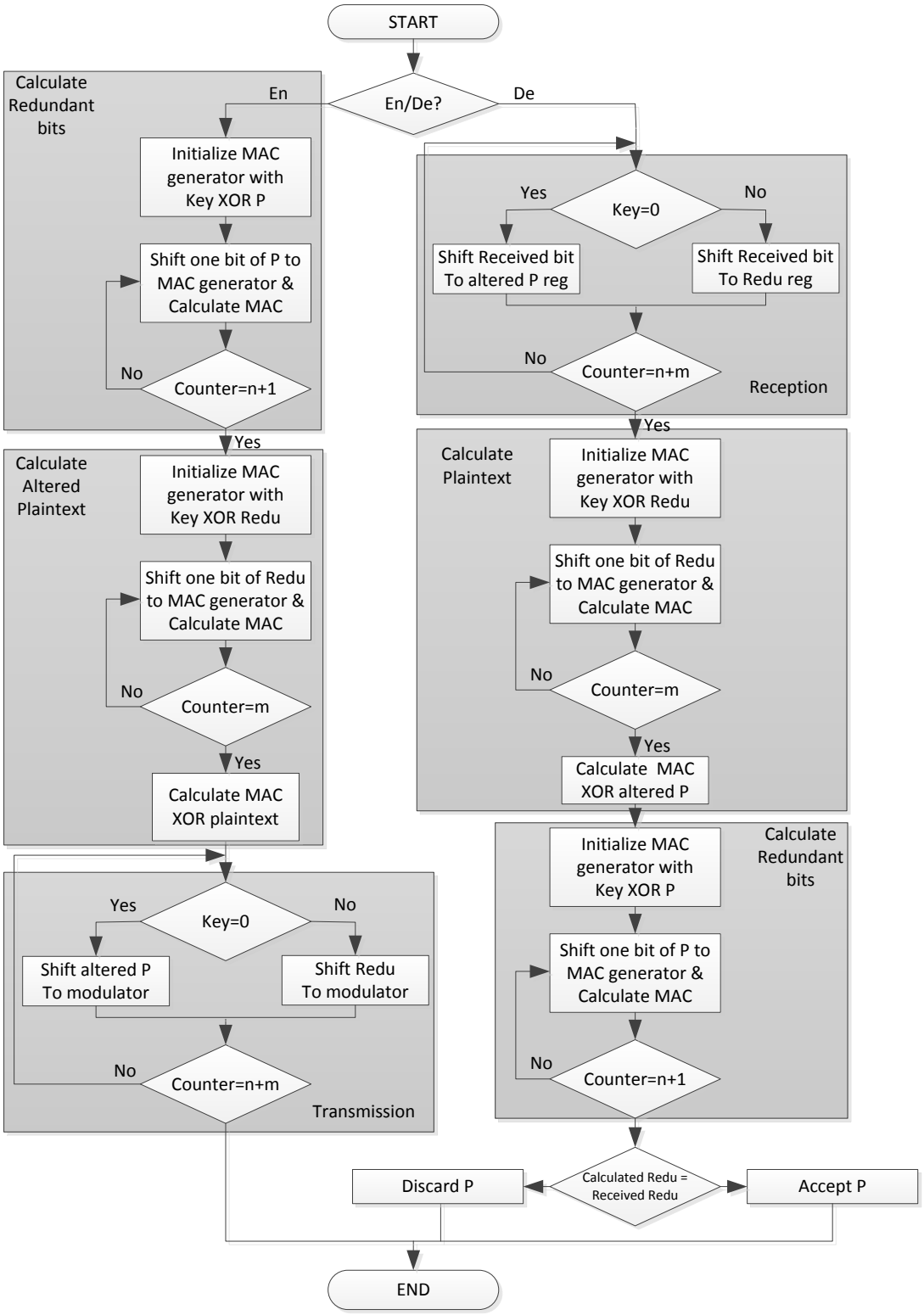


Figure 3-16 The flowchart of the RBS algorithm of the overall system

CHAPTER 4: Security Analysis

One of the most important factors for having a high secure cryptosystem is resisting against attacks. These attacks are well known and there is a possibility that RFID systems be attacked by them. Having a light cipher in terms of area and power consumption can be applicable when this factor is provided. Unless, the proposed cipher will not useful against all of advantages in its hardware implementation.

In CHAPTER 3, RBS algorithm and its hardware implementation are introduced and discussed in details. In this chapter, the security of this proposed algorithm is investigated against some powerful and well-known attacks such as known-plaintext attack, chosen-plaintext attack, chosen-ciphertext attack, differential attack, substitution attack, related key attack, linear cryptanalysis algebraic attack, cube attack and side channel attack. It is shown in the following that how RBS algorithm is resisting against these attacks.

4.1 Security Model

A typical RFID system consists of one eligible reader and N number of RFID tags. Per each tag there is a unique key that is shared with the authorized reader. Consider the scenario where a sample tag A sends its encrypted message through ciphertext C to an eligible reader B. Listening to all signals on the channel and modifying them, adversary ε tries to discover the encryption key over the following operations:

- Eavesdropping: ε captures signals transmitting between A and B, demodulates and decodes the received signals to extract C.

- Modification: ε modifies the signals on the channel in order to i) alter the data through bit flipping (converts bit “0" into “1" or vice versa), ii) append bits to data, and iii) delete some of data bits.

Changing the length of data by ε in this security model either via appending some bits or deleting some of the bits, invalidates the ciphertext C in RBS system as the size of the ciphertext is fixed. Therefore, this issue will not be discussed here.

4.2 Mathematic Background

In RBS algorithm, plaintext P ∈ F_2^n and encryption key K ∈ F_2^{n+m}, are used together to generate the ciphertext C ∈ F_2^{n+m} where *n* is length of plaintext and *m* is length of redundant data.

The Key K is divided into two sub-keys $K_{NFSR} \in F_2^m$ and $K_{LFSR} \in F_2^n$.

$$K = \{K_{LFSR}, K_{NFSR}\} \text{ while } K_{LFSR}=\{K_0,\ldots,K_{n-1}\},\ K_{NFSR}=\{K_n,\ldots,K_{n+m-1}\} \quad (4\text{-}1)$$

Let $rd \in F_2^m$ represent redundant data in the ciphertext. Then

$$rd = \mathrm{MAC}_{KA}(P) \text{ where } K_A = \{(K_{NFSR} \oplus P), g(K_{LFSR},P)\} \quad (4\text{-}2)$$

Let $ks \in F_2^n$ be representative of keystream. Then

$$ks = \mathrm{MAC}_{KB}(rd) \text{ where } K_B = \{(K_{NFSR} \oplus rd), g(K_{LFSR},rd)\} \quad (4\text{-}3)$$

The altered plaintext, $ap \in F_2^n$, is generated by bitwise addition of keystream *ks* and plaintext *P*.

$$ap = ks \oplus P \tag{4-4}$$

The ciphertext is generated by merging *rd* with *ap* under the encryption key *K*. In other words, $C = E_K(rd, ap)$. In this security model, tag *A* encrypts its message *P* with encryption key *K* and sends it to reader *B* using above equations.

$$A \rightarrow B: \ \{\mathrm{MAC}_{KA}(P), \mathrm{MAC}_{KB}(\mathrm{MAC}_{KA}(P)) \oplus P\}_K \tag{4-5}$$

Assuming that MAC encrypts the message with *K*, the following Lemmas hold true if ε does not have *K*.

- Lemma 1- if ε does not have the K, he cannot retrieve the corresponding redundant data *rd* for plaintext P as $rd = \mathrm{MAC}_{KA}(P)$.

Proof: To find the MAC of any message, having both the key and the message is required. In RBS terminology, $rd = \mathrm{MAC}_{KA}(P)$ and since the attacker does not have the key, he cannot find the redundant data of that plaintext.

- Lemma 2- if ε does not have the *K*, he cannot retrieve the corresponding keystream *ks* for plaintext *P*.

  Proof: Based on Lemma 1, ε cannot find *rd* from *P* so he cannot find $ks = \mathrm{MAC}_{KB}(rd)$

- Lemma 3- if ε does not have the *K*, he cannot retrieve the altered plaintext *ap* from plaintext *P*.

Proof: based on Lemma 2, *ks* is not revealed through *P* and so $ap = ks \oplus P$ is not revealed as well.

- Lemma 4- if ε does not have the K, he cannot retrieve keystream ks from redundant data rd as $ks = \mathrm{MAC}_{KB}(rd)$.

  Proof: It is proved like Lemma 1.

- Lemma 5- if ε does not have the K, he cannot retrieve altered plaintext ap from redundant data rd.

Proof: Based on Lemma 2, ε cannot find ks from rd and consequently he cannot find ap since in $ks \oplus P$, ks is unknown for ε.

MAC is a one-way encryption algorithm and having the digest/key combination, it is practically impossible to retrieve the plaintext. The following Lemma holds true over this fact if ε does not have access to the key K:

- Lemma 6- plaintext P cannot be retrieved from redundant data rd.
- Lemma 7- plaintext P cannot be retrieved from keystream ks.
- Lemma 8- plaintext P cannot be retrieved from altered plaintext ap.
- Lemma 9- redundant data rd cannot be retrieved from keystream ks.
- Lemma 10- redundant data rd cannot be retrieved from altered plaintext ap.
- Lemma 11- if ε does not have the K, he cannot retrieve neither rd nor ap from Ciphertext C.

Proof: Ciphertext C is a mixture of redundant data rd, and altered plaintext ap. Based on Lemma 5 and Lemma 10, rd is not revealed from ap and vice versa. Therefore, ε may try arbitrary combinations of rd and ap in the ciphertext C. Although there is dependency between rd and ap, but based on Lemma 3, even having rd, ε cannot retrieve ap.

4.3 RBS Security

In the rest of this chapter, the security of RBS algorithm is studied against powerful and well-known attacks in cryptanalysis such as known-plaintext attack, Chosen-Plaintext Attack and Chosen-Ciphertext Attack etc.

4.3.1 Brute Force Attack

The security strength of a cryptographic algorithm depends on how difficult it is to break an encrypted message without the knowledge of encryption key used by the algorithm. Unconditional security would be perfect, however the only known such cipher is one-time pad. For all other encryption algorithms, providing the assumed computational security is possible if it either takes too long, or is too expensive to break the cipher, assuming that the attacker has access to reasonable computing resources and time to break the cipher. The cryptographic strength may also depend on other parameters such as the worth of the message, the lifetime of the message privacy etc. which are not interested here. The best parameter for measuring the security strength can be the number of possible keys which the adversary has to try to retrieve the right key.

Brute-force attack is a powerful attack which involves trying all possible keys until an understandable relation between the ciphertext and its plaintext is obtained based on that key. The length of key used in the cipher determines the security level of the cipher against brute-force attack. Increasing the length of keys will increase the time required for breaking the cipher exponentially. Therefore, for a cipher with n-bit key, it can be broken in a worst-case time proportional with 2^n. However, because of weak keys, half of all possible keys must be tried to accomplish success on average.

Table 4-1 shows how much time is required to perform a brute-force attack for various common key sizes when either a single decryption process or a million parallel processes are used. Regarding the table, keys with 128 bit size or longer provides enough security for contemporary cryptosystems. However, for RFID systems the length of key used in the lightweight encryption algorithms is between 80 bits (e.g. PRESENT) to 128 bits because of the trade-off among the security, cost and performance. To have a higher security with keys longer than 128 bits, the cost of area and power will become higher while the performance is decreasing. In RBS algorithm, the length of key can be varied between 83 to 132 bits which provides the same key space that keys with length of 80 to 128 bits provide.

Table 4-1 Time required for breaking key by Brute Force attack

Key size [bits]	**Key space**	**Time required at 1 decryption/µsec**	**Time required at 10^6 decryption/µsec**
32	2^{32}	35.8 minutes	2.15 msec
56	2^{56}	1142 years	10.01 hours
80	2^{80}	1.9×10^{10} years	1.9×10^{4} years
128	2^{128}	5.4×10^{24} years	5.4×10^{18} years
168	2^{168}	5.9×10^{36} years	5.9×10^{30} years

4.3.2 Known-Plaintext Attack

In known-plaintext attack, the attacker ε has a pair of valid plaintext/cipher (P/C) and tries to discover the key, K. It can happen through eavesdropping on the channel between tag and reader when the tag is sending a special message that is likely ε has access to it such as EPC number of the tag.

Based on Lemma 1 and Lemma 3, ε cannot retrieve *rd* and *ap* from P so he cannot locate them inside C as well.

To regenerate P through C, ε tries arbitrary combinations of $\dot{K}, \dot{rd}$ and $\dot{ap}$ from C where $C=Merge_{\dot{K}}\{\dot{rd}, \dot{ap}\}$. Based on Lemma 6, P cannot be revealed from only rd. Likewise, P cannot be revealed from only ap based on Lemma 8. The only way to regenerate P is through right $\{\dot{rd}, \dot{ap}, \dot{K}\}$ combination which satisfies $C=E_{\dot{K}}(\dot{rd}, \dot{ap})$. Meanwhile, there are as many as key space size different combinations for $\dot{K}$ in RBS cipher whereas only one $\dot{K}$ satisfies $C= E_{\dot{K}}(\dot{rd}, \dot{ap})$. In other words, knowing one pair of P/C does not shrink the key space.

4.3.3 Chosen-Plaintext Attack

A chosen-plaintext attack is an attack model for cryptanalysis by assuming that the attacker has the ability of choosing arbitrary plaintexts to be encrypted and gain their corresponding ciphertexts [69]. The goal of the attacker is to find out some further information which reduces the security strength of the encryption algorithm by comparing ciphertexts and their plaintexts. In the worst case, the attacker may find the secret key by this attack.

Based on the definition of this attack, it is assumed that the attacker ε has access to decryption device and he can encrypt his own plaintexts into their ciphertexts. Regarding RBS algorithm, the attacker will try to find location of rd bits or ap bits inside C by this information.

Suppose that ε has two pairs of (P_1/C_1) and (P_2/C_2) while $C_1=Merge_k\{rd_1, ap_1\}$ and $C_2=Merge_k\{rd_2, ap_2\}$. The attacker performs peer to peer comparison of bits in C_1/C_2 for extracting some further information in order to reduce the key space.

Changing each bit of plaintext reflects itself over some of the bits of redundant data and altered plaintext, but the number of changed bits as well as their location depends on the used MAC algorithm key which is also unique per each plaintext and is unknown to ε. Therefore, ε cannot predict any changes in ciphertexts of two plaintexts.

Comparing each pair of bits, $C_1[i]$, $C_2[i]$, they might be either equal or not. If $C_1[i] \neq C_2[i]$, this change might represent either change in rd_1 or ap_1. Besides, for $C_1[i] = C_2[i]$ the i^{th} bit might belong to *rd* or *ap*. For these reasons, no useful information about the key can be obtained by tracing changes in C_1 and C_2.

Considering plaintext data as well as ciphertext data, changing just one bit of plaintext changes about half of ciphertext bits since *rd* and *ap* are generated through MAC and these changed bits are randomly distributed in ciphertext based on this fact that the underling MAC is a PRNG. Besides, these changed bits may belong either to *rd* or *ap*. Since this is a very special case, studied in differential attack, we leave it here and discuss it in more detail later.

4.3.4 Chosen-Ciphertext Attack

A chosen-ciphertext attack is an attack model which the attacker ε has the capability to decrypt his own ciphertexts and retrieve their corresponding plaintexts. Suppose that ε has captured a valid ciphertexts C through eavesdropping on the channel and has also decrypted its corresponding plaintext *P*. Since he has access to decryption device, ε can modify some bits of $C=Merge_k\{rd, ap\}$ to interpret their reflection on its decrypted data. It must be noted that ε does not know *K*, *ap*, and *rd* based on Lemma 11.

For a particular modified bit, if it belongs to *rd*, the decryption part will not authenticate *C* as *rd* is unique per *ap*. Likewise, if the modified bit belongs to *ap*, *C* will not be authenticated as *ap* is also unique per *rd*. In both cases, the decryption part will return string of *0*'s which does not reveal any information about *K*.

In order to get a valid plaintext *P* from modified *C*, the changed bits must belong to both *rd* and *ap* such that they must satisfy $ap = \mathrm{MAC}_{KB}(rd) \oplus P$. On the other hand, the number of possible ciphertexts in RBS cipher is 2^{m+n} while 2^n out of them can be authenticated and accepted at the receiver party where *m* and *n* are the length of redundant data and plaintext respectively. It means that the probability of finding the right match of *ap* and *rd* is as low as 2^{-m}. Suppose even such a rare match occurs and ε collects valid ciphertexts $\{C_1, C_2\}$ and their corresponding plaintexts $\{P_1, P_2\}$. This scenario would be similar to chosen-plaintext attack which finding that the modified bits belong to *rd* or *ap* is not practically possible. Consequently, having these data does not shrink the key space.

4.3.5 Differential Attack

Differential attack is a chosen-plaintext attack which extracts the relationship between the difference of two inputs and the difference of their corresponding outputs. In this attack, the attacker searches for plaintext, ciphertext pairs whose difference is constant or non-random, and investigates the differential behavior of the cryptosystem with this hope that he may detect statistical patterns in distribution of differences between the ciphertexts (Figure 4-1). The difference between two plaintexts is very small and usually the difference is in just one bit.

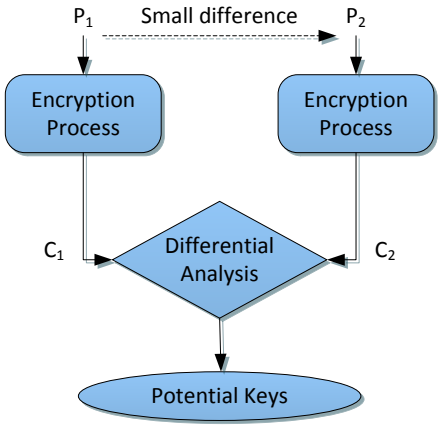


Figure 4-1 Differential attack

If (P_1,C_1) and (P_2,C_2) are two pairs of different plaintext and ciphertext then $(\Delta P,\Delta C)$ states the difference between two given pairs of plaintexts and ciphertexts where $\Delta P=P_1 \oplus P_2$ and $\Delta C=C_1 \oplus C_2$.

In order to resist against differential attack, the cryptosystem should be a good pseudorandom number generator (PRNG). The underlying implemented MAC in RBS is based on pseudorandom generator as discussed in CHAPTER 3. This random behavior has been verified by simulating the output of RBS cipher for a set of 64-bit input plaintexts where each two adjacent inputs in this set are different in only one bit. The RBS algorithm has been applied for 100 different keys per each input in the set. The results for the first ten keys are summarized in Table 4-2. Considering RBS-132, the 64 generated 132-bit ciphertexts for 64 input plaintexts in the set as a two dimensional array of 132×64, we have investigated the effect of one single bit change in all 132 columns and 64 rows of this array.

Regarding the changes in the columns of this array, Table 4-2 lists the minimum, average, and maximum transitions in ciphertext bits by changing only one bit of the plaintext. For instance by applying Key1, 31.06 transitions has occurred per each bit of the redundant data in average. In other words, by a single bit change in the plaintext each bit of the

redundant data will change by the probability of 31.0/64 ≈ 0.49. Almost the same results are generated for altered plaintext.

Regarding the rows of the array, the average number of redundant bits and altered plaintext bits that are different from the output of previous input is also presented in Table 4-2 as well. The reflected changes on redundant part and altered plaintext part of ciphertext are again close together. As the last column of the table shows, just a single bit change in the plaintext, transforms almost half of ciphertext bits uniformly in both redundant part and altered plaintext part of ciphertext.

Performing the simulation test for special case inputs, i.e. plaintext of all 0's or all 1's almost the same results have been generated. So, this simulation practically confirms the good pseudorandom behavior of RBS which is a prerequisite for being resistant against differential attack. These results were expected as the underlying implemented MAC in RBS is based on pseudorandom generator as discussed in CHAPTER 3.

Table 4-2 Simulation of RBS outputs when the inputs are different in one bit

	Redundant data (#bits: 68)				**Altered plaintext (#bits: 64)**				**Total**
	Column			**Row**	**Column**			**Row**	
Key	**Max Transition**	**Min Transition**	**Average Transition**	**%Changed bits**	**Max Transition**	**Min Transition**	**Average Transition**	**%Changed bits**	**%Changed cipher**
1	41	23	31.06	51%	42	23	31.53	50%	50%
2	41	21	30.57	51%	41	22	31.02	49%	49%
3	40	22	33.46	50%	41	23	32.2	50%	50%
4	40	25	32.16	49%	37	24	31.8	48%	49%
5	39	21	32.82	50%	44	22	32.47	51%	50%
6	45	23	32.94	52%	39	24	31.73	51%	51%
7	39	23	31.6	48%	41	24	32.52	50%	48%
8	43	22	31	48%	41	21	31.42	50%	49%
9	42	22	32.44	50%	41	20	31.58	50%	50%
10	40	22	31.81	51%	41	21	31.66	49%	49%

4.3.6 Substitution Attack

Substitution attack is especially introduced for stream ciphers which their ciphertext is the result of performing bitwise addition of the keystream and the plaintext. In this attack, the attacker tries to replace a legitimate pair of message and MAC ($\boldsymbol{m}$, $\boldsymbol{t}$) with his own pair of message and MAC ($\boldsymbol{m}'$, $\boldsymbol{t}'$) and succeed with probability P_S. Therefore if the attacker can find such a pair, he can send his own message while it will be authenticated in the reception party.

It has already been proved in [56] that the utilized MAC generator in RBS is resistant against this attack and the probability of substitution of the pair of message and MAC is very low while the message and MAC are separated and can be distinguished in the ciphertext. The probability of this attack in RBS will be even less since based on the lemmas 1 and 3; the attacker even cannot distinguish the message from its MAC in the output.

4.3.7 Related Key Attack

Related key attack is first introduced in [70]. In this attack, it is assumed that the attacker ε can obtain the ciphertext from the cipher for different plaintexts under different keys *K1*, *K2*, etc. The values of these keys are initially unknown for the attacker. Using these data, he tries to look for some information about the secret keys by observing the operation of the cipher under several different keys for same plaintexts and finding the relation between their output ciphertexts.

One possible scenario in RBS implementation could be supposed as the attacker gives the same plaintext P to different tags which have different keys (K_1, K_2) and tries to analyze the generated ciphertexts (C_1, C_2) for that particular message P.

As discussed before, the redundant data (*rd*) that is the MAC generator's output is implemented as a function of the plaintext *P* and key *K*. In other words, $rd_1 \neq rd_2$ as $K_1 \neq K_2$. Since the attacker does not have access to K_1 and K_2, he cannot find corresponding rd_1 and rd_2 based on Lemma 1. There is the same condition for altered plaintexts ap_1 and ap_2 based on Lemma 3. Therefore, the attacker cannot extract (rd_1, ap_1) and (rd_2, ap_2) from ciphertexts C_1 and C_2 respectively. In other words, he cannot extract any useful information to shrink the key space size.

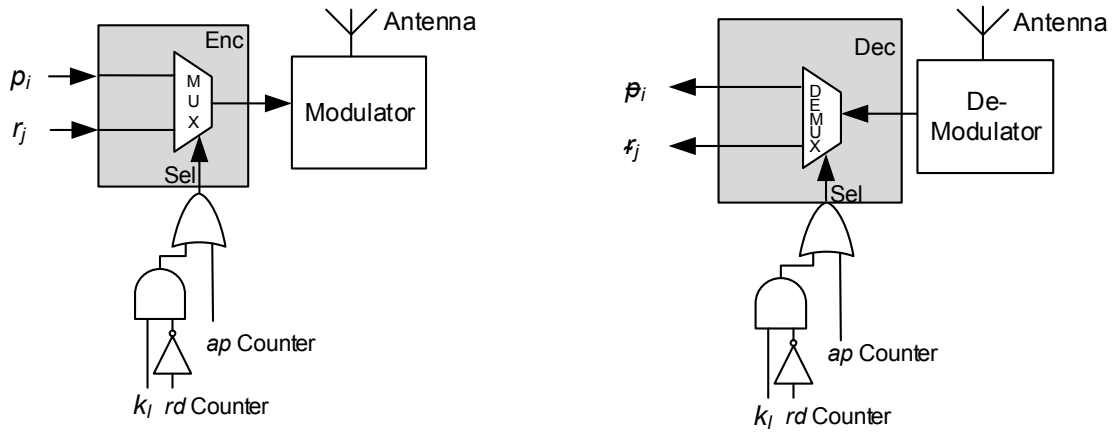
The same simulation for differential attack has been performed for related key attack with this difference that for the same plaintext, ciphertexts are generated for secret keys different in one or more bits. The result of this simulation, confirms that outputs of RBS cipher have random behavior by changing key as we expected.

The keys used for encryption/decryption and authentication are same in RBS ciphers. Therefore, any changes in the key will change both the value of redundant bits and altered plaintext along with changing their locations in the ciphertext. So, by applying different keys different ciphertexts will be generated which makes the cipher resistant against related-key attacks. Having different keys for encryption and authentication may make RBS cipher vulnerable to related-key attacks unless the dedicated authentication key becomes XOR-ed with the encryption key before applying to MAC generator. Thus, the applied key to the MAC will be a function of encryption key and authentication key and any changes in any of keys will have effect on the whole of the ciphertext.

Apart from comparing the output of RBS cipher for different valid secret keys, the attacker may modify the secret key partially and encrypt the message under an invalid

modified key while the key remains secret for attacker. For example, in RBS, there is a possibility that filliping any bits of the secret key may change the balance of number of redundant bits and altered plaintext bits in the ciphertext. If it does not change the balance in the key, the message will be authenticated in the receiver side (if attacker has modified the key at both sender and receiver sides). In this case, those flipped bits in the key are complement of each other. If modifying the key changes the balance of the encryption key, the message will not be authenticated in the receiver side since the receiver has not received all bits of either altered plaintext or redundant bits. Therefore, authenticating the message at the reception helps the attacker to find the relation between the bits of the key.

Preventing this attack is easy by error correcting of the secret key which can be performed by adding two counters. These counters are responsible to count the number of redundant bits and altered plaintext bits inside the ciphertext (Figure 4-2) and check the balance of ciphertext when the encrypted message is sending out or being received. If one of the counters reaches to its maximum pre-defined number, it will dominate the select pointer to correct selecting the inputs/outputs in encryption/decryption process and fix the balance of redundant bits or altered plaintext at the rest of the ciphertext.



(a) Transmission side **(b) Reception side**

Figure 4-2 Error correcting of secret key

4.3.8 Linear Cryptanalysis

Linear attack is a known plaintext attack which was introduced for first time in [71]. In this attack, the attacker tries to find linear expressions involving some bits of plaintext, ciphertext and secret key. This expression can be stated as Equation 4-6:

$$X_i + X_j + \ldots + Y_a + Y_b = K_m + \ldots + K_n \tag{4-6}$$

Where X_i represents the i-th bit of the plaintext, Y_a represents the a-th bit of the ciphertext and K_m represents the m-th bit of the secret key.

In RBS cipher, the key is applied after initialization for two rounds which passes through a nonlinear function of NFSR. Also, during the process of generating the redundant data and the keystream, at each clock cycle, one pseudo random is generated and shifted into the NFSR. The value of each bit of the output is the result of bitwise addition of these random numbers AND-ed by the input bits. Since the coefficient of the random number for

each bit of output is different, factorizing of the coefficients to find a linear function is not possible.

4.3.9 Algebraic Attack

Algebraic attack is a new cryptanalytic method which is designed especially for stream ciphers based on LFSRs. The main idea behind this method is discovering and solving a system of multivariate polynomial equations over a finite field. In this kind of attack, the attacker has a plaintext and its corresponding ciphertext. Therefore the attacker can find the corresponding keystream based on these two given inputs. Based on the obtained keystream, he sets up a system of polynomial equations which its entry is the keystream [72]. The goal is recovering the initial state of LFSR which is the key. Let assume one equation is found for *t*-th bit of the keystream as shown in equation 4-7:

$$0 = f(k_t, \dots, k_{t+r-1}, z_t, \dots, z_{t+r-1}) \tag{4-7}$$

Where z_i is the *i*-th bit of the keystream and k_i is the *i*-th bit of the secret key. Then the same equation is correct for any clock:

$$\begin{cases} 0 = f(k_0, \dots, k_{r-1}, z_0, \dots, z_{r-1}) \\ 0 = f(k_1 + \cdots + k_r, z_1, \dots, z_r) \\ 0 = f(k_2 + \cdots + k_{r+1}, z_2, \dots, z_{r+1}) \\ \vdots \end{cases} \tag{4-8}$$

If the relation between states and output bits can be stated as one multivariate equation of low degree without extra variables then the cipher will be broken in polynomial time [72]. In block ciphers, attackers set up a matrix of multivariate functions which its

variables are bits of the input plaintext, the ciphertext and the secret key. Solving this system of function will lead to recovering the key.

The output of RBS cipher is composed of mixture of two strings, redundant data, *rd* and the altered plaintext, *ap*. Redundant data is a function of the secret key and the input plaintext and it is independent of the altered plaintext which is a function of and the keystream, *ks* and the input plaintext while the keystream is a function of redundant. In this kind of attack, the attacker is required to set up a system of equations for each of these two strings separately because each of these two strings has different initial key and also inputs. Setting up these equations will not happen unless these strings have been already distinguished in the ciphertext. While in known-plaintext attack, it is proved that by having a pair of the plaintext and ciphertext it is not possible for the attacker to find the redundant data and its corresponding altered plaintext in the output.

4.3.10 Cube Attack

Cube attack is a chosen-plaintext attack introduced for symmetric cryptosystems. The ciphertext bits produced by this algorithm are values of polynomials depending on initial vectors in stream ciphers and bits of a plaintext plus secret key for block ciphers [73]. In this attack the attacker tries to obtain a linear equation of the secret key bits by combining the equations for an output bit of the cipher for a set of inputs and keys. A cipher is vulnerable to this attack, if an output bit of the cipher can be represented as a sufficiently low degree polynomial over GF(2) of key and input bits. This attack is a type of algebraic attack which its degree of variables in the equation is more than one. For example, in the following equation p is a polynomial of degree 3 with 5 variables:

$$p(x_1, x_2, x_3, x_4, x_5) = x_1x_2x_3 + x_1x_2x_4 + x_2x_4x_5 + x_1x_2 + x_2 + x_3x_5 + 1 \quad (4\text{-}9)$$

The success probability of the cube attack is high if the degree of the internal state transit function in a stream cipher is low. For example, Trivium is vulnerable to this attack because the degree of its internal state transit function grows slowly [49]. In RBS, resisting against this attack depends on the nonlinear function of NFSR. This function can be introduced such that it can provide a high degree of states.

4.3.11 Side Channel Attack

Until now, all investigated attacks were based on the analysis of attacker's knowledge about the algorithms not the hardware to find their weak points. This information may consist of a set of plaintexts and their corresponding ciphertexts like known-plaintext and chosen-plaintext attacks. Based on this information attacker tries to find a relationship between them and the secret key. However, in a side channel attack, the attacker relies on the information he harvested from the physical implementation of the cipher like timing analysis [74], power monitoring [75], fault attack [76], electromagnetic radiation [77] etc. Figure 4-3 shows a cryptographic model consisted of side channel attacks

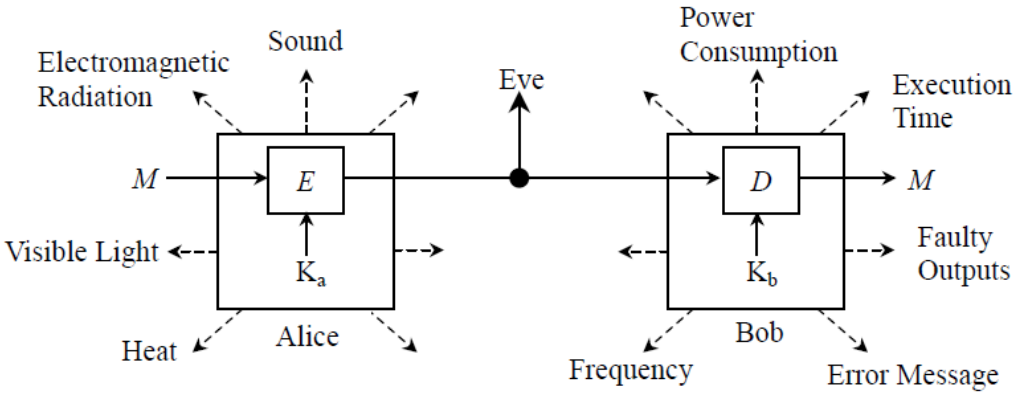


Figure 4-3 The cryptographic model including side channel attack [78]

Among all type of side channel attacks, power analysis is most powerful attack since the power consumption of a cipher may provide a lot of information about the running operations and their involved parameters. Besides, this attack needs very simple set of equipment like a PC with an oscilloscope and a small resistor in power supply line to measure the power. Power analysis attacks have been proved to be very effective attack against implementation of many symmetric and public key algorithms [79].

In RBS algorithm, the value of two registers NSFR and Accumulator are XOR-ed and stored in the Accumulator if the input message to the MAC is one unless nothing happens. Therefore when the input is one the power consumption will be high to perform bitwise of two registers while the power consumption will be considerably low when the input is zero. However, the current will not be zero since some other operations like calculating the nonlinear function of NFSR and shifting it required to be performed. Based on this information, an attacker by tracking the drained current in time can find if the input message is zero or not during the generating the redundant data. Also he can find the redundant data during generating the keystream since the redundant data is the input of the MAC in this process. By having the redundant data, the key space will be shrunk dramatically and finding the secret key for the attacker will be easier.

One solution for resisting against side channel attack is adding some extra hardware as a redundant circuit such that when the input message into the MAC is zero, this part will become activated so the power consumption of the circuit goes higher. This trick can confuse the attacker in finding the value of input (Figure 4-4). However, it is estimated that this solution will impose about 50% more in cost of area and power consumption.

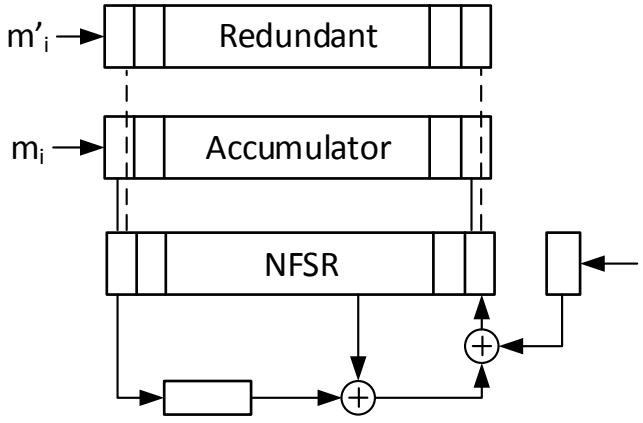


Figure 4-4 Adding redundant MAC generator to RBS cipher

Another reasonable solution is adding extra hardware to the implantation in order to have a parallel architecture which can process two or more bits of input message at the same time (Figure 4-5). This solution also needs increasing the cost of power and area about 30 to 40%. However, in result, the performance of the cipher will be increased 2 or 3 times more.

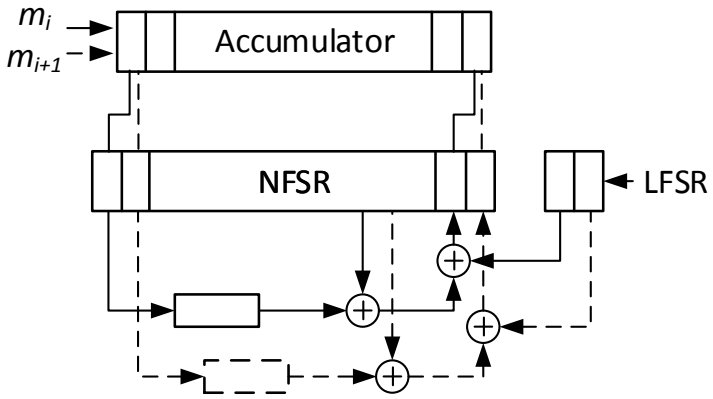


Figure 4-5 RBS cipher with radix-2

4.4 Conclusion

In this chapter, the security of the RBS algorithm against the following attack has been investigated: known-plaintext attack, chosen-plaintext attack, chosen-ciphertext attack, differential attack, related-key attack, substitution attack, linear attack, algebraic attack, cube attack and side channel attack. These attacks are very powerful attacks that have broken

many contemporary ciphers. In this analysis, it was shown that how the RBS cipher is or can be resistant against these attacks. However, there are still more attacks that may threaten the RBS cipher. Most of these attacks are the improved version of the mentioned attacks or a combination of them. Nevertheless, for sake of having secure communications, it is required to investigate the security of this cipher against these attacks in future.

CHAPTER 5: Experimental Results

In CHAPTER 3, RBS algorithm along with its proposed hardware implementation design was introduced and in CHAPTER 4, the security of this algorithm against some powerful and well known attacks was explored. In this chapter, the experimental results of the hardware implementation of RBS algorithm will be investigated and its performance metrics in ASIC design will be presented.

The results of RBS implementation will be compared with other lightweight block ciphers and stream ciphers like PRESENT [24], Trivium [48], Grain [54] and Hummingbird (HB-2) [62] studied in CHAPTER 2 while AES [29] is used as a benchmark in this analysis. This comparison will be progressed in two categories. First when competitor ciphers do not provide the authentication service and second, when they do. The important parameters for this comparison is the operating clock frequency, key size, size of data block and used technology to compare one-dimensional metrics like area, performance, throughput, power and multi-dimensional metrics like energy, hardware efficiency, Area-Time product and Power-Area-Time product.

5.1 ASIC Implementation of RBS

The different components of RBS implementation presented in CHAPTER 3 are written in Verilog code description and synthesized by Synopsys Design Compiler in the 90nm technology mode while no tool optimization is set neither for area nor for power. The operating clock frequency is set to be 10 MHz. The operating conditions are set to be typical while the supply voltage is fixed at 1V, and the temperature is set to 25°C. Tables 5-1, 5-2 and 5-3 summarize the reported area and power for different parts of the RBS designs.

Table 5-1 Area report for each part of RBS design [GE]

	RBS-83	RBS-100	RBS-116	RBS-132
MAC Generation	678	816	905	1051
Enc/Dec Cipher	10	10	10	10
Counter	53	53	53	73
Transmitter/Receiver	956	956	956	956

The area report shown in table 5-1 is based on GE which is independent of the used fabrication technology. Therefore changing the underlying technology does not have any effect on the result of area which makes it a respectable parameter in comparisons.

Table 5-2 Static power consumption for each part of RBS design [μW]

	RBS-83	RBS-100	RBS-116	RBS-132
MAC Generation	3.7953	4.5808	4.8024	5.89
Enc/Dec Cipher	0.123	0.123	0.123	0.123
Counter	0.287	0.287	0.287	0.335
Transmitter/Receiver	4.53	4.53	4.53	4.53

Power consumption in CMOS circuits consists primarily of static power dissipated by leakage currents even when there is no change in inputs and dynamic power, which is in turn comprised of the switching power consumed for charging and discharging load capacitances. Static power is a type of dissipation which changing in the level of inputs and outputs does not have any effect on it. Current leakage is a technology dependent parameter. Therefore, for a same design, static power dissipation differs in different technologies. In new technologies, the length of the channel becomes shorter which leads to smaller Gate Oxide thickness. Thus, newer devices have higher static power consumption. To have a fair comparison, all designs need to be fabricated in the same technology.

Similar to static power dissipation, dynamic power is dependent on the underlying technology too with this difference that in newer technology the dynamic power becomes

less for the same design. Because in new technology the size of capacitance is reducing and the power needed to charge and discharge them becomes less too. Besides, dynamic power is dependent on the switching of the logics at their outputs too. The number of switching in a circuit increases proportionally with the operating frequency. Therefore, in comparison of dynamic power of different design it is required for all designs to work in a same frequency and have the same technology. Table 5-3 demonstrates the dynamic power dissipation for different modules in RBS designs.

Table 5-3 Dynamic power consumption for each part of RBS design [μW]

	RBS-83	RBS-100	RBS-116	RBS-132
MAC Generation	15.6047	18.7028	23.1774	24.28
Enc/Dec Cipher	0.143	0.143	0.143	0.143
Counter	0.38	0.38	0.38	0.414
Transmitter/Receiver	24.37	24.37	24.37	24.37

Among the listed modules in Tables 5-1, 5-2 and 5-3, counter and transmitter/receiver are common parts of every typical data communication system regardless of whether the data must be encrypted/ decrypted or not. Therefore, these parts are not considered as overhead in RBS implementation. In other words, RBS algorithm imposes only MAC generator and Enc/Dec Cipher modules to the system. Total power consumption is obtained from summation of the static power and dynamic power. Considering just RBS modules, the total area and also total power consumption overheads of RBS implementation is calculated in Table 5-4.

Table 5-4 Total area and power consumption overhead for RBS designs

	RBS-83	**RBS-100**	**RBS-116**	**RBS-132**
Area [GE]	688	826	915	1061
Power consumption [μW]	19.67	23.55	28.25	30.46

Concerning the required clock cycles, encryption/decryption in RBS algorithm is performed along with data transmission/reception and the performance of RBS is only limited by the time required for generating MAC outputs which are the redundant bits and the keystream. Generating the redundant bits takes m+1 clock cycles where m is the length of the plaintext. Producing the keystream takes n clock cycles where n is the size of redundant bits. Besides, one clock cycle is required for bitwise addition of the keystream with the plaintext plus 2 cycles for generating authentication keys. Altogether, m+n+4 clock cycles is the overhead for encryption/decryption plus authentication in RBS algorithm. For example, in RBS-132, 65 clock cycles takes for generating redundant data and 68 clock cycles for the keystream. Table 5-5 demonstrates the total number of clock cycles to generate the ciphertext for different designs in RBS algorithm.

Table 5-5 The number of clock cycles required for generating the output in RBS

Cipher	Redundant bits time overhead	Keystream time overhead	Total
RBS-83	41	43	87
RBS-100	49	52	104
RBS-116	53	56	120
RBS-132	65	68	136

5.2 Comparison Ciphers

This section presents a comprehensive analysis of all ciphers introduced in CHAPTER 2 from a hardware perspective. The number of ciphers designed and published recently for restricted resources is more than those investigated in this dissertation. However, only those ciphers that cryptanalysis indicates promising security and also considered to have low resource hardware have been chosen to be compared together.

For comparison, considering only one metric like speed or area results into one-dimensional analysis which is not effective in finding the best cipher since each cipher might be good at just one of the metrics while in other metrics they might be unacceptable. In contrary, a cipher might be worse in one of metrics while in other metrics it might be better than other competitors To have a comprehensive analysis, hardware performance is required to be investigated in multi-dimensional with various quantities such as area, performance, throughput, power and energy. It is required to be stated that the AES algorithm is to be used as the benchmark for comparison and to be counted as lightweight cipher; they should be smaller and faster than the AES.

The RBS algorithm is compared with five other encryption algorithms in terms of i) required key and initial vector size, ii) data block size, iii) the required number of clock cycles for completing the encryption process, iv) 2-input NAND GE equivalent area, and v) total power consumption when the clock frequency is set to 10MHz. Table 5-6 summarizes the comparison results in terms of area, performance and power. In this table, the initial clock cycles are shown in parenthesis.

Table 5-6 Comparing RBS with other encryption methods

	Key/IV (bits)	Block Size (bits)	Clock Cycles	Area (GE)	Total Power (μW)	Freq. (MHz)	Tech.
AES	128/-	128	160	3200	300	10	130nm
PRESENT	128/-	64	32	1884	7.34	0.1	180nm
Trivium	80/(80)	1	(1333)*+ 1	2599	181.18	10	130nm
Grain	128/(128)	1	(512)* + 1	1857	167.73	10	130nm
HB-2	128/64	16	(80)* + 16	2332	156.8	10	130nm
RBS-83	83/-	40	(86)*+87	688	19.67	10	90nm
RBS-100	100/-	48	(104)*+104	826	23.55	10	90nm
RBS-116	116/-	56	(120)*+120	951	28.25	10	90nm
RBS-132	132/-	64	(68)*+136	1061	30.46	10	90nm
* Cycles required for initialization							

All compared methods' reports are for 130nm technology, except PRESENT using 180 nm, whereas RBS is synthesized in newer 90nm technology which has considerable effect in area and power overhead. However, area reports are given in GE (Gate equivalent) which is independent of used technology. Considering the effect of technology scaling (α) on power consumption, total power could be normalized by α^2.

The timing, area and power consumption reports in Table 5-6 for AES, PRESENT, Trivium, Grain, and HB-2 algorithms are calculated without considering the authentication's implementation. While RBS algorithm provides also authentication service as mentioned before. Besides, area and power reports for RBS algorithm listed in Table 5-6 include the MAC generator part's area/power overheads which are still better than other compared designs. Similar to area and power overheads, RBS timing overhead is still comparable with other algorithms while their reported timing overhead just considers encryption/decryption process.

Comparing all metrics in one table gives a lot of information which makes finding the best cipher very difficult. In the following sub sections, ciphers are compared for only one metric then they are studied under two or more different metrics in two states. First, when all competitor ciphers provide only confidentiality. Second, when all ciphers provide message authentication service along with confidentiality.

Providing the authentication service in ciphers especially stream ciphers like Grain and Trivium is necessary since the ciphertext is the result of bitwise addition of the plaintext and the keystream. Therefore, if the attacker knows the plaintext by changing some bits of the ciphertext, he can easily replace his own message with the real message while the reception accept the message since it does not have any knowledge that the message has been

manipulated. This fact demonstrates the importance of providing authentication in stream ciphers. On contrary, in block ciphers, by changing any bits of the ciphertext, decryption party may generate an irrelevant and unmeaning message from the corresponding block. However, providing the authentication in block cipher is still necessary since it plays a basic role in providing privacy.

Generally speaking, current hash functions are not acceptable for performing authentication in constrained environments. They require significant amounts of overhead and they are not hardware friendly. Table 5-7 shows the results of hardware implementation of some MAC algorithms designed for RFID systems. As this table states, either their area or their performance overhead is still very high to be integrated with a cryptosystem. Therefore, in case of comparison of authenticated ciphers, those ciphers that cannot provide authentication alone will not be considered.

Table 5-7 Hardware implementation of MAC

Design	Area [GE]	# of Cycles	Dynamic Power	Leakage Power	Freq.	Tech. [nm]	Output Size[bits]
PSQUASH[81]	1624	25.1k	18.3 nW	7.7 nW	100 K	130	48
SHA-1 [30]	4276	405	3.74 µW	23 µW	500 K	130	160
SHA-256 [12]	10868	1128	52.37 µW	N/A	100 K	350	224
MD4[12]	7350	456	N/A	N/A	N/A	130	128
MD5 [82]	10332	68	N/A	N/A	133 M	130	128
RIPEMD[82]	17446	96	N/A	N/A	143 M	130	160

Apart from RBS algorithm which provides the message authentication service along with providing confidentiality, other ciphers either provide this service optionally by applying some small modifications on their hardware like HB-2 and Grain or required to be integrated with other algorithms to provide this service like Trivium and AES. Also, there are some ciphers which have their own dedicated hash functions for this favor like PRESENT

presented in [80]. To compare ciphers when they provide authentication, those which have the ability to authenticate the message will be considered like RBS, HB-2, PRESENT and Grain. Since there is no dedicated hardware for authentication for Trivium and AES, these two ciphers are not considered in comparison of ciphers when the authentication service is provided.

Grain and HB-2 use the same hardware for authentication which is used for confidentiality with some modifications. Unfortunately, there is no official report for this area and its power overhead. Therefore, in this comparison, it will be supposed that this overhead is negligible. However, since confidentiality and authentication services cannot perform at the same time because of sharing hardware, the time required to produce the output will be the summation of time needed for performing for each of them. For PRESENT, some dedicated hash functions are introduced in [80]. Their output sizes are fixed with 64, 128 and 192 bits while their inputs are 64, 80 and 128 bits (Table 5-8). Among these designs, for the sake of collision attack and also satisfying conditions of comparison, H-PRESENT-128 is considered for comparison of authenticated ciphers which uses PRESENT core and supports 64-bit input and 128-bit output at 180 nm and 100 KHz frequency. Like Grain and HB-2, because of resource sharing, encryption and authentication do not happens at the same time for this cipher too.

Table 5-8 The performance of different hash functions based on PRESENT

	Hash output size [bits]	**Input size [bits]**	**Cycles per block**	**Area [GE]**	**Power [µW]**
DM-PRESENT-80	64	80	33	2213	6.28
DM-PRESENT-128	64	128	33	2530	7.49
H-PRESENT-128	128	64	32	4256	8.09
C-PRESENT-192	192	64	108	8048	9.31

For authentication, HB-2 adds a payload of 64 bits for messages between one to eight words. While the authentication code in Grain, is at least 32 bits for messages smaller than 32 bits. For messages longer than 32 bits it is required to expand the authentication message with the message to prevent substitution attack. To have a fair comparison when ciphers provide the authentication service, it is supposed that the length of plaintext as input for all ciphers is 64 bits.

5.2.1 Area

Area is one of the important parameters for comparing different algorithms. It states the amount of silicon required for the core design, excluding power rings and I/O cells. This metric is typically expressed in μm^2. However, the more practical independent method of expressing the area is to express it in terms of the Gate Equivalence (GE), calculated by dividing the total area by the lowest power two-input NAND gate's area. In definition of the lightweight encryption algorithm, a complete RFID tag, including the analog part, might have between 1000 to 10000 GE and for security part; this margin may be kept between 200 to 2000 GE [15]. Based on this defined limitation, having lower area for hardware implementation is one of important factors to helps find a better cipher for encryption. In this metric, the one has the lower area is the better one.

Figure 5-1 compares all ECC designs designated for restricted resource environments introduced in CHAPTER 2. This figure shows that the total area for these designs ranges between 30k to 6k GE which is 15 to 3 times more than the area limitation defined for lightweight cryptosystems while these designs are limited to a special prime fields or one

special elliptic curve. The result in this figure explains why public key encryption algorithms are not proper for providing security in RFID systems.

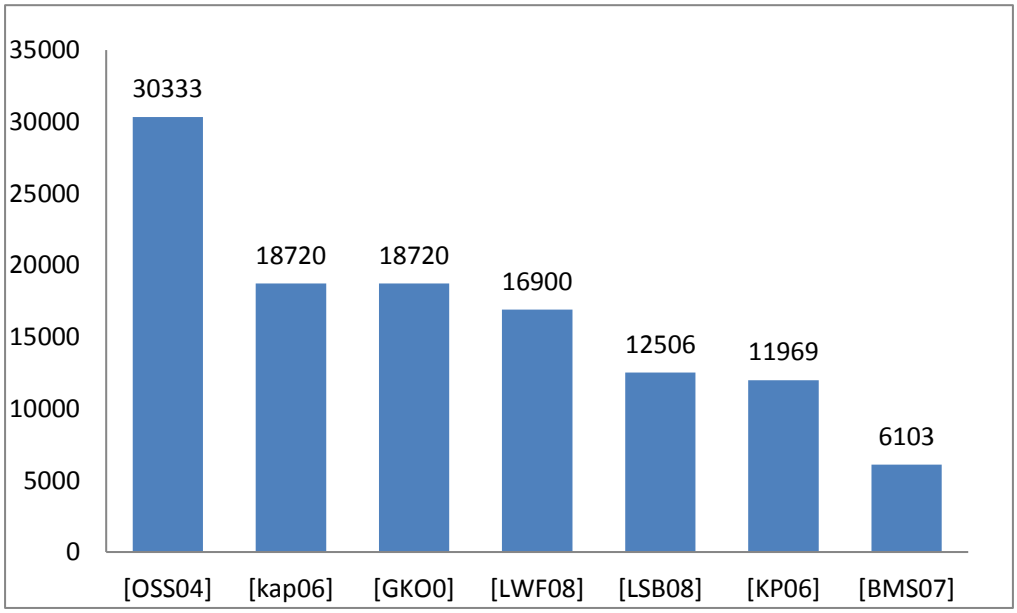


Figure 5-1 Comparing area for different design in ECC

Relying on just one metric like area to remove a cipher from the list of the lightweight cryptosystems is not acceptable. However, regarding to the reported performance for ECC designs in CHAPTER 2, public key cryptosystems are several times slower than private key cryptosystems in term of performance. This correlates at least with a two- to three orders-of-magnitude higher power consumption. Also time for key computation is another overhead which is not negligible factor in asymmetric algorithms.

These two metrics, area and performance are enough to take asymmetric algorithms out of the comparing ciphers list despite all of advantages of these algorithms have over symmetric algorithms like key exchange and key management which they provide. From now on, just private key ciphers will be studied for comparing them based on the different metrics.

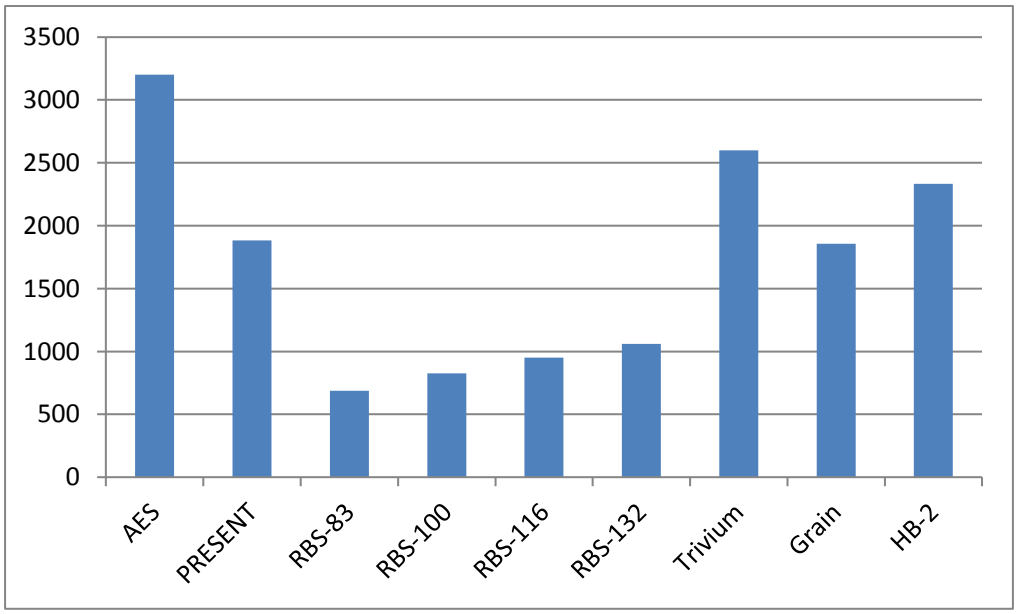


Figure 5-2 Area comparison of symmetric ciphers without providing authentication

Figure 5-2 compares ciphers based on area metric for three block ciphers, AES, PRESENT and RBS, two stream ciphers, Grain and Trivium and one hybrid cipher, HB-2 while AES is considered as a benchmark in this comparison. It must be noticed that the hardware implemented for AES and PRESENT in this report only is provided for encryption process while to support the decryption process, it is required extra hardware which makes their total area more than the reported one. On the contrary, Trivium and Grain support both encryption and decryption processes since both processes need the same hardware. However, area report of Trivium does not cover authentication part which implementing it will impose extra area. For Grain and HB-2, to add authentication service, they need to add some modification on the presented hardware to use it for this purpose too which means a small overhead in area.

Regarding to figure 5-2, among all of the ciphers, RBS designs have the smallest area overhead. The area of RBS-132 is about three times less than area of AES. After RBS, Grain

has the smallest area while the difference between these two ciphers is 796 GE which means RBS-132 is 43% smaller than Grain. This shows the huge distance between RBS and other ciphers in term of area metric. Among four different RBS designs, RBS-83 has the smallest area overhead among them. RBS-83 has 1.5 times less area than RBS-132. This result was predictable since the length of the supporting plaintext and the secret key for this design is the least among all of RBS designs. In fact, this benefit in area overhead is obtained at the cost of degrading the security strength.

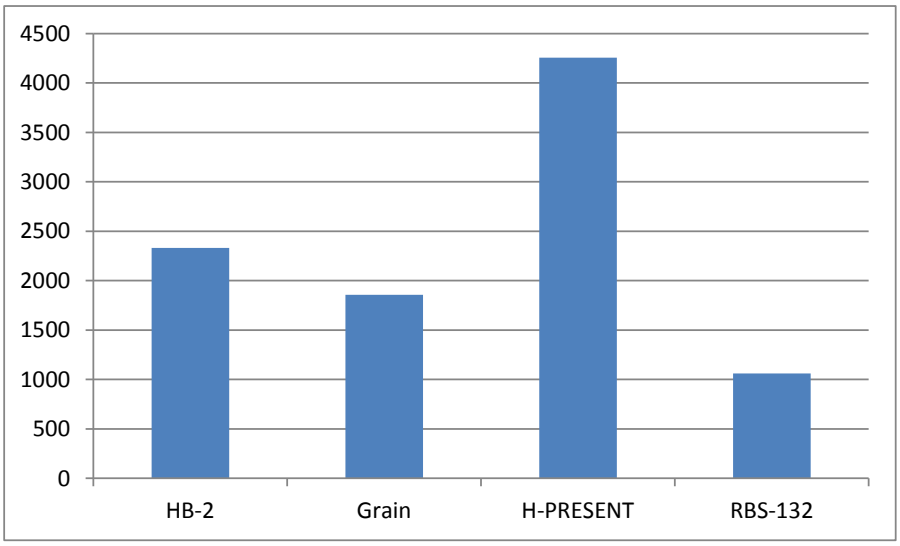


Figure 5-3 Area comparison of different ciphers with providing authentication

Figure 5-3 compares four ciphers when they provide authentication besides confidentiality. Among these ciphers, RBS, Grain and HB-2 are using the same hardware to provide authentication while PRESENT in [80] have an overhead in area to adapt the using hash function with its cipher. Again, RBS has the smallest area among all of these ciphers.

5.2.2 Performance

Performance is one of the important metrics which helps to compare different designs in term of speed. This metric can be studied in two aspects. First, the total number of clock cycles required for generating the ciphertext in the encryption process by ciphers. Since it is independent of the used fabrication technology and the operating clock frequency, it will be a suitable criterion for comparison of different ciphers. This metric will be helpful in finding the number of bits completed per cycles (bits-per-cycle) and the consuming energy in ciphers. The first aspect of performance, total number of clock cycle required for completing processes is stated in two parts; load or initialization cycles and computation cycles.

The second aspect of performance metric is time in term of seconds which it takes to complete a process. This aspect is dependent on the maximum frequency that the designed hardware can work in. The maximum frequency of a circuit is defined by the worst delay caused by its critical path. Measuring the performance in time will be helpful in calculating maximum throughput. In the following subsections, each of these aspects will be explained and symmetric ciphers will be compared based on them. In this metric, the one with the lower clock cycles for computation is the better one.

Load/Initialization Cycles

This period starts from RESET time, through loading the key and IV, until the first bit of the output is ready. RBS cipher like Trivium, Grain, and HB-2 ciphers have used initial vectors (IV) for refreshing the key which imposes extra clock cycles for initializing the cipher process during the algorithm startup or whenever the key changes. The number of

cycles for initialization is independent from the size of the data block and varies between different designs.

In table 5-6, the number of clock cycles required for initialization for each cipher is depicted in parenthesis. For RBS designs, the key initialization happens once for 132 clock cycles. The result is kept as the authentication key and bitwise addition of this key with the input will be applied to the MAC generator after the process of initialization of the MAC. Since initialization happens once for several messages this overhead is not considered for performance comparison. However, these initial vectors for stream ciphers open new opportunities for attackers since the generated key is a function of initial vectors.

Computation Cycles

The second part of performance metric is the computation cycles which states the number of cycles required to encrypt the message. Despite the initialization, this one depends on the length of the input message. Table 5-9 displays the number of cycles required for computation for different size of data blocks in different cryptosystems.

Table 5-9 Comparison of clock cycles to encrypt

Cipher	Init	16 bits	32 bits	48 bits	64 bits	96 bits	128 bits
HB-2	16	16	32	48	64	96	128
Grain-128	513	16	32	48	64	96	128
Trivium	1333	16	32	48	64	96	128
PRESENT-80	0	32	32	32	32	64	64
PRESENT-128	0	32	32	32	32	64	64
AES-128	0	160	160	160	160	160	160
RBS-83	43	87	87	174	174	261	348
RBS-100	52	104	104	104	208	208	312
RBS-116	60	120	120	120	240	240	360
RBS-132	68	136	136	136	136	272	272

For HB-2, Grain and Trivium, the number of clock cycles increases same as increasing the size of data block. While in PRESENT, for data blocks equal with or smaller than 64 bits, fixed clock cycles are needed since in this cryptosystem, the number of rounds, the size of plaintext and ciphertext are fixed to 32, 64 and 64 respectively. In AES, like PRESENT, the number of clock cycles for data blocks equal with or smaller than 128 bits is fixed to 160 cycles.

In RBS, since each design works on a specified fixed size of message, depends on the chosen design, the number of clock cycles for the same size of data blocks will change. For instance, to encrypt a 48-bit message, in RBS-83 it takes 174 cycles while in RBS-100 it takes 104 cycles. Thus, it is required to choose the right design based on the size of message before implementing it. For messages equal with or shorter than 40, 48, 56 and 64, it would be better in term of timing and size of ciphertext to use RBS-83, RBS-100, RBS-116 and RBS-132 respectively.

Comparing all designs in table 5-9, for each size of data block, RBS has the highest number of clock cycles. However, in this comparison, it must be stated that except RBS, all other designs do not computes the authentication code. Table 5-10 shows the total number of cycles required to produce the ciphertext plus authentication code for different cryptosystems when the size of the plaintext is 64 bits.

Table 5-10 Number of required cycles for encryption 64-bit plaintext plus authentication

	Encryption [Cycles]	**Authentication [Cycles]**	**Total [cycles]**
HB-2	64	64	128
Grain-128	64	64	128
H-PRESENT	32	32	64
RBS-132	64	68	132

Comparing the results in table 5-10, it shows that PRESENT has the lowest number of cycles while RBS needs the highest number of cycles to complete its process. However, RBS-132 needs only 8 clock cycles more than HB-2, Grain which means 6% more clock cycles. This difference is resulted of the difference between the size of MACs and extra operations for preparing authentication keys.

Bits-per-Cycle

One of helpful metrics in performance, derived from computation time is bits-per-cycle. For stream ciphers, bits-per-cycle means the number of bits of output keystream per clock cycle. However, this definition can be expanded for all ciphers to describe their output rate. Therefore, to cover block ciphers and hybrid cyphers, this definition is modified to the number of bits of output divided by the number of cycles per each block [48]. In this metric, the one has the higher bits-per-cycle is the better one.

Table 5-11, based on bit-per-cycle metric, compares the number of bits, encrypted with different ciphers for various amounts of data when authentication is not included. Among all ciphers in this table, AES has the lowest rate of output while PRESENT has the highest one. Since, none of competitors provide MAC in their output, for RBS, in table 5-11, the redundant part (MAC) is not considered as a part of the output and it is supposed that the length of new output is equal with the length of generated keystream not the ciphertext.

Table 5-11 Bits-per-clock without authentication

Cipher	Block size [bits]	Output size [bits]	Clock Cycles	Bits per cycle
HB-2	16	16	16	1
Grain-128	1	1	1	1
Trivium	1	1	1	1
PRESENT-80	64	64	32	2
PRESENT-128	64	64	32	2
AES-128	128	128	160	0.8
RBS-83	40	83	87	0.46
RBS-100	48	100	104	0.46
RBS-116	56	116	120	0.47
RBS-132	64	132	136	0.47

Table 5-12 compares bits per clock while encryption is along with authentication. To have a fair comparison, it is supposed that all ciphers use 64-bit data block to encrypt. For Grain and HB-2 ciphers, the rate of number of produced output bits to the number required clock cycles for generating them is same when the authentication is included and when it is not. In this comparison, RBS cipher is very close to ciphers HB-2 and Grain.

Table 5-12 Bits-per-clock with authentication

Cipher	Plaintext [bits]	# of Clock for Enc.	MAC [bits]	# of Clocks for MAC	Bits per clock
HB-2	64	64	64	64	1
Grain-128	64	64	64	64	1
H-PRESENT	64	32	128	32	3
RBS-132	64	66	68	70	0.97

Max. Clock Frequency

Connections between inputs outputs and registers form a timing path between them. Among these paths, the slowest path in the design is known as the critical path which defines the upper bound on the clock frequency. The operating clock frequency of a design is usually

at a significantly lower rate than maximum frequency. In this metric, the one with the higher max frequency is the better one.

Maximum frequency for each cipher is displayed in Table 5-13 along with the used technology. For HB-2 and PRESENT no maximum frequency in ASIC implementation has been reported.

Table 5-13 Maximum clock frequency

Cipher	Tech. [nm]	Max. Freq. [MHz]	Normalized Freq. [MHz]
HB-2	130	N/A	N/A
Grain-128	130	925.9	925.9
Trivium	130	358.4	358.4
PRESENT-128	180	N/A	N/A
AES-128	130	130	130
RBS	90	5000	2500

To find the maximum frequency, for all RBS designs, the operating frequency in simulation was increased to 5 GHz while the time slack was still positive. Maximum frequency depends on the used technology. In newer technology, the size of capacitance loads is scaled down. Therefore, the circuit can work in higher frequency. To compare all ciphers for this metric, it is required that all ciphers are implemented with the same technology or the frequency become normalized with the technology scaling, α.

In Table 5-13, after normalization, RBS ciphers still have the highest maximum frequency among all ciphers. The reason is having simple circuit with short paths in their implemented hardware. On contrary, AES has the lowest maximum frequency among the ciphers.

Throughput

Throughput is the rate of producing the new output with respect to time, typically expressed in bits-per second [48]. This metric reaches to its sustainable rate when the initialization is completed at a given operating clock frequency. So, it is simply calculated through multiplying bits-per-cycle by the clock frequency. To have a fair comparison, it is required that all competitors work in the same operating clock frequency. For this metric, the cipher that has higher throughput is the better cipher.

Figure 5-4 compares throughput of all ciphers when the operating frequency for all ciphers is set to 10MHz and ciphers do not provide the authentication service. In this figure, PRESENT has the highest throughput among all ciphers since it has the highest bits-per-cycle. On the contrary, RBS has the lowest throughput in this comparison because of its low bits-per cycle.

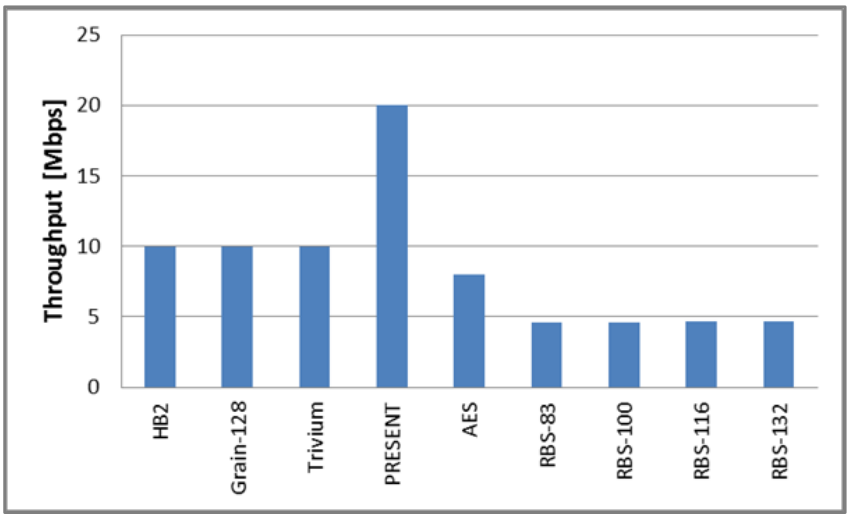


Figure 5-4 Throughput when the operating frequency is 10MHz without authentication

Figure 5-5 compares the throughput of ciphers when the operating frequency is set to be 10MHz when all ciphers provide the authentication service too. In this comparison,

PRESENT still has the highest throughput. However, the throughput of RBS is very close to HB-2 and Grain ciphers

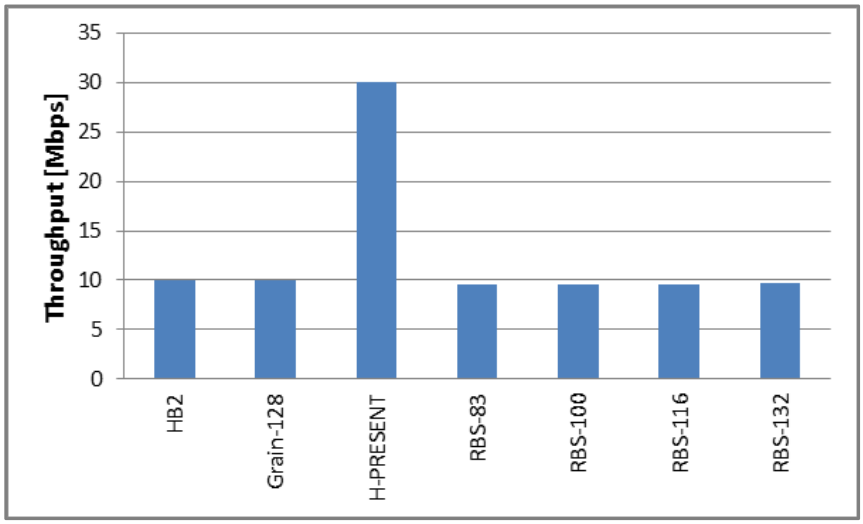


Figure 5-5 Throughput when the operating frequency is 10MHz with authentication

The maximum throughput will occur at the maximum clock frequency. In this metric, the cipher that has the higher maximum throughput is the better cipher. Table 5-14 compares the maximum throughput among all ciphers. Despite the low bits-per-cycle, RBS ciphers have the highest maximum throughput because of having highest maximum frequency among ciphers.

Table 5-14 Maximum throughput

Cipher	Bits per cycle	Max. Freq. [MHz]	Max Throughput [Mbps]
HB-2	1	N/A	N/A
Grain-128	1	925.9	925.9
Trivium	1	358.4	358.4
PRESENT-128	2	N/A	N/A
AES-128	0.8	130	104
RBS-83	0.46	2500	1150
RBS-100	0.46	2500	1150
RBS-116	0.47	2500	1175
RBS-132	0.47	2500	1175

5.2.3 *Area-Time Product*

The size of hardware and performance are two important metrics which have been studied in this chapter separately. Area-Time product is a cost function which is equal with the product of the time taken to produce each new output bit and the area of the design [48]. To have the optimal value for this metric it is required that the implemented hardware has the low overhead in area while it provides the high speed in producing the output. This metric is expressed in gate equivalent by second [GE-µs]. In this metric, the one has the lower product is the better one. Table 5-15 demonstrates this metric for each cipher when the operating frequency is set to be 10MHz and ciphers do not provide the authentication service.

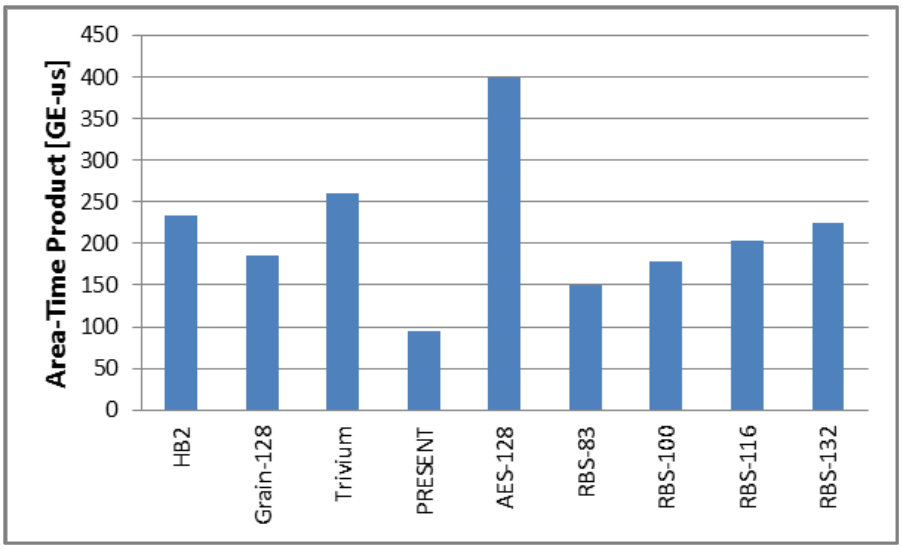


Figure 5-6 Area-Time product when authentication is not provided

In Figure 5-6, PRESENT has the optimal Area-Time product among all ciphers. After that, Grain and RBS ciphers have the best AT products.

Figure 5-7 shows the AT product for ciphers when the operating frequency is 10MHz and all ciphers provide the authentication service as well. Among all of these ciphers, RBS ciphers have the best AT product.

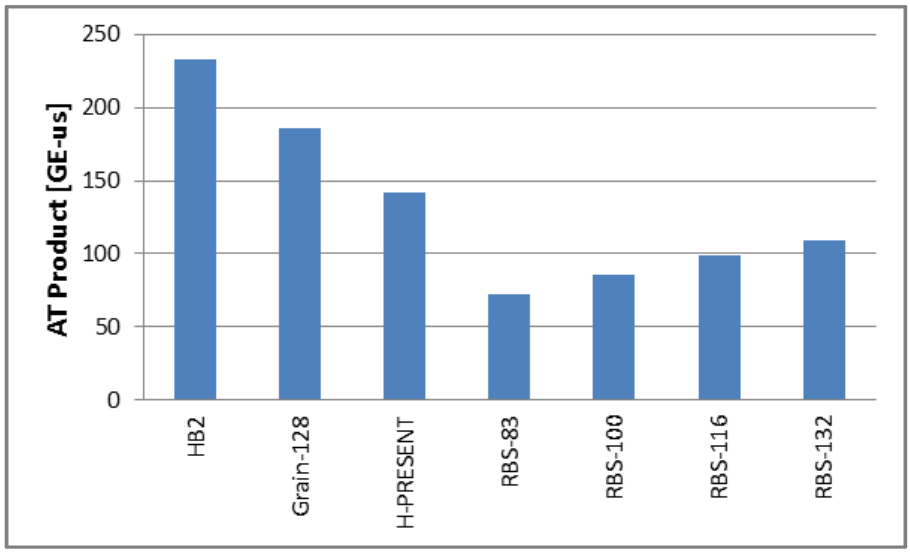


Figure 5-7 Area-Time product when authentication is provided

5.2.4 Hardware Efficiency

The total hardware performance cannot be determined by only measuring gate counts or performance. The hardware efficiency is defined as throughput per gate which states the balance between the size of implemented hardware and its speed [48]. It is simply calculated by dividing the throughput by the area overhead. Hardware efficiency is expressed in Mbits per second per gate equivalents [Mbps/GE]. To have the highest hardware efficiency, it is required that the optimal balance exists between the throughput and the size of hardware. In this metric, the one has the higher efficiency is the better one.

Figure 5-8 demonstrates the hardware efficiency when the operating frequency is set to 10MHz and ciphers do not provide the authentication service. In this comparison,

PRESENT has the highest hardware efficiency among all ciphers. After that, RBS designs and Grain has the best rate in hardware efficiency.

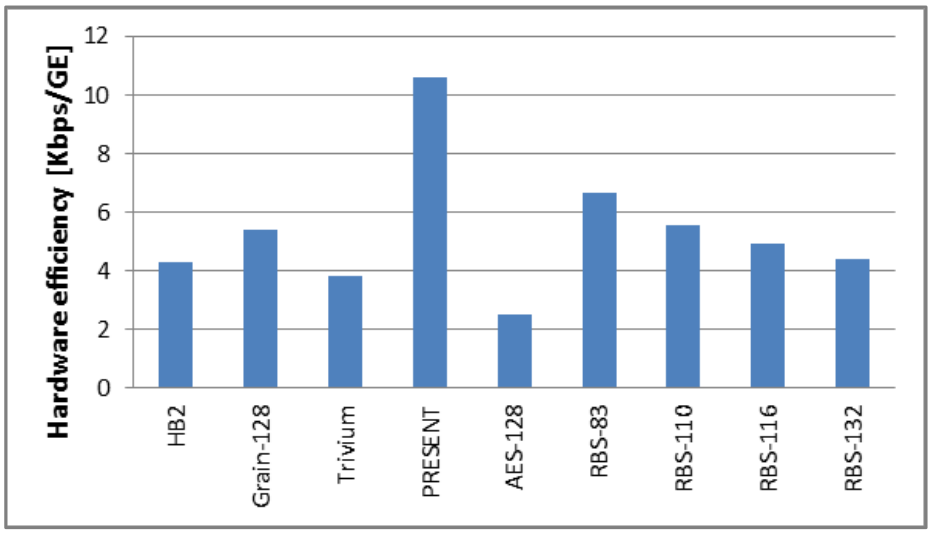


Figure 5-8 Hardware efficiency when frequency is 10MHz and without authentication

Figure 5-8 demonstrates the hardware efficiency when the operating frequency is 10MHz and ciphers provide the authentication service. RBS-132 has the highest hardware efficiency among all ciphers.

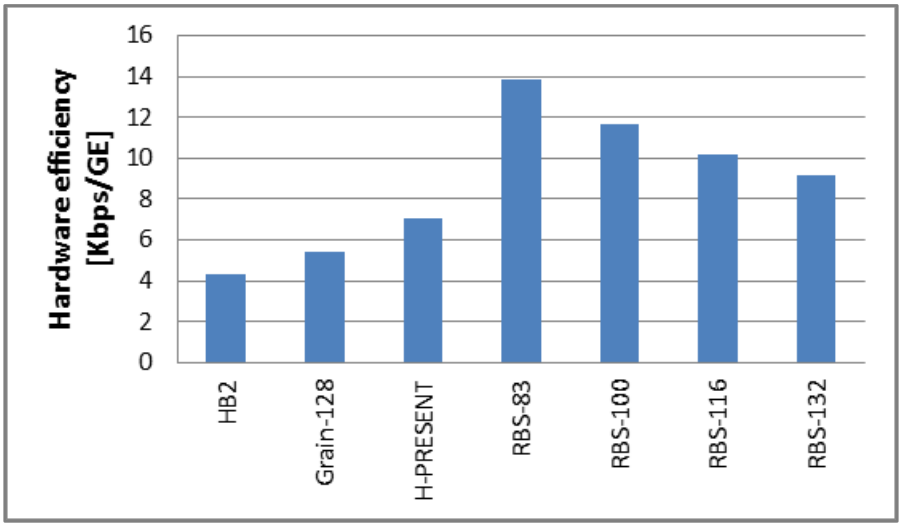


Figure 5-9 Hardware efficiency when frequency is 10MHz and authentication is provided

5.2.5 Power

This metric demonstrates the required power for computation at each clock cycle. In this metric, the one needs the lower power is the better one. The total power consumption obtains by adding up static power and dynamic power. To compare total power dissipation in different ciphers, it is necessary that all designs have been fabricated at the same technology and also work at the same operating frequency. Since both power components also depend on supply voltage, the typical core voltage should be used. However, it is difficult to satisfy these conditions for all competitors. Measured power can be scaled with an acceptable margin of error to other frequencies and technologies if the static and dynamic components are treated separately. With this assumption, dynamic power can be assumed that it is directly proportional with the frequency and inversely with technology coefficient scaling, α.

At low frequency the static power is significant whereas at the other frequencies it may be trivial. Unfortunately, there is not any straight formulation for normalizing static power like dynamic power. However, since the static power takes a small percentage of the total power, for different technologies these effects are assumed to be negligible and are not considered for comparison. So the total power of ciphers in different frequencies can be approximately normalized by multiplying it with the coefficient of frequency growth and multiplying with α^2 when the used technologies are different. It is not accurate but it gives a good estimation for comparison.

Based on above assumptions, it can be estimated that the power report in RBS must be doubled in order to be comparable with other compared designs in 130nm which is still lower than other designs' power consumption. In contrast, the power consumption of

PRESENT cipher is needed to be multiplied by 10 because of frequency and then divided by technology coefficient scaling, $\alpha = (180/130)^2$ or 1.9 to be normalized for comparison. Figure 5-10 shows the normalized total power consumption for all ciphers when the operating frequency is 10MHz and the used fabrication technology is 130 nm. In this table, all RBS designs have the lowest power consumption among all ciphers.

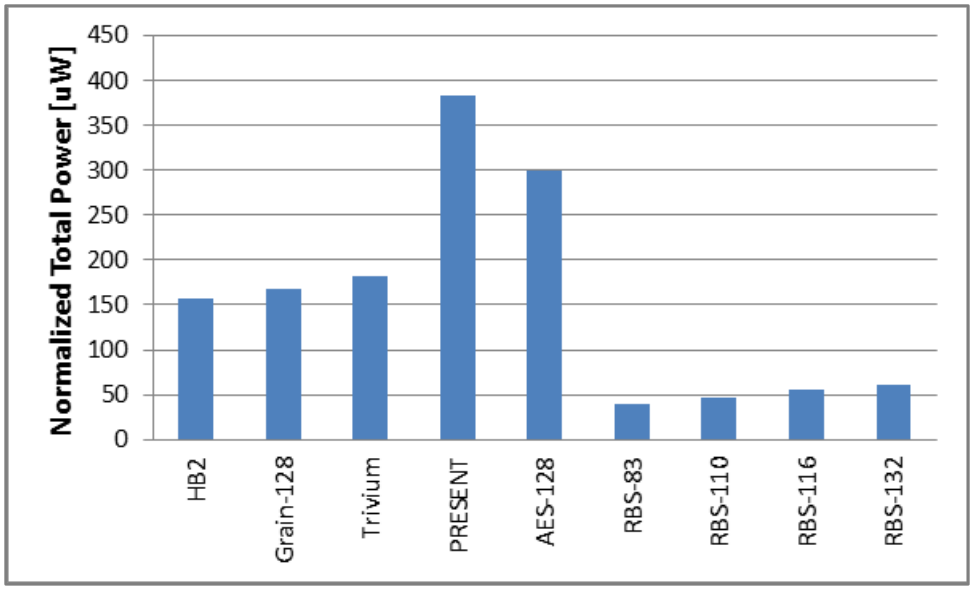


Figure 5-10 Power consumption without authentication

The normalized power consumption for ciphers at 130nm and frequency of 10MHz when the authentication is provided is shown in Figure 5-11. In this figure, the power consumption for Grain and HB-2 is same as in the last table since they use the same resource with no overhead in area. The power consumption of H-PRESENT is the average of power consumption required for encryption (7.34 µW) and power needed for authentication service (8.09 µW). The power consumption of RBS-83, RBS-100 and RBS-116 ciphers is increased in this figure since these ciphers perform encryption and authentication for 64-bit plaintexts.

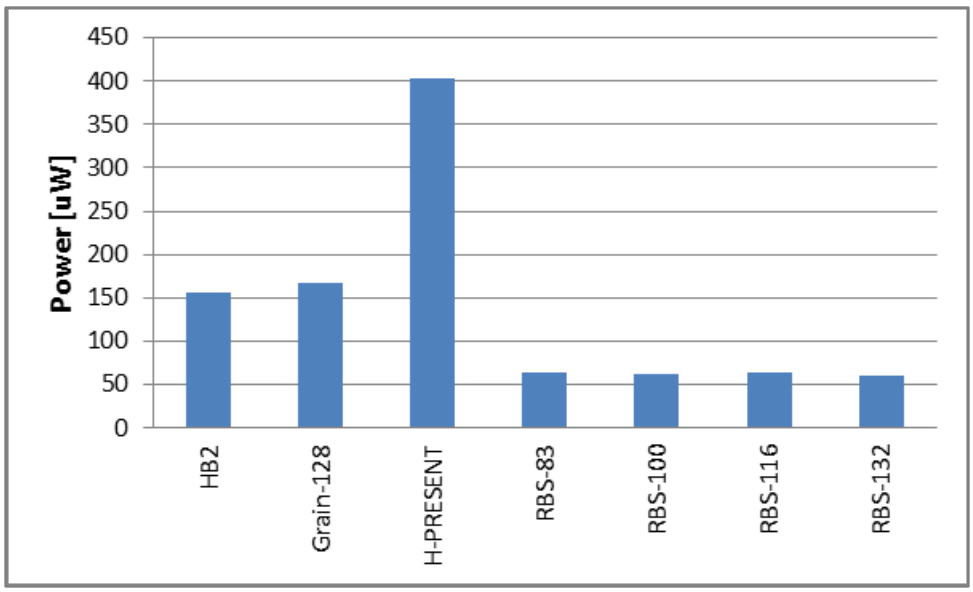


Figure 5-11 Power consumption for 64-bit plaintext when authentication is provided

5.2.6 Energy

In battery operated devices, the power consumption may not be a good metric for comparison. Instead, the amount of energy needed for operations may be a more useful criterion because a battery stores a limited amount of energy, not power [48]. Two other metrics, power consumption and performance are the basics of this metric. Energy is defined by the total power required to accomplish an operation in a given time. It is calculated by multiplying the power by taken time and expressed in joules. In this metric, the one which has the lower energy is the better one.

Table 5-15 shows the required energy for encrypting 64-bit plaintext in different ciphers when the operating frequency is 10MHz while except RBS; none of ciphers provide the authentication service. It should be noticed that since AES works on 128-bit data blocks, here to have a fair comparison, it is supposed that AES is encrypting two 64-bit plaintexts. In

this comparison, RBS with 0.82 pJ needs the lowest energy while AES needs the highest energy, three times more than RBS-132.

Table 5-15 Energy required for encryption of a 64-bit plaintext without authentication

Cipher	Normalized total power [µW]	Clock Cycles	Energy [nJ]
HB-2	156.8	64	1.0035
Grain-128	167.73	64	1.0734
Trivium	181.18	64	1.1595
PRESENT-128	382.86	32	1.2251
AES-128	300	160	2.4
RBS-132	60.92	136	0.8285

Table 5-16 shows the required energy for encrypting a 64-bit plaintext when the operating frequency is set to 10MHz and all ciphers provide authentication beside confidentiality. The results show that HB-2 and Grain need 2.5 times more energy than RBS-132 to provide the authentication. In this comparison, H-PRESENT uses the highest energy to encrypt and authenticate a 64-bit plaintext. On the contrary, RBS-132 needs the lowest energy to perform both services compared to other ciphers.

Table 5-16 Energy required for encryption and authentication of a 64-bit plaintext

Cipher	Normalized total power [µW]	Clock Cycles	Energy [pJ]
HB-2	156.8	128	200.7
Grain-128	167.73	128	214.68
H-PRESENT	402.42	64	257.5
RBS-132	60.92	136	82.85

5.2.7 Energy-per-Bit

Energy-per-bit is calculated by dividing the total power consumption by the throughput. It is required to calculate this metric when both power and throughput are measured at the same operating clock frequency. It seems that this metric is independent of

frequency. However, in low frequencies, the static power may dominate the total power consumption. While, at higher frequencies, dynamic power has a substantial effect on it [48]. In this metric, the one which has the lower energy-per-bit is the better one. Figure 5-12 shows the energy-per-bit required for each cipher when the operating frequency is 10MHz and the used technology is 130nm when all ciphers do not provide the authentication service. Since the measured power for all ciphers does not satisfy these conditions, the normalized power from table 5-10 is used for this comparison.

In Figure 5-12, RBS ciphers have the lowest energy per bit while it has the lowest throughput. On the contrary, PRESENT has the highest energy-per-bit after AES while PRESENT has the highest throughput among all ciphers.

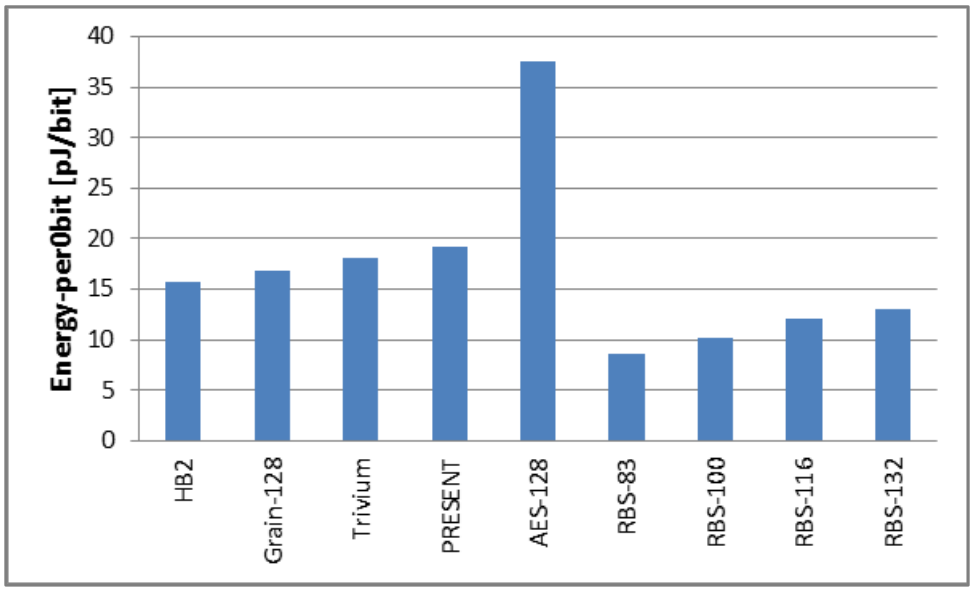


Figure 5-12 Energy-per-bit without authentication

Table 5-24 shows the energy-per-bit required for each cipher when the operating frequency is 10MHz and the used technology is 130nm when all ciphers provide the

authentication service. In this comparison, RBS ciphers again have the lowest energy-per-bit while other ciphers need at least 2.5 times more energy-per-bit than RBS.

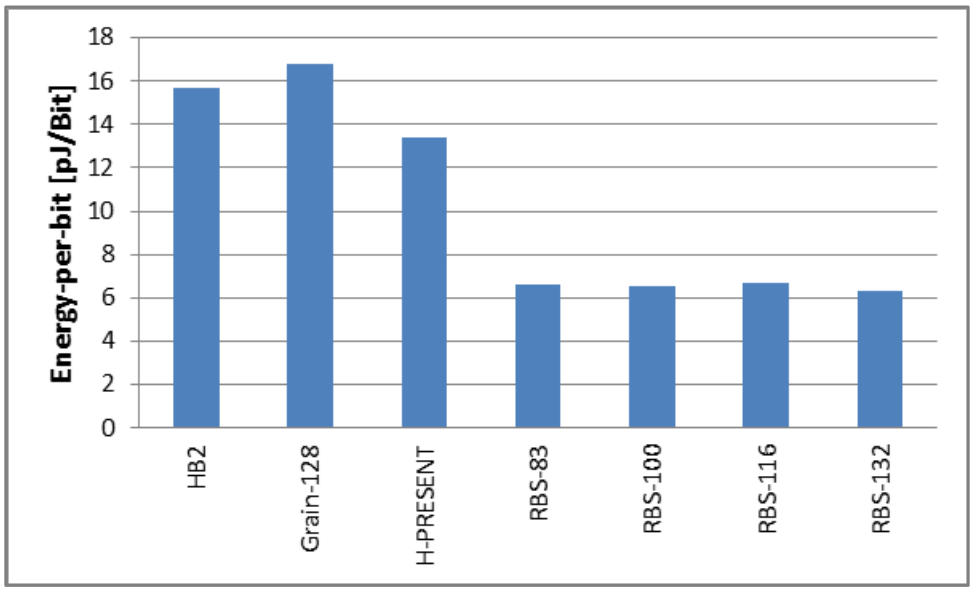


Figure 5-13 Energy-per-bit with authentication

Comparing Figure 5-12 and Figure 5-13, energy-per-bit for RBS ciphers in Figure 5-13 is about 50% less than in Figure 5-12. The reason is counting out redundant bits in the number of output bits which has direct effect on bits-per-cycle.

5.2.8 Trade-offs

For the future wireless network application, battery life, throughput and area are three most important metrics to the designer [48]. Therefore, considering the trade-off between the Energy per bit and Throughput/Area metrics may be a good measure for comparing designs.

Figure 5-14 shows the energy-per-bit metric versus hardware efficiency of nine ciphers when the frequency is 10MHz and all competitors do not provide the authentication service. In this comparison, the best ciphers are located in the most left and most up in the

figure which need less energy-per-bit while they provide higher hardware efficiency. Figure 5-14 shows that RBS ciphers have the best energy-per-bit while their hardware efficiency is medium. On contrary, PRSENT with a big difference has the best Hardware efficiency while its energy-per-bit is worse than other ciphers except AES. Among all ciphers, AES is the worst ciphers in terms of both metrics.

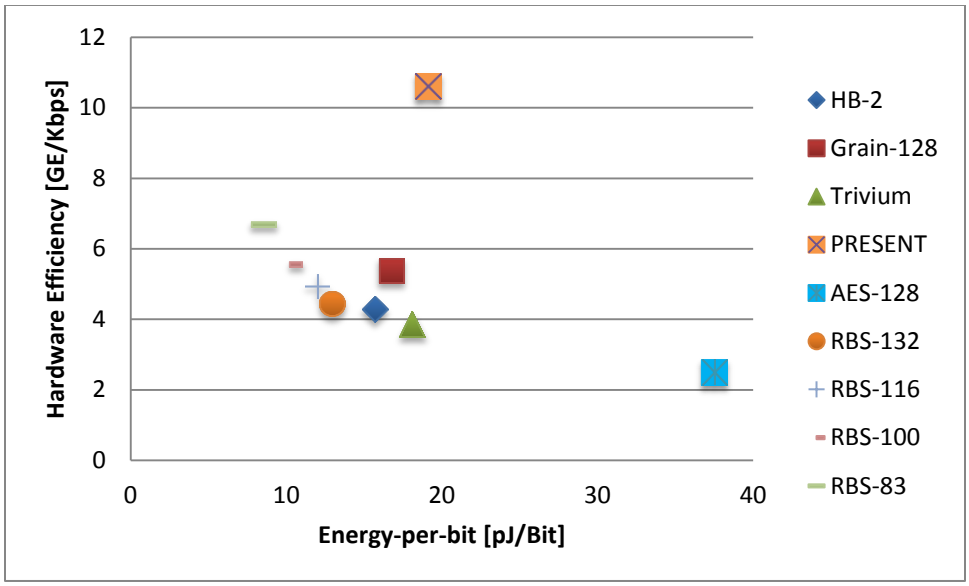


Figure 5-14 Energy-per-bit vs. hardware efficiency without authentication

Figure 5-15 shows the energy-per-bit metric versus hardware efficiency of four ciphers when the frequency is 10MHz and they all provide the authentication service for 64-bit plaintexts. In this figure, RBS is the best in both metrics while there is a big difference between RBS and other ciphers in both two metrics.

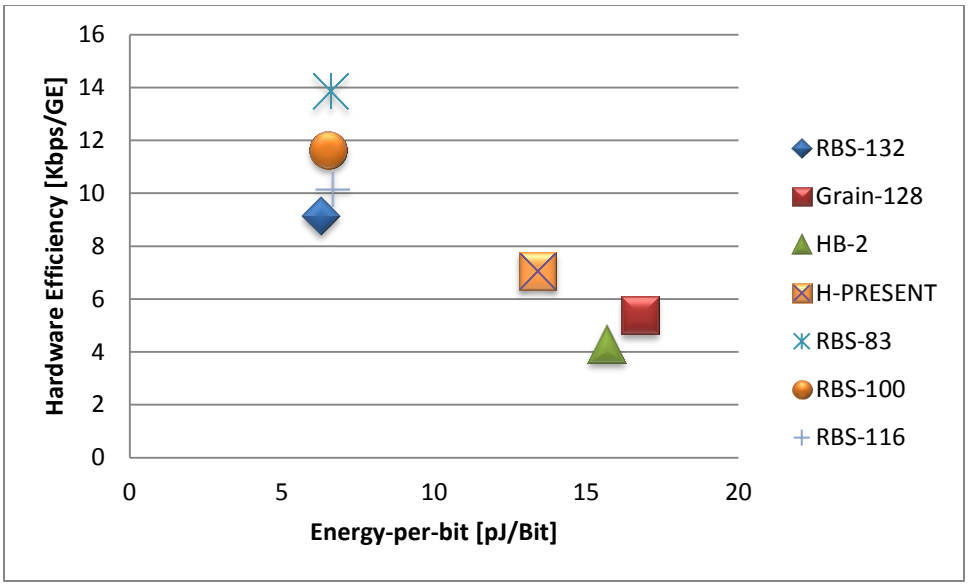


Figure 5-15 Energy-per-bit vs. hardware efficiency with authentication

5.2.9 Power-Area-Time Product

Power-Area-Time metric is a triple product calculated by Area-Time product and power consumption. It is expressed in [nJ-GE]. To have a better result in this metric, it is required that have a cipher with small size in hardware which consumes low power consumption with high performance [48]. In this metric, the cipher which has the lower product is the better cipher. Figure 5-16 shows the results for Power-Area-Time product when the operating frequency is set to10MHz and all competitor ciphers do not provide authentication service.

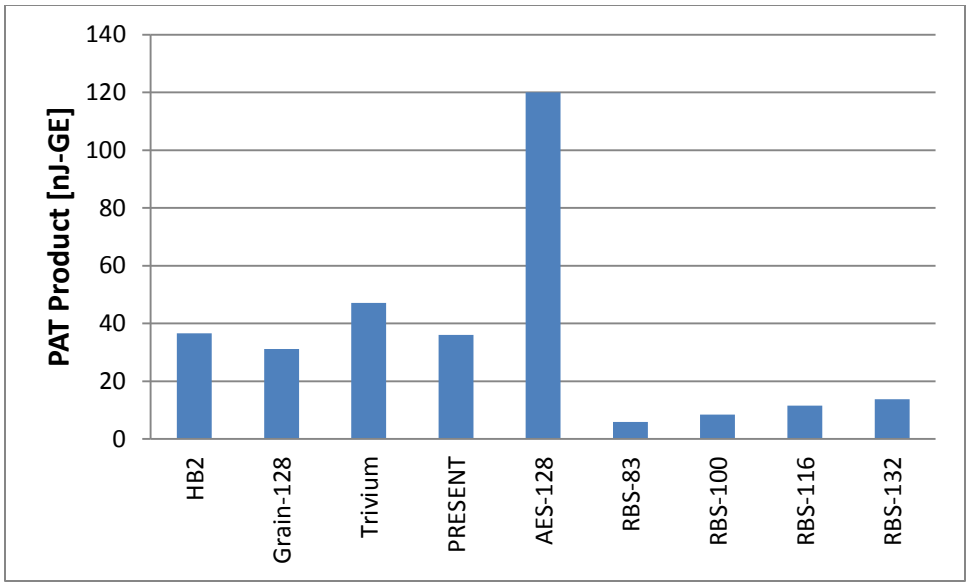


Figure 5-16 Power-Area-Time product when frequency is 10MHz without authentication

In this metric, RBS ciphers have the lowest product among all competitors, while AES has the highest one.

Figure 5-17 shows the results for Power-Area-Time product when the operating frequency is set to10MHz and all ciphers provide the authentication service. In this metric, RBS ciphers have the lowest product among all other ciphers, while, there is a huge difference between RBS ciphers and other ciphers in this metric.

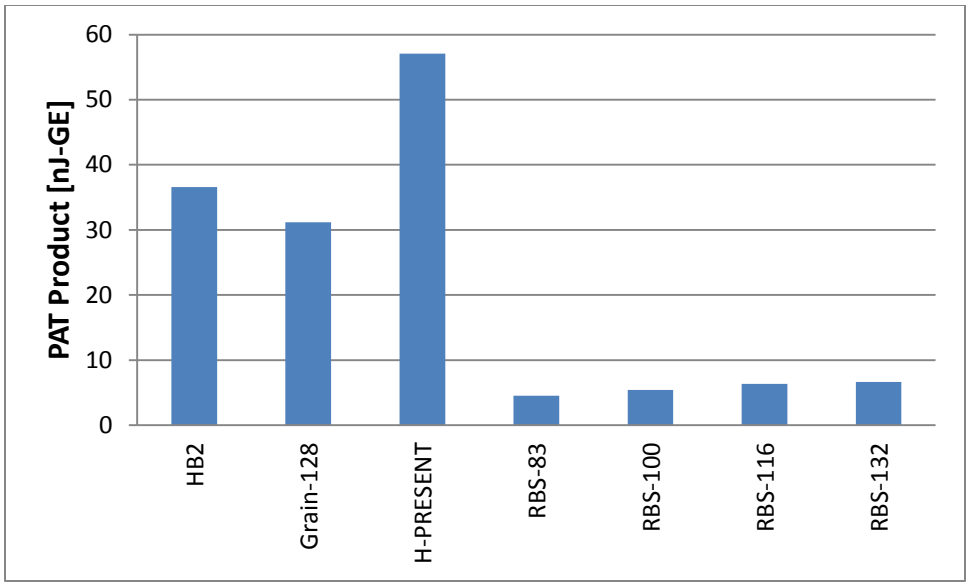


Figure 5-17 Power-Area-Time product when frequency is 10MHz with authentication

5.3 Conclusion

In this chapter, the result of hardware implementation of RBS ciphers was compared with some well-known lightweight ciphers designated for RFID systems which their security are promised and also have low cost and overhead in area and power consumption for their implementation. These ciphers had already been studied in CHAPTER 2 including HB-2, Grain, Trivium, PRESENT and AES. AES cipher was studied as a benchmark here.

Since RBS is an authenticated cipher, the comparison was investigated in two categories. First, when all ciphers except RBS do no provide the authentication service. Second, when all ciphers in the comparison provide the authentication service along with confidentiality. Among these ciphers AES and Trivium do not have any dedicated hash function or authentication part and they need to be integrated with other MAC algorithms to provide this service while implementing them imposes a huge overhead in area and performance. Therefore, in the second category, these two ciphers did not get participated

and with four ciphers, HB-2, Grain, H-PRESENT and RBS this comparison was progressed. To have a fair competition in the second group, it was supposed that all ciphers encrypt and authenticate 64-bit plaintexts.

Table 5-17 give a summary of all one-dimensional metrics like area, bit-per-cycle, throughput, maximum throughput and estimated power consumption and also multi-dimensional metrics like required energy for encryption of a 64-bit plaintext, energy-per-bit, Area-Time product, hardware efficiency and Power-Area-Time product. In table, the results are estimated in 130nm technology and 10MHz operating frequency when ciphers do not provide the authentication service.

Table 5-17 Summary of normalized metrics without authentication

	Area [GE]	Bits per Cycle	Max. Throug. [Mbps]	Throughput [Mbps]	Estimated Power [µW]	Energy/Bit [pJ/Bit]	Area-Time Product [GE-µs]	Hardware Efficiency [Kbps/GE]	Power-Area-Time Product [nJ-GE]
HB-2	2332	1	N/A	10	156.8	15.68	233.2	4.29	36.56
Grain-128	1857	1	925.9	10	167.73	16.77	185.7	5.38	31.147
Trivium	2599	1	358.4	10	181.18	18.11	259.9	3.85	47.088
PRESENT	1884	2	N/A	20	382.86	19.13	94.2	10.61	36.065
AES-128	3200	0.8	104	8	300	37.5	400	2.5	120
RBS-83	688	0.46	1150	4.6	39.34	8.55	149.6	6.69	5.88
RBS-100	826	0.46	1150	4.6	47.1	10.24	179.2	5.57	8.44
RBS-116	951	0.47	1175	4.7	56.5	12.02	203.5	4.94	11.5
RBS-132	1061	0.47	1175	4.7	60.92	12.96	225.4	4.43	13.73
Better is:	Lower	Higher	Higher	Higher	Lower	Lower	Lower	Higher	Lower

At table 5-17, PRESENT is first cipher at four metrics, bits-per-cycle and throughput, Area-Time product and hardware efficiency while RBS is best at the remaining metrics such as area, maximum throughput, power consumption, energy-per-bit and Power-Area-Time product. In conclusion, when the authentication is not required, RBS ciphers have the best

results in most of the performance metrics for being selected as a candidate cipher. However, considering the energy required for transmitting the extra bits as MAC along with the ciphertext, HB-2 is the best cipher in this condition since it needs the least power consumption. On contrary, PRESENT has the best throughput among all ciphers. However because of its high power consumption and energy-per-bit, compared to other ciphers PRESENT is not a good choice.

Table 5-18 gives a summary of all one-dimensional metrics such as area, bit-per-cycle, throughput, estimated power consumption and also multi- dimensional metrics like required energy for encryption of a 64-bit plaintext and providing its authentication, energy-per-bit, Area-Time product, hardware efficiency and Power-Area-Time product. In this table, the results are estimated in 130nm technology and 10MHz operating frequency when ciphers when all ciphers provide the authentication service besides confidentiality.

Table 5-18 Summary of normalized metrics with authentication

	Area [GE]	Bits per Cycle	Throughput [Mbps]	Estimated Power [µW]	Energy [pJ]	Energy/Bit [pJ/Bit]	Area-Time Product [GE-µs]	Hardware Efficiency [Kbps/GE]	Power-Area-Time Product [nJ-GE]
HB-2	2332	1	10	156.8	200.7	15.68	233.2	4.29	36.56
Grain-128	1857	1	10	167.73	214.6	16.77	185.7	5.38	31.147
H-PRESENT	4256	3	30	402.42	257.5	13.41	141.8	7.05	57.06
RBS-83	688	0.95	9.54	62.94	87.61	6.6	72.12	13.86	4.54
RBS-100	826	0.96	9.61	62.8	87.1	6.53	85.95	11.63	5.397
RBS-116	951	0.97	9.66	64.57	88.55	6.68	98.45	10.16	6.357
RBS-132	1061	0.97	9.7	60.92	82.85	6.28	109.3	9.14	6.658
Better is:	Lower	Higher	Higher	Lower	Lower	Lower	Lower	Higher	Lower

At table 5-18, when all ciphers provide the authentication plus confidentiality, H-PRESENT is best at just two metrics, bits-per-cycle and throughput while RBS ciphers are the best at the rest of metrics. In this table, the only metrics that RBS ciphers do have best results are bits-per-cycles and throughput. However, the result of these two metrics in RBS is very close to HB-2 and Grain.

Comparing the results in the last two tables, shows that RBS ciphers are better than other ciphers especially when providing the authentication along with confidentiality is required. However, using the RBS cipher is recommended when the size of plaintext is between 40 to 64 bits and the required space key is between 2^{80} to 2^{128}. For environments that need very short plaintext with high key space, RBS may not be a good choice. Because, to provide this space key, it is required to either insert a big number of redundant bits or select a big size block of data. Both solutions will impose a lot of overhead in hardware implementation and also transmission the output. On contrary, stream ciphers like Grain and Trivium are very good choice when the size of plaintexts is very small like few bits and key size is large. However, using these ciphers without providing the authentication service is not a proper choice since if one or more bits of the ciphertext flipped during the transmission either by accident or by attacker, the integrity of the generated ciphertext cannot be verified at the reception. Trivium does not have any dedicated hardware for this purpose, so it is not recommended at all.

Until now, all studied performance metrics were related to hardware implementation results which are required for computation of the ciphertext. However, besides these metrics, extra resources are also required for transmitting the ciphertext which depends on the length of the ciphertext. Although, experimental results show that RBS ciphers are the best cipher

for computing ciphertext along with authentication, it is required to compare it with other ciphers for new metric, length of output which is counted as an overhead in transmitting messages.

The length of output in cryptosystems is equal with the length of ciphertext plus the length of MAC if it is provided. Adding the MAC to the ciphertext for sake of providing authentication service, will make the output longer than before. Since, for transmitting each bit of the message, an extra power is required, therefore the total energy required for transmission the authenticated message will be higher than when authentication is not provided. Table 5-19 shows the length of output for different ciphers with different input sizes. For each entry, the first number shows the length of ciphertext and the second one shows the length of the MAC.

Table 5-19 The size of ciphertext for different input sizes when authentication is provided

	16 bit	**32 bit**	**48 bit**	**64 bit**	**96 bit**	**128**
HB-2	16+64	32+64	48+64	64+64	96+64	128+64
Grain	16+32	32+32	48+48	64+64	96+96	128+128
H-PRESENT	64+128	64+128	64+128	64+128	(64+128)*2	(64+128)*2
RBS	83	83	100	132	208	272

The relation between the size of plaintext and the size of the output composed of ciphertext and MAC is shown in figure 5-18. Base on this figure, PRESENT cipher generates the longest authenticated ciphertexts among all ciphers. Therefore because of the huge overhead in the length of MAC and also energy required for sending it out, PRESENT is not a good cipher at all for environments which authentication is necessary.

For plaintexts shorter than 40 bits, Grain has the shortest output while for plaintext longer than 64 bits HB-2 has the shortest output. For plaintexts between 40 and 64 bits HB-2,

Grain and RBS ciphers produce about the same size of output. It must be noticed that for different range of plaintext, different RBS designs are utilized.

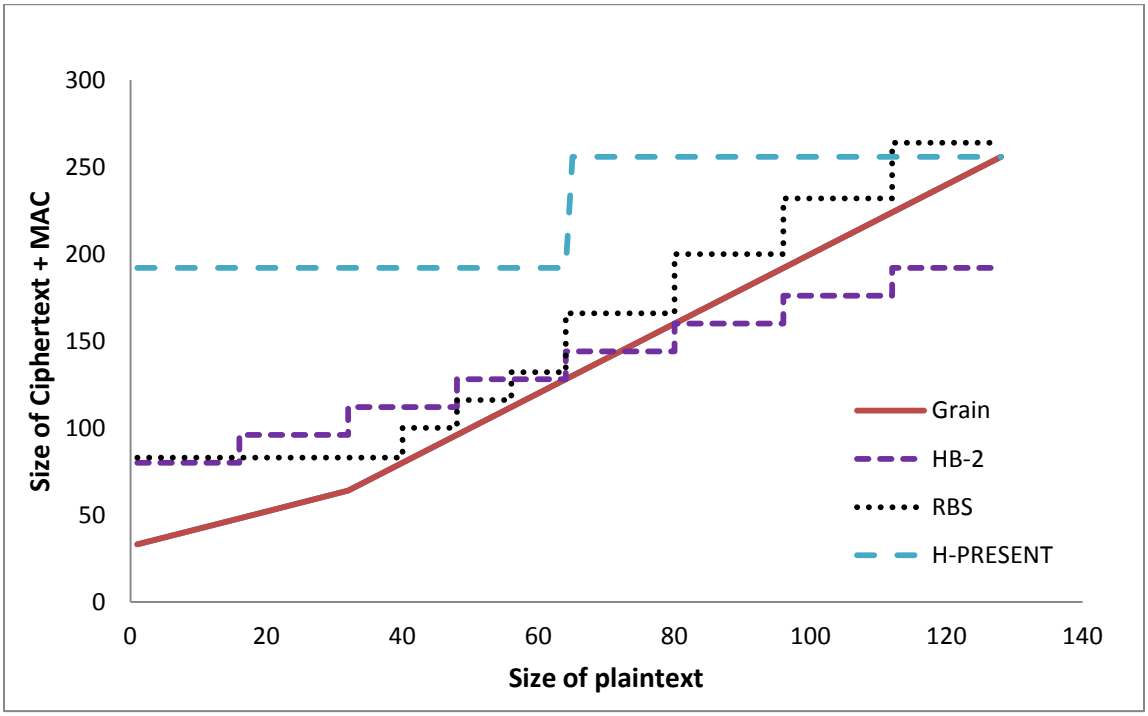


Figure 5-18 Size of output for different sizes of plaintext

In conclusion, when authentication is required, based on figure 5-8, it can be concluded that for messages shorter than 40 bits, Grain is a good choice and for messages longer than 64 bits, HB-2 is a good candidate because of short MAC and consequently short output they generate. However, for messages between 40 to 64 bits, RBS is a good cipher because the length of its output and also the energy required for sending it out is as same as HB-2 and Grain ciphers while the energy required for computing the output is very less than its competitors.

CHAPTER 6: Conclusions and Future Work

6.1 Conclusions

A new lightweight symmetric authenticated encryption cipher called RBS; Redundant Bit Security, is proposed for RFID systems. RBS provides authentication, integrity and confidentiality services at the same time with low overhead in area and power consumption. This algorithm presents block ciphers working with 40 to 64-bit data blocks and 83 to 132-bit keys.

In RBS algorithm, confidentiality of the plaintext is accomplished by changing the location of plaintext bits through inserting some redundant bits inside the plaintext. The location of redundant bits inside the ciphertext is the secret key shared between two parties. To distinguish plaintext from redundant bits, attacker will need accessing this key. Obviously, increasing either the size of plaintext or the number of redundant bits, the number of possible keys for attackers will increase. Therefore, these two parameters have important effect on the security strength of RBS. In other words, the security level of this algorithm is adjustable through these parameters.

Besides confidentiality, the redundant bits provide other services as well like integrity and authentication so the receiver can validate the data if data gets altered during transmission either by an adversary or environmental issues. To fulfill all these services, redundant bits are defined as MAC (message authentication code) of the plaintext.

The MAC algorithm in RBS is based on the purposed MAC in [56] as it supports different digest sizes. Supporting different digest sizes means different redundant data sizes and accordingly different security levels could be adjusted without changing the underlying

MAC algorithm. However there is a limitation using the MAC algorithm in [56] as it is mainly proposed for stream cipher systems. In order to adapt it with RBS block cipher algorithm, some modifications are performed on this authentication approach. Rather than adapting the algorithm, these modifications saved the considerable resources in the hardware in term of area and power in the cost of some marginal throughput performance.

Besides the redundant data, how the plaintext appears inside the ciphertext also contributes in the confidentiality of RBS. In RBS, the plaintext gets altered through bitwise addition of the plaintext bits with a variable keystream which itself is a function of the plaintext before merging with redundant data. This way, any change in the plaintext results a different keystream and consequently a different ciphertext. Like redundant data, this altered plaintext is also generated through MAC.

The security of RBS against powerful and well-known attacks such as known-plaintext attack, chosen-plaintext attack, chosen-ciphertext attack, differential attack, substitution attack, related key attack, linear cryptanalysis algebraic attack and cube attack has been proved in this thesis.

The main hardware component contributing in resource usage in RBS is MAC generator which is responsible to produce the redundant bits and the keystream. Rather than MAC, there are encryption/decryption components for inserting/separating redundant bits to/from the altered plaintext which are embedded in the sender/receiver. These two parts are composed of a multiplexer and de-multiplexer which can be implemented by few gates.

Compared to other proposed symmetric ciphers for RFID systems, RBS offers the lowest cost of area by decreasing it to 43% which is concluded from serialization input,

resource sharing and using simple elements such as XOR and multiplexers. Reducing the cost of area results 53% power saving with only 3% performance degradation. When authentication is not required, RBS is better than other ciphers in terms of area, power, energy, Area-Time product and Power-Area-Time product and the worst in performance and throughput. Also, the cost of energy for transmission is about two times more than other ciphers. When authentication is a must like hostile environments or payment systems, the performance and throughput of RBS is very close to other ciphers.

6.2 Future Work

Throughout this research, some ideas have been occurred into mind that may expand the scope of original goals and mitigate restrictions of RBS. This section provides an overview of possible ideas that could be followed in further work.

- One of the limitations in RBS is the length of the generated ciphertext. Making a hybrid version of RBS cipher, the value of MAC will depend on the initial vector so there would be no worry on MAC collision. In other words, the size of the MAC can be the same as the plaintext size like Grain cipher. However, the main challenge would be resistance of the algorithm against chosen-plaintext and IV attacks.
- Underling serialized architecture of RBS is the main source of its low throughput. Instead of processing one input bit at each clock cycle, two or more bits can be processed by adding some parallel hardware resources to the system. It is predicted that by increasing the cost of area into 30% to 40%, the performance can be approximately doubled.

- In RBS, the location of plaintexts changes by inserting the redundant bits inside the ciphertext while their order is intact. The security of RBS cipher can be improved considerably by changing the order of plaintext bits as it will increase the key space size, exponentially. The main challenge for this approach would be defining keys that store the information of bits orders as well as their location in the ciphertext.
- In the presented algorithm, the same key is used for generating authentication, keystream and encryption processes. Splitting the key into sub-keys, will increase the key space. However in order to be strong against related key attack, these keys are required to be divided and utilized somehow that any change at each of sub-keys has significant change in both generated redundant data and the keystream.

BIBLIOGRAPHY

[1] K. Finkenzeller: RFID Handbook, BOOK, John Wiley & Sons. 1999.

[2] http://www.idtechex.com.

[3] B. Glover, H. Bhatt, "RFID Essentials", O'Reilly, 2006.

[4] E. Zeisel, R. Sabella, "RFID+ Exam Cram", Que, 2006.

[5] A. Mitrokotsa, M. R. Rieback, A. S. Tanenbaum, "Classifying RFID attacks and defenses", Information Systems Frontiers, Springer, November 2010, Volume 12, Issue 5, pp 491-505.

[6] S. A. Weis, S. E. Sarma, R. L. Riovest, and D. W. Engels, (2004), "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems", In D. Hutter et al. (Eds.): Security in Pervasive Computing 2003, LNCS, volume 2802, Springer-Verlag, pages 201–212.

[7] Y.Y. Chen, M.L. Tsai, "The Study on Secure RFID Authentication and Access Control", Book, 2011, ISBN: 978-953-307-356-9, InTech, DOI: 10.5772/20750.

[8] S. E. Sarma, S. A. Weis, D.W. Engels, "Radio-frequency identification systems", CHES '02, pages 454–469. Springer-Verlag, 2002. LNCS no. 2523.

[9] S. Garfinkel, B. Rosenberg, "RFID; Applications, Security, and Privacy", Upper Saddle River, NJ: Addison-Wesley, 2006.

[10] A. Juels, R.L. Rivest, M. Szydlo, "The blocker tag: Selective blocking of RFID tags for consumer privacy", In V. Atluri, editor, 8th ACM Conference on Computer and Communications Security, pages 103–111. ACM, Press, 2003.

[11] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, “Handbook of Applied Cryptography”, CRC Press, 1997. Available online at http://www.cacr.math.uwaterloo.ca/hac.

[12] M. Feldhofer, C. Rechberger, “A Case against Currently Used Hash Functions in RFID Protocols”, In First International Workshop on Information Security (IS’06), volume 4277 of LNCS, pages 372-381, Springer-Verlag, 2006.

[13] A. Poschmann, “Lightweight Cryptography: Cryptographic Engineering for a Pervasive World”, Ph.D. Thesis 2009.

[14] K.S. Lee, J.H. Chun, and K.W. Kwon, “A low power CMOS compatible embedded EEPROM for passive RFID tag”, Microelectronics Journal, 41(10):662–668, 2010.

[15] A. Juels and S. A. Weis, “Authenticating pervasive devices with human protocols”, In V. Shoup, editor, Advances in Cryptology — CRYPTO 2005, volume 3126 of Lecture Notes in Computer Science, pages 293–198. Springer-Verlag, 2005.

[16] H. Eberle, N. Gura, S.C. Shantz, V. Gupta, L. Rarick, “A Public-key Cryptographic Processor for RSA and ECC”, Application-Specific Systems, Architectures and Processors, 2004. Page(s): 98 – 110.

[17] P. Luo; X. Wang; J. Feng; Y. Xu ,“Low-power hardware implementation of ECC processor suitable for low-cost RFID tags”, Solid-State and Integrated-Circuit Technology, ICSICT 2008.

[18] Y. K. Lee, K. Sakiyama, L. Batina, I. Verbauwhede, “Elliptic-Curve-Based Security Processor for RFID“, Computers, IEEE Transactions on Volume: 57, Issue: 11, 2008, Page(s): 1514 – 1527.

[19] S. S. Kumar and C. Paar, “Are standards compliant Elliptic Curve Cryptosystems feasible on RFID?”, In Proceedings of Workshop on RFID Security, page 19, Graz, Austria, July 2006.

[20] L. Batina, N. Mentens, K. Sakiyama, B. Preneel, and I. Verbauwhede, “Public-Key Cryptography on the Top of a Needle”, In Proceedings of the IEEE International Symposium on Circuits and Systems, ISCAS’07, pp. 1831-1834, May 2007.

[21] G. Gaubatz, J. P. Kaps, E. Öztürk, and B. Sunar, “State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks”, In Proceedings of the 3rd IEEE international conference on pervasive computing and communications workshops, March 2005.

[22] J. P. Kaps, “Cryptography for Ultra-Low Power Devices”, Ph.D. Dissertation, ECE Department, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, May 2006.

[23] E. Öztürk, B. Sunar, and E. Savaş, “Low-Power Elliptic Curve Cryptography Using Scaled Modular Arithmetic”, In Proceedings of the sixth International Workshop on Cryptographic Hardware in Embedded Systems (CHES), volume 3156 of Lecture Notes in Computer Science, pp. 92-106. Springer-Verlag, Aug 2004.

[24] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe, “PRESENT: An Ultra-Lightweight Block Cipher”, CHES 2007, LNCS 4727, pp. 450–466.

[25] M. Feldhofer, J. Wolkerstorfer, V. Rijmen, “AES Implementation on a Grain of Sand”, Information Security, IEE Proceedings, Vol. 152, Nr. 1, pp. 13-20, 2005.

[26] I. Verbauwhede, F. Hoornaert, J. Vandewalle, and H. De Man, “Security and Performance Optimization of a New DES Data Encryption Chip”, IEEE Journal of Solid-State Circuits, 1988.

[27] J. Daemen and V. Rijmen, “The Design of Rijndael: AES - The Advanced Encryption Standard”, Springer-Verlag, 2002.

[28] http://en.wikipedia.org/wiki/Advanced_Encryption_Standard.

[29] P. Hamalainen, T. Alho, M. Hannikainen, and T.D. Hamalainen, “Design and implementation of low-area and low-power AES encryption hardware core”, In DSD, pages 577-583. IEEE Computer Society, 2006.

[30] J.P. Kaps and B. Sunar, “Energy comparison of AES and SHA-1 for ubiquitous computing”, EUC Workshops, volume 4097 of LNCS, pp. 372-381. Springer, 2006.

[31] H. Gilbert, T. Peyrin, “Super-Sbox Cryptanalysis: Improved Attacks for AES-like permutations”, Cryptology ePrint Archive, Report 2009/531, http://eprint.iacr.org/2009/531.

[32] [Rij10] V. Rijmen, “Practical-titled attack on AES-128 using chosen-text relations”, Cryptology ePrint Archive, Report 2010/337, 2010, http://eprint.iacr.org/2010/337.

[33] A. Bogdanov, D. Khovratovich, and C. Rechberger, “Biclique cryptanalysis of the full AES”, in Advances in Cryptology, ASIACRYPT, ser. Lecture Notes in Computer Science, D. H. Lee and X. Wang, Eds., vol. 7073, pp. 344–371, Springer-Verlag, 2011.

[34] P. Yalla, J.P. Kaps, “Lightweight Cryptography for FPGAs”, In International Conference on ReConFigurable Computing and FPGAs – ReConFig, IEEE 2009, pages 225–230.

[35] B. Collard and F.X. Standaert, “A statistical saturation attack against the block cipher PRESENT”, In Proceedings of CT-RSA 2009, volume 5473 of LNCS, pages 195-210, Springer, 2009.

[36] M. Wang, “Differential cryptanalysis of Reduced-Round PRESENT”, In Proceedings of AFRICACRYPT 2008, LNCS, pages 40-49. Springer-Verlag, 2008.

[37] K. Ohkuma, “Weak keys of reduced-round present for linear cryptanalysis”, In Selected Areas in Cryptography, SAC, volume 5867 of LNCS, pages 249-265. Springer, 2009.

[38] J.Y.Cho, “Linear cryptanalysis of reduced-round present”, In Topics in Cryptology - CT-RSA, volume 5985 of LNCS, pages 302-317, Springer, 2010.

[39] M.R. Z'Aba, H. Raddum, M. Henricksen, and E. Dawson, “Bit-pattern based integral attack”, In Fast Software Encryption: 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, Revised Selected Papers, pages 363-381, Springer, 2008.

[40] O. Ozen, K. Varici, C. Tezcan, and C. Kocair, “Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT”, In Proceedings of the 14th Australasian Conference on Information Security and Privacy, volume 5594 of LNCS, pages 90-107. Springer, 2009.

[41] J. Nakahara Jr, P. Sepehrdad, B. Zhang, and M. Wang, “Linear (hull) and algebraic cryptanalysis of the block cipher PRESENT”, In Cryptology and Network Security (CANS), volume 5888 of LNCS, pages 58-75, Springer, 2009.

[42] M. Kumar, P. Yadav, M. Kumari, “Flaws in Differential Cryptanalysis of Reduced Round PRESENT”, Cryptology ePrint Archive, Report 2010/407, 2010.

[43] C. Shannon, “Communication Theory of Secrecy Systems”, Bell System Technical Journal, pages 656-715, 1949.

[44] X. Zhao, T. Wang, S. Guo, “Improved Side Channel Cube Attacks on PRESENT”, Cryptology ePrint Archive, Report 2011/165 (2011), http://eprint.iacr.org/2011/165.

[45] D. Kahn, “The Codebreakers”, Macmillan 1996, pp. 397–398, ISBN 0-684-83130-9.

[46] J.A. Reeds, N.J.A. Sloane, “Shift-Register Synthesis (Modulo n)”, SIAM Journal on Computing, 14 (3), pp. 505–513, 1985.

[47] C.D Canniere, “Trivium: A stream cipher construction inspired by block cipher design principles”, Information Security Conference, pages 171–186, ISC 2006.

[48] T. Good, M. Benaissa, “Hardware Results for Selected Stream Cipher Candidates”, State of the Art of Stream Ciphers (SASC), Workshop Record, February 2007.

[49] I. Dinur, A. Shamir, “Cube Attacks on Tweakable Black Box Polynomials. Cryptology ePrint Archive”, ePrint 2008, http://eprint.iacr.org/2008/385.

[50] M. Vielhaber, “Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack”, IACR Cryptology eprint Archive, 2007.

[51] H. Raddum, “Cryptanalytic results on Trivium”, eSTREAM submitted papers, http://www.ecrypt.eu.org/stream/papersdir/2006/039.ps.

[52] A. Maximov and A. Biryukov, “Two Trivial Attacks on Trivium”, Cryptology ePrint, http://eprint.iacr.org/2007/021.

[53] M. Feldhofer, “Comparison of Low-Power Implementations of Trivium and Grain”, eSTREAM, ECRYPT Stream Cipher Project, 2007.

[54] M. Hell, T. Johansson, W. Meier, “Grain - A Stream Cipher for Constrained Environments”, International Journal of Wireless and Mobile Computing, Special Issue on Security of Computer Network and Mobile Systems, 2006.

[55] M. Agren, M. Hell, T. Johansson, W. Meier, “A New Version of Grain-128 with Authentication”, Symmetric Key Encryption Workshop, SKEW 2011.

[56] M. Agren, M. Hell, T. Johansson, “On Hardware-Oriented Message Authentication with Applications towards RFID”, Lightweight Security & Privacy (LightSec), 2011.

[57] O. Kucuk, “Slide Resynchronization Attack on the Initialization of Grain 1.0”, eSTREAM, ECRYPT Stream Cipher Project, 2006.

[58] H. Zhang, X. Wang, “Cryptanalysis of Stream Cipher Grain Family”, IACR Cryptology ePrint Archive, 2009.

[59] M. Lehmann, W. Meier, “Conditional Differential Cryptanalysis of Grain-128a”, 11th International Conference, CANS 2012, Volume 7712, pp 1-11, 2012.

[60] D. Engels, X. Fan, G. Gong, H. Hu and E. Smith, “Hummingbird: ultra-lightweight cryptography for resource-constrained devices”, Financial Cryptography and Data Security, FC'10, LNCS 6054, pp. 3-18, 2010.

[61] M.-J. O. SAARINEN. “Cryptanalysis of Hummingbird-1.” FSE 2011, LNCS 6733, pp. 328–341, Springer 2011.

[62] D. Engels, M.J.O. Saarinen and E. Smith, “The Hummingbird-2 lightweight authenticated encryption algorithm”, In Proceedings of Workshop on RFID Security, RFIDSec, pp. 1-14, 2011.

[63] Q. Chai, G. Gong, “A Cryptanalysis of HummingBird-2: The Differential Sequence Analysis”, IACR Cryptology ePrint Archive 2012.

[64] K. Zhang, L. Ding, J. Gua, “Cryptanalysis of Hummingbird-2”, IACR Cryptology ePrint Archive 2012.

[65] D. Hankerson, A. Menezes, S. Vanstone, “Guide to Elliptic curve cryptography”, book, Springer-Verlag 2004.

[66] J. Black and P. Rogaway, “ CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions”, In Proceedings of Advances in Cryptology-CRYPTO, Lecture Notes in Computer Science, Springer-Verlag vol. 1880, pp. 197-215, 2000.

[67] K. Yuksel, J. Kaps, and B. Sunar, “Universal Hash Functions for Emerging Ultra-Low-Power Networks”, In Proceedings of CNDS, 2004.

[68] L.H. Nguyen, A.W. Roscoe, “New combinatorial bounds for universal hash functions”, IACR Cryptology ePrint Archive, 2009.

[69] R. Anderson, “Security Engineering: A Guide to Building Dependable Distributed Systems”, the first edition, 2001.

[70] E. Biham, “New types of cryptanalytic attacks using related keys”, Journal of Cryptology 7.4, pp. 229-246, 1994.

[71] M. Matsui, A. Yamagishi, “A new method for known plaintext attack of FEAL cipher”, Advances in Cryptology - EUROCRYPT 1992.

[72] N. Courtois, W. Meier, “Algebraic Attacks on Stream Ciphers with Linear Feedback”, EUROCRYPT, pp. 345-359, 2003.

[73] I. Dinur, A. Shamir, “Cube Attacks on Tweakable Black Box Polynomials”, EUROCRYPT, pp. 278-299, 2009.

[74] P. Kocher, “Timing attacks on implementations of Diffie-Hellmann, RSA, DSS and other systems”, CRYPTO’96, LNCS 1109, pp. 104-113, 1996.

[75] P. Kocher, J. Jaffe, B. Jun, “Differential power analysis”, CRYPTO’99, LNCS 1666, pp. 388-397, 1999.

[76] D. Boneh, R.A. DeMillo, R.J. Lipton, “On the importance of checking cryptographic protocols for faults”, EUROCRYPT '97, LNCS 1233, pp.37-51, 1997.

[77] M.G. Kuhn, R.J. Anderson, “Soft tempest: hidden data transmission using electromagnetic emanations”, Information Hiding, LNCS 1525, pp. 124-142, 1998.

[78] Y. Zhou, D. Feng, “Side-Channel Attacks: Ten Years after Its Publication and the Impacts on Cryptographic Module Security Testing”, Cryptology ePrint Archive, 2005.

[79] T.S. Messerges, E.A. Dabbish, R.H. Sloan, “Examining smart-card security under the threat of power analysis attacks”, IEEE Trans. Computers, 51(5), pp.541-552, 2002.

[80] A. Bogdanov, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, “Hash Functions and RFID Tags: Mind the Gap”, Cryptographic Hardware and Embedded Systems – CHES, 2008.

[81] S. Zhilyaev, “EVALUATING A NEW MAC FOR CURRENT AND NEXT GENERATION RFID”, Master thesis, 2010.

[82] A. Satoh, T. Inoue, “ASIC-Hardware-Focused Comparison for Hash Functions MD5, RIPEMD-160, and SHS”, International Conference on Information Technology: Coding and Computing, ITCC, 2005.

Zahra, Jeddi. Bachelor of Science, Iran University of Science and Technology, Spring 2000; Master of Science, AmirKabir University of Technology, Spring 2006; Doctor of Philosophy, University of Louisiana at Lafayette, Summer 2014
Major: Computer Engineering
Title of Dissertation: A Lightweight Authenticated Symmetric Encryption Cipher for RFID Systems
Dissertation Director: Dr. Magdy Bayoumi
Pages in Dissertation: 178; Words in Abstract: 186

ABSTRACT

Radio Frequency Identification, RFID, is a type of automatic identification system which has gained popularity in recent years for being fast and reliable in keeping track of individual objects. Due to limited available resources in RFID tags, providing privacy and security for RFID systems is one of the important challenges nowadays. In this dissertation, a lightweight symmetric encryption algorithm called RBS, Redundant Bit Security, is presented which is suitable for resource constrained applications like RFID systems. Confidentiality of the plaintext in this algorithm is achieved through inserting some redundant bits inside the plaintext bits where the location of redundant bits inside the ciphertext is the secret key shared between sender and receiver. Besides confidentiality, these redundant bits are calculated in such a way that they provide authentication and integrity as well. The security of the algorithm is analyzed against some well-known attacks such as known plaintext, known ciphertext, chosen plaintext, and differential attacks. Experimental and simulation results confirm that RBS implementation requires less power and area overhead compared to other known symmetric algorithms proposed for RFID systems, especially when the authentication is essential like in harsh environments.

BIOGRAPHICAL SKETCH

Zahra Jeddi received her BS degree in Electrical Engineering from Iran University of Science and Technology in 2001 and her MS degree in Computer Engineering from Amirkabir University of Technology in 2006.

Zahra Jeddi joined the Center for Advanced Computer Studies (CACS) at the University of Louisiana at Lafayette in Spring 2009. She contributed in several research projects at CACS such as Body area network, RFID security, and Crypto architecture design. She completed the requirements for this Doctor of Philosophy in Computer Engineering in Summer 2014. Her research interests include hardware security, low power design, and computer architecture.