TAMPERE UNIVERSITY OF TECHNOLOGY

MONA AGHABABAEETAFRESHI

A SECURITY ARCHITECTURE FOR A WIRELESS MEMORY

Master's thesis

Examiner: Prof. Mikko Valkama
Examiners and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 5 June 2013.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
AGHABABAEETAFRESHI, MONA: A Security Architecture for a Wireless Memory
Master of Science Thesis, 67 pages, 4 Appendix pages
October 2013
Major: Digital and Computer Electronics
Examiner: Prof. Mikko Valkama
Keywords: Security architecture, Access control, Authentication, Integrity, Encryption, Implementation on FPGA

Wireless memories are the new trend in memory technology and a result of the latest advances in wireless and data transfer technologies. Allowing transfer of large amounts of data between a host device (e.g., a computer, a mobile phone) and a battery-free wireless memory is essentially the goal of these devices. The advent of this class of memories has opened up the door to a wide range of applications for storing and sharing contents in a wireless manner. Most of the applications of a wireless memory system require a secure transfer of the data between the two sides. In this thesis, means to provide the required security for the wireless memory system is studied, implemented and demonstrated.

This thesis first studies some of the common security threats and the corresponding mechanisms to protect the communication of sensitive data from these threats. Additionally, it analyses some of the threats that are most probable in case of the communication between a wireless memory tag and a host device. Then, the security architecture implemented on the wireless memory's tag side to secure the tag's life-cycle is reviewed. This architecture is implemented based on the limited processing power available in the memory which is due to the fact that the memory tag is wirelessly powered by the host. It is also assumed that more complex mechanisms should be employed in the host side of the system.

The introduced security architecture was implemented using a Cyclone II FPGA board and the employed mechanisms were tested using a Linux machine as the host device. The implemented mechanisms guarantee confidentiality and integrity of the wireless channel between the two side of the communication as well as authentication, access control and secure life-cycle management of the wireless memory.

The number of clock cycles that different security operations need to be performed and the size of the security software were measured using the prototype hardware and synthesis tools confirm the feasibility of the implementation on the actual memory tag. In the future, when more processing capabilities are available on the memory tag, the wireless memory features may be expanded.

PREFACE

This thesis is made as a completion of the Master of Science (MSc) degree in the Department of Electronics and Communications Engineering at Tampere University of Technology. This project was done in the summer and fall of 2012 at Nokia Research Center.

I would like to express my deep appreciation to my supervisor at Tampere University of Technology, Prof. Mikko Valkama for his help and constructive comments. I would also like to extend my gratitude to my supervisors in Nokia Research Center, Dr. Jan-Erik Ekberg and Prof. N. Asokan for their invaluable guidance. Also, many thanks to Ilari Teikari and Joni Jantunen from Nokia Research Center for their help.

My deepest appreciation goes to my family. I owe everything I have achieved to them and without their support, none of this would have been possible.

I would also like to thank my friends in Tampere especially Orod Raeesi whose constant support during my whole Master studies and specifically during the completion of this thesis has been beyond helpful.

Tampere, 14 August 2013

MONA AGHABABAEETAFRESHI

Contents

List of Figures

List of Tables

TERMS AND DEFINITIONS

Security attack Any action that compromises the security of information owned by an entity is called a security attack[35].

Security mechanism A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack[35].

Plaintext Plaintext is the original intelligible input message or data that is fed into the encryption algorithm[35].

Ciphertext Ciphertext is the scrambled message generated as the output of encryption. Ciphertext depends on the plaintext and the encryption key. The ciphertext is a random stream of data and, as it stands, is unintelligible.[35]

Bob and Alice Generally, Alice and Bob are the parties communicating using a non-secure channel. These names were used by Ron Rivest in the 1978 Communications of the ACM article presenting the RSA cryptosystem, and in A Method for Obtaining Digital Signatures and Public-Key Cryptosystems published April 4, 1977, revised September 1, 1977 as technical Memo LCS/TM82.[15]

Mallory Mallory is the malicious attacker in the communication.

One-way function One-way function is defined as a function in which given an argument value x, it is easy to compute the function value $f(x)$, whereas it is intractable to compute x from $f(x)$[33].

Trapdoor one-way function Trapdoor one-way function is a one-way function $f : X \to Y$ with the additional property that given some extra information (called the trapdoor information) it becomes feasible to find for any given $y \in Im(f)$, an $x \in X$ such that $f(x) = y$[24].

A Feistel cipher A feistel cipher is an iterated cipher mapping a $2t$-bit plain text (L_0, R_0), for t-bit blocks L_0 and R_0, to a ciphertext (R_r, L_r), through

an r-round process where $r \geq 1$. For $1 \leq i \leq r$, round i maps $(L_{i-1}, R_{i-1}) \rightarrow (L_i, R_i)$ as follows:

$$Li = R_{i-1}, R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

where each sub-key K_i is derived from the cipher key K.[24]

ICv49 UMETAG ICv4 is one of the integrated circuits designed in the first phase of UMETAG project which integrates the memory interface (SPI) and a base band digital block. ICv49 is the next generation of the UMETAG circuits. ICv49 evaluates features such as NFC remote powering and reader, and device-to-device functionalities.

SPI SPI is a general purpose synchronous serial interface created by Motorola. Using an SPI interface, transmit and receive data can be serially shifted in or out simultaneously. SPI interface can be used for communications with another serial peripheral device or a micro-controller with an SPI interface.[7]

UART The UART performs serial-to-parallel conversions on data received from a peripheral device and parallel-to-serial conversion on data received from the CPU[17].

1. INTRODUCTION

1.1 Wireless Memory Trend and Fundamentals

The very latest advances in data transfer technology has led to the advent of wireless memories. Wireless memory technology introduces multiple alternatives for storing and sharing contents in a wireless manner. Integration of wireless technology with the major group of the products nowadays brings out a variety of applications for wireless memories. Many use cases can be based on the ability of wirelessly transferring contents from/to a mobile device especially at high speeds and capacities[40]. Promising fast and large data transfers using battery-free wireless memories proposed by this generation of storage devices offers a wide variety of potential applications[8].

A wireless memory system consists of a high capacity non-volatile storage device and a memory subsystem embedded in a mobile device. The former is addressed as the "tag" and is often a battery-free device. The latter, on the other hand, is called a host and is a battery powered system embedded in another device. A Tag device is initially composed of an RF unit and a memory while a host device is comprised of an application processor, an RF unit and a memory. The wireless memory system is meant to be used basically for transferring data between the host and the tag within short proximity. The basic elements of a wireless memory system are as follows:

Radio The RF unit employs ultra-wide band technology which enables achieving high data rates but in near proximity according to the power restrictions made by the Federal Communications Commission. The ultra-wide band technology operating at 7.9 GHz center frequency enables data rates up to 108 Mb/s. The tag device is designed to be wirelessly powered by the host device which is provided by implementing a super-regenerative architecture in the tag device's RF unit.

Memory Various non-volatile memory technologies may be adopted in the wireless memory system on the condition that they satisfy the required high data rates, high storage capacity, low power consumption and long life span. Some candidates studied are NAND/NOR flashes and phase change memory.

Processing A processing unit is required at both sides to handle, control and respond properly to the commands which is, as a result, responsible for running the software required for the security too. On the tag side (the target of this thesis), a finite state machine implemented in the tag's circuit holds the responsibilities of communicating with the host based on the commands received.

1.2 Security

Information security is defined as the protection afforded to a system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of the information system resources[14]. This definition highlights three key aspects of security: first, confidentiality which includes data confidentiality and privacy, second, integrity which consists of both data and system integrity and last availability. This triad embodies the vital objectives of security for a system[35]. However, some feel that some additional concepts are needed to ensure a completely secure system. Two other commonly mentioned are authenticity and accountability[35].

Confidentiality basically means preserving authorized access and disclosure of information[16]. Integrity aims for protecting information from abnormal modifications or total destructions. Availability ensures proper access for the users of the system. Authenticity is gaining trust and the knowledge of genuineness of the different parties involved in the communication. Accountability is the ability of tracing an entity's actions to that entity uniquely[35].

Some threats that may endanger preserving the aforementioned concepts can be listed as:

- Eavesdropping
- Impersonation
- Man-In-The-Middle
- Skimming
- Jamming
- Denial of service

To prevent or lower the risks of such attacks, some mechanisms have been introduced. These mechanism are mainly categorized into two groups of symmetric key and public key schemes as well some other hybrid schemes based on the two first categories.

- Symmetric key encryption: Symmetric key encryption consists of five ingredients: plaintext, encryption algorithm, secret key, ciphertext and decryption algorithm.[35] The basic idea in symmetric key encryption is that same key is used for both encryption and decryption. A symmetric key algorithm transforms the plaintext into a ciphertext using the secret key and the encryption algorithm and on the other side using the same key and a decryption algorithm, the plaintext can be retrieved from the ciphertext.[35]

- Public key encryption: In public key encryption (also known as asymmetric encryption) a party processes a pair of keys (as opposed to one secret key in symmetric encryption) : a public key and an associated secret key [2]. A party's public key can be publicly known. Asymmetric encryption transforms plaintext into ciphertext using one of the two keys and an encryption algorithm. On the other side, the plaintext can be retrieved using the paired key and an encryption algorithm[35]. Public key encryption can be used for confidentiality, authentication, or both.

1.3 This Thesis

One of the very essential aspects that most of the applications of the wireless memory systems rely on is the degree of access control and security provided when transferring information. Moreover, the increasing number of cyber security threats along with the wide range of applications for the wireless memory system strengthens the urge for equipping these devices with security mechanisms

Wireless memory security means wireless memory access control, authentication and secure life-cycle management, as well as the confidentiality and integrity of the wireless channel between the host and the wireless memory.

The goal of this thesis is to enable the wireless memory system to maintain confidentiality, integrity and authentication when exchanging information between tag and the host. In this thesis, the possible threats that may endanger the security of the wireless memory tag are discussed and means to protect the tag from these potential attacks are introduced.

This thesis highlight some ways to add the required security support in wireless memory tag, based on conditional access to parts of the wireless memory. This means that "read"s and "write"s to some memory parts would be only possible in a certain manner or would be restricted by specific PINs. The access control mechanisms that will be described basically prevent "reads" by unintended parties and accidental, or malicious "overwrite"s of user's data stored on the wireless memory.

The aforementioned access control is achieved by organizing the available memory capacity to different segments, each segment configured to respond to the commands

from the reader in a specific manner. As a result, there will be three main areas in the memory: an area to hold the data for different segment's configuration, some access controlled segments and a public area to be used for insensitive information transfer. Precise memory layout and details on the behaviour of access controlled segments are discussed later in section 5.1.

When designing the security architecture for the memory tag, it should be taken to account that the memory can only take advantage of a limited processing capability. Additionally, as the tag is powered by the host device, it has to be considered that only consuming extremely low power is acceptable. Consequently, it is assumed that the master host (the "reader") implements standard cryptographic functions such as the AES (Advanced Encryption Standard), while the wireless memory supports only the security functions requiring low processing capability.

1.3.1 The Prototype

During the development of the wireless memory system, all communications between the host and the tag are wired. The memory is connected to a board equipped with a Cyclone II FPGA, regulators, connectors and converters. This board along with the memory is basically operating as the tag side in the system while a Linux machine is responsible for the operations done by the host. On the host side, the PC is connected to the board using a USB cable which is converted to the UART afterwards. Then it is connected to the UART port of the Cyclone II FPGA board where the Nios II processor, as mentioned earlier, will be responsible for the processing tasks. Using the SPI bus, the Cyclone II FPGA is connected to the non-volatile memory. In this mode of operation, the tag system (including the FPGA and the Nios II processor) is directly powered by the regulators on the board and thus no super regenerative architecture is needed.

1.3.2 Structure

This thesis is organized as follows:

Chapter 2 discusses some of the well known security threats, how they attempt to gain access to information, the cases in which a certain security attack may be successful and some examples of each type of the threats. This discussion is followed by introducing some mechanisms commonly used to provide information security. These mechanisms are represented with mathematical expressions and details on how they work are described.

Chapter 3 addresses some of the suitable memory technologies for the wireless memory system and their advantages and disadvantages. Furthermore, a comparison on latency, speed and density is done between the addressed technologies. In

addition to memory technologies, the wireless technology employed for the wireless memory system is studied. This section explains how the super-regenerative architecture is employed to enable the memory tag to operate, being powered by the host. It is also clarified how the ultra-wide band technology allows transferring data with high data rates.

Chapter 4 explains the hardware and software architecture of both the tag and the host side. The hardware section addresses the basic elements of the wireless memory system: radio, memory and processing. Following the hardware, the software section depicts an abstract view of the software architecture and how the security software is positioned in the overall software architecture of the tag. It is also clarified how the architecture changes during the development of the project when the connection between the two sides is wired and a PC is used to replicate the actions of the host side.

Chapter 5 first analyses some of the threats introduced in Chapter 2 and looks at them in the context of the communications between the two sides of a wireless memory system. It also studies how the information exchange between the tag and the host can be endangered in different types of security attacks. Secondly, it is explained how the security and access control techniques are implemented in the tag side of the system. Furthermore, the memory tag's organization, layout and operations are clarified. This section includes detailed demonstration of the different elements of the security architecture.

Chapter 6 studies the security mechanisms implemented in this project. It defines how these mechanisms are integrated to the memory's architecture and how they can prevent the possible threats that endanger the memory's security. Additionally, some measurement results obtained from the software tools on the timing, code size and area of the implemented architecture are presented. Furthermore, a few test cases for some of the access control and security operations are provided.

Finally, in the conclusion chapter, the suggested mechanisms, their importance and strengths in protecting the memory from attacks are summarized. It is summed up how with the available resources in an stand-alone powerless tag device, adequate security is obtained. Some recommendations for future research are also provided at the end.

2. SECURITY THREATS AND MECHANISMS

This chapter addresses some common security threats and how they can access sensitive data and endanger the security of a system. Furthermore, some mechanisms that can help lower the risks of such threats are introduced.

2.1 Security Threats

Wireless devices are prone to many security threats as a result of the nature of the wireless medium. Over the air communications can be overheard by any receiver in the transmission range of the sender.

2.1.1 Eavesdropping

Eavesdropping is referred to the act of listening to the communication between (the tag and a genuine reader) by an adversary. Eavesdropping is usually done with the goal of achieving data that can be used by the adversary to pretend to be an authorized reader[39]. So the information will not be changed in favour of the eavesdropper but it remains intact, however, it can be used for impersonation. Depending on the network topology and the communication standard eavesdropping may be simple or difficult. While eavesdropping on a wired communication is relatively difficult(due to lack of physical access to the medium), eavesdropping on a wireless communication can be greatly easier. Data captured by eavesdropping can be useful if unencrypted, or encrypted using known encryption method. Data with an unknown encryption is without any value for the eavesdropper.

2.1.2 Impersonation

The adversary's actions aiming to represent himself as another user is referred to as impersonation. If Mallory successfully deceives Alice into believing that she is communicating with Bob while actually she is exchanging the information with Mallory, She is impersonating Bob.Therefore, besides consistency, impersonation undermines a range of security goals such as authentication, authorization, non-repudiation, accountability, and possibly data integrity and confidentiality. Impersonation often requires to forge authentication data or to send messages with forged source addresses. A special case of impersonation attack is Man-In-The-Middle attack

presented below.[13]

2.1.3 Man-In-The-Middle

"Man In The Middle" security threat occurs when a two party communication is intercepted by a third party without letting the two main parties know. The man in the middle will use the information and alter it for his own purposes. The man in the middle should be placed in the network path between the two parties (as shown in Figure 2.1) so that it can delay,modify or drop packets[3].

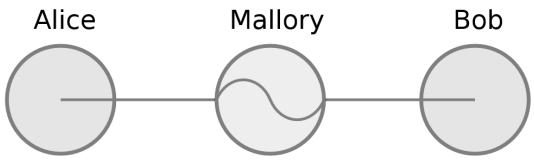

Figure 2.1. *Man-In-The-Middle attack*

Imagine Alice and Bob as the two sides of the communication and Mallory as the Man In The Middle. Let's say Alice needs Bob to send his bank account information to her. Alice sends her public key in a message to Bob so that Bob can encrypt his bank account information with her key. Mallory reads the message and alters it into his own public key and remembers Alice's. Then Bob encrypts his bank information with Mallory's public key, thinking that it is Alice's and then sends it to Alice. Mallory receives the message decrypt it with his own key and therefore has Bob's bank information. Mallory encrypts an irrelevant number with Alice's key and sends it to Alice. Alice decrypts it with her own key assuming that it she will get Bob's info. So the Man in the middle attack compromises the public key cryptography.

2.1.4 Skimming

Authorization is the operation used to prove that the reader touching the wireless device is genuine and the upcoming communications with the reader can be trusted in that sense. The security threat caused when an adversary starts a communication with the tag without any authorization is known as "skimming". Skimming attacks are mostly known by credit card skimmings. credit card skimming can be done using devices that read all the digital content on a magnetic stripe of a credit or debit card.[12]

2.1.5 Jamming

Radio communications are subject to "jamming" which basically means transmitting unwanted radio signals on the communication channel "intentionally" to decrease the signal to noise ratio(SNR) of the received signal. Received signal with SNR value less than 1 (which basically means having bigger noise value in comparison with the signal value) indicates a successful jamming attack[31]. Spot jamming, sweep jamming, barrage jamming and deceptive jamming are among the existing jamming methods[27].

2.1.6 Denial of Service

Denial of service attacks are based on wasting all the available resources from the victim which leaves the victim without means to provide services for the eligible hosts. This type of attacks usually aim for hosts providing a type of service and thus by disrupting the services, the reliability of the victims are targeted. Denial-of-service attacks come in a variety of forms and aim at a variety of services. There are three basic types of attack:[34]

- consumption of scarce, limited, or non-renewable resources
- destruction or alteration of configuration information
- physical destruction or alteration of network components

2.2 Security Mechanisms

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication. Cryptography is not the only means of providing information security, but rather one set of techniques. Cryptography is about the prevention and detection of cheating and other malicious activities.[24]

Cryptographic schemes are basically divided into two main groups: symmetric key, public key. Additionally, there are some hybrid schemes commonly applied today using both symmetric key and public key algorithms relying on the advantages and avoiding disadvantages of both schemes.

- ***Symmetric key encryption***

  A symmetric key encryption, also called single-key, one-key, private key, and conventional encryption, is defined as follows.[24]

  Consider an encryption scheme consisting of the sets of encryption and decryption transformations

  $$E = \{E_e : e \in K\}$$

and

$$D = \{D_d : d \in K\},$$

respectively, (where K is the key space). The encryption scheme is said to be symmetric-key if for each associated encryption/ decryption key pair (e, d), it is computationally "easy" to determine d knowing only e, and to determine e from d. d and e being equal in most practical matter makes the term "symmetric" appropriate.

Symmetric key encryption is usually described as a safe box with a strong key. Anyone with the key can send and receive messages safely using the box.

Consider a communication between Bob and Alice where an unknown third party can hear the communication channel between them and the data needs to be hidden from him. Alice encrypts her message using Bob's key so that Bob can decrypt the message using the same key and access the contents of the message. The eavesdropper cannot understand the message and the data is safe. This case can be useful only assuming that there is a secure channel for Bob to send his key to Alice without the eavesdropper hearing it.

Disadvantages of the symmetric key encryption :

- Need for a secure channel for the key transmission.
- Need for having various keys in a network where each communicating pair needs a different key.

- ***Public Key encryption***

  A public key encryption is defined as follows.[24]

  Let

  $$E = \{E_e : e \in K\}$$

  be a set of encryption transformations, and let

  $$D = \{D_d : d \in K\}$$

  be the set of corresponding decryption transformations, where K is the key space. Consider any pair of associated encryption/decryption transformations (Ee,Dd) and suppose that each pair has the property that knowing Ee it is computationally infeasible, given a random cipher-text $c \in C$, to find the message $m \in M$ such that $Ee(m) = c$. This property implies that given e, it is infeasible to determine the corresponding decryption key d. (Of course e and d are simply means to describe the encryption and decryption functions, respectively.) Ee is being viewed here as a trapdoor one-way function with d

being the trapdoor information necessary to compute the inverse function and hence allow decryption. This is unlike symmetric-key ciphers where e and d are essentially the same. The encryption key e is called *the public key* and can be transmitted over an unsecure channel, while the decryption key d is called *the private* and is kept safe with the receiver.

In the safe box example the public key cryptography can be described as a safe box that can only be opened by the receiver. If the receiver let the lock open, anyone can deposit in the box, however, only the receiver can understand the messages. Even the sender cannot retrieve the message, in case he erases it after depositing.

Mechanism using the public key encryption:

1. Establishment protocols (e.g., Diffi-Hellman key exchange)
2. Digital signature algorithms (e.g., RSA, DSA or ECDSA)
3. Encryption

Disadvantages of the public key encryption :

Computations used for public key transmissions are much more demanding than the symmetric key encryption algorithms.

2.2.1 Block Cipher

Block ciphers divide the plaintext message into blocks of equal length over an alphabet and does the encryption one block at a time. [24] This means that the encryption of any plaintext bit in a given block depends on every other plaintext bit in the same block.

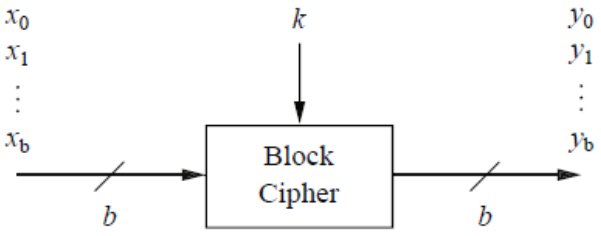

Figure 2.2. *Block Cipher*

As it can be seen in Figure 2.2, the encryption function $e_k()$ is used for all blocks $x_1, x_2, ..., x_b$ to obtain:

$$y_1, y_2, ..., y_b = e_k(x_1), e_k(x_2), ..., e_k(x_b).$$

The encryption function e_k needs to be a one to one function to enable unique decryption. As a fundamental building block, the block ciphers' versatility allows construction of pseudo-random number generators, stream ciphers, MACs, and hash functions.[24] They provide confidentiality and are used as a basic and prominent element in many cryptographic applications.

Majority of block ciphers use 128 bit (16 Byte) or 64 bit (8 Byte) blocks. An example of the former is Advanced Encryption Standard (AES) and Data Encryption Standard (DES) for the latter.

The data encryption standard(DES)

Although nowadays, DES is not considered secure againts a determined attacker because of its small key space[29] and is on the way out, it is still used in legacy applications. The effective secret key length for DES is 56 bits and it uses block-length of 64 bits and it consists of 16 rounds of what is called a *"Feistel network"*.[2] Looking at Figure 2.3, it can be understood that the DES algorithm consists of three major phases. First of which is the *Initial permutation* (IP) which rearranges bits to produce the *permuted input*.[35] Then the permuted input goes through 16 rounds of the same function which involves both permutation and substitution functions. The result from these 16 rounds is a 64 bit output which is a function of both the secret key and the plaintext input. Then a 32 bit swap is done to generate the *pre-output*. In the third and last phase, the pre-output goes through the inverse permutation (IP^{-1}) that was used in the first phase to produce the 64 bit ciphertext output.

The right-hand part of Figure 2.3 shows how the 56 bit key is used. First the key is passed through a permutation function. Then using a left circular shift and a permutation a *sub key* is generated for each of the 16 rounds. The initial permutation (IP) is defined in table 2.1.

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Table 2.1. *Initial permutation* (IP)

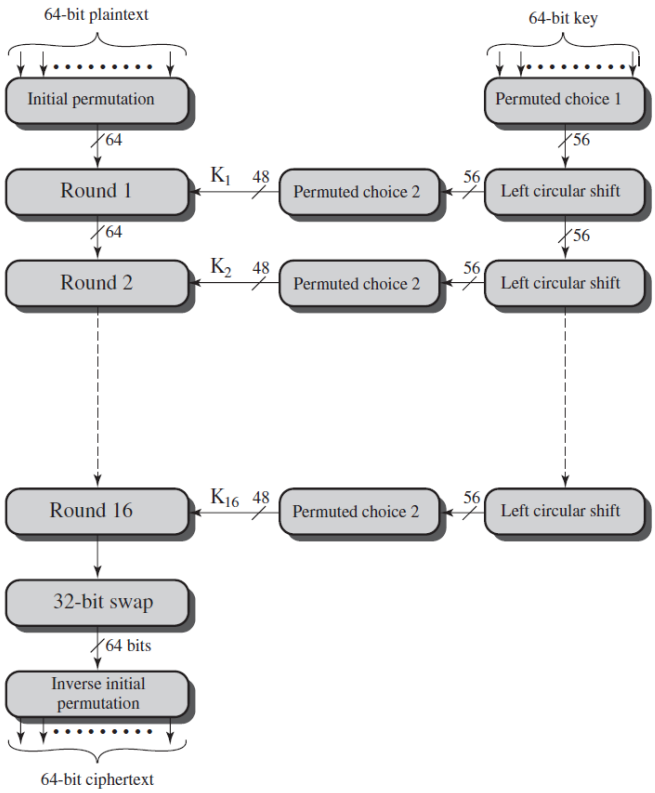

Figure 2.3. *General description of DES encryption algorithm*

The advanced encryption standard(AES)

AES is a block cipher and a successor to DES. Unlike DES, AES uses 128 bit block size and a key size of 128, 192, or 256 bits. Instead of *Feistel network* structure, AES encrypts all 128 bits in one iteration and as a result it has a comparably small number of rounds.[28] Each full round consists of four separate functions: byte substitution, permutation, arithmetic operations over a finite field and XOR with a key.[35] AES contains N rounds, and the number of rounds needed depends on the length of the key. Each round uses the four transformation functions except for the final round which only contains of three transformation functions. There is also a single transformation before the first round. Each 128 bit block of input is considered as $4 * 4$ square matrix of bytes. Each transformation takes in one $4 * 4$ matrix as input and produces a $4 * 4$ output. The details of AES cipher encryption and decryption are illustrated in Figure 2.4.

TEA, XTEA and XXTEA

Tiny encryption algorithm(TEA) is a block cipher operating on 64 bit blocks and using 128-bit key. TEA is fast and simple and has a very small size. Its security weaknesses are having equivalent keys, related-key and slide attacks[30]. Having

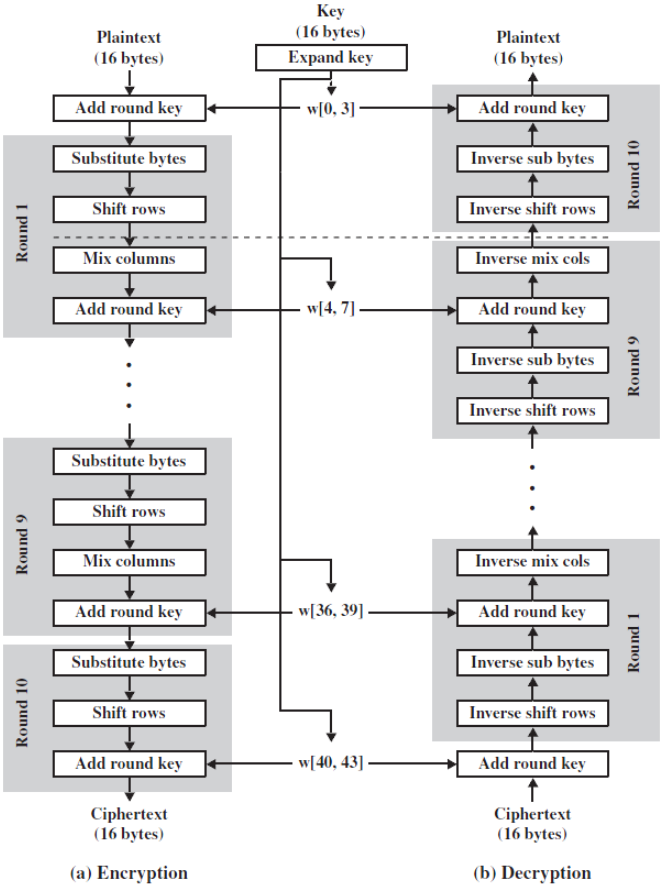

Figure 2.4. *AES encryption and decryption*

equivalent keys reduces the effective key size to 126 bits. XTEA (eXtended TEA) is an extension of TEA which corrects some of the weaknesses of original TEA.[43] XTEA has a more complex key schedule, and XXTEA (corrected block TEA) which is explained in more details.

XXTEA is a block cipher designed by Roger Needham and David Wheeler of the Cambridge Computer Laboratory. Formally speaking, XXTEA is a consistent incomplete source-heavy heterogeneous UFN (unbalanced Feistel network).[1] It operates on a block consisting of at least two 32-bit words, using a 128-bit key. The block can be viewed as a circular array. A single XXTEA full cycle consists of looping through the block words, adding to each word a function of its immediate neighbours, full cycle number and the key; a single XXTEA round for a fixed block length can be concisely described as: [42]

$$v_r \leftarrow v_r + F(v_{r-1}, v_{r-1}, r, k)$$

A full cycle is n rounds, where n is the number of words in the block. The number of full cycles to perform over the block is given as $6 + 52/n$.[42]

One round of XXTEA is shown in Figure 2.5

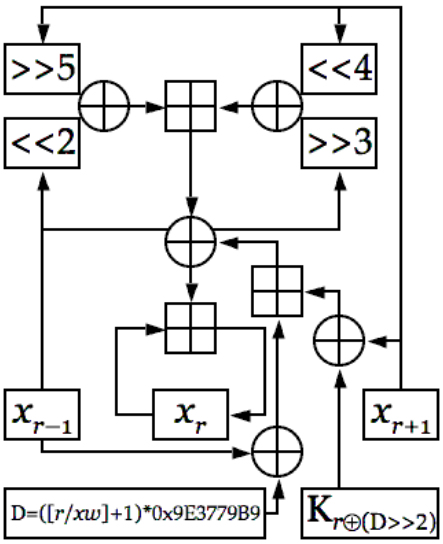

Figure 2.5. *One round of XXTEA*

An attack published in 2010 by E. Yarrkov presents a chosen plaintext attack against full-round XXTEA, requiring 259 queries and negligible work.[42]

2.2.2 Stream Cipher

Using time-varying encryption transformation functions, stream cipher encrypts the plaintext message bit by bit.

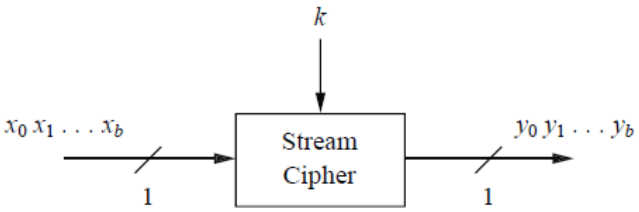

Figure 2.6. *Stream Cipher*

As it can be observed from Figure 2.6, for each single bit $x_1, x_2, ..., x_b$ different encryption function are used to obtain:

$$y_1, y_2, ..., y_b = e_{z_1}(x_1), e_{z_2}(x_2), ..., e_{z_b}(x_b).$$

where $z_1, z_1, ..., z_b$ is the key stream.[28]

This is achieved by adding a bit from a key stream to a plaintext bit. There are synchronous stream ciphers where the key stream depends only on the key, and asynchronous ones where the key stream also depends on the ciphertext. [29]

In comparison with block ciphers, stream ciphers are faster and smaller and thus suitable for applications with little computational resources like in cell phones. A prominent example for a stream cipher is the A5/1 cipher, which is part of the GSM mobile phone standard and is used for voice encryption.

2.2.3 Digital Signature

The idea for a digital signature is basically extracted from the conventional way of signing on a paper. Digital signatures bind the identity of an entity to a particular message or piece of information.[23]. What digital signatures bring to the table in the security field is not privacy or secrecy. Digital signatures provide the integrity of the message and verification of the person who has written the message.

To achieve a better understanding of digital signatures, let's study the subject using the example of the communication between Bob Alice who share a secret key which is used for encryption with a block cipher. Assuming that only Bob and Alice know the key, the communication can be considered reasonably safe in the sense that a third party has not changed the message. However, is the attack always initiated by a third party? Can the two parties always trust each other?Are the symmetric schemes sufficient to secure the two sides of the communication from each other?

Bob sends Alice a message saying that he is going to meet her at 6 o'clock. Bob encrypts this message using their shared secret key and sends it to Alice. However, he forgets to meet her. Being dishonest, Bob claims that he never sent her such message. Without a signature of Bob at the end of message, there is no way to prove his dishonesty. This example is rather simple but in practice there are many cases that there is a need to prove the fact that a specific person has created a certain message. This is where digital signatures come in handy. Bob signs the message x with his private key k_{pr} :

$$y = sign_{k_{pr}}(x)$$

Bob sends (y, x) to Alice. Alice runs the verification function $ver_{k_{pub}}(x, y)$ with Bob's public key.

As it is shown in Figure 2.7, the signature should be both a function of the private key and the message. It is a function of the private key so that only the holder of the key can sign it. To make sure that it changes with every message, it should be a function of the message.

Properties of digital signatures:[28]

- Only Bob can sign his document (with the private key k_{pr}).
- Everyone can verify the signature (with the public key k_{pub}).

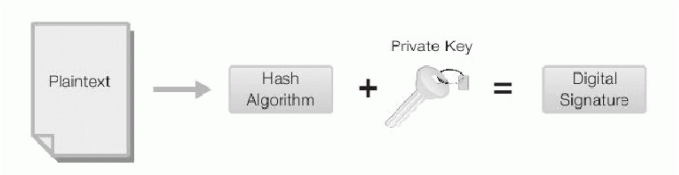

Figure 2.7. *A digital signature*

- Authentication: Alice is sure that Bob signed the message.
- Integrity: Message x cannot be altered since that would be detected through verification.
- Non-repudiation: The receiver of the message can prove that the sender had actually send the message.

Lamport Signature

The first hash based scheme is Lamport's one time signature scheme. To sign message of length k the system is set up as follows:[38]

Let H be a one way function

$$H : \{0,1\}^k \to \{0,1\}^k.$$

The signer chooses $2k$ random strings $x_1[0], x_1[1], ..., x_k[0], x_k[1]$ and stores them as his secret key X. Then he computes

$$y_i[b] = H(x_i[b])$$

for $b \in \{0,1\}$ and $1 \leq i \leq k$. The public key is the vector

$$Y = (y_1[0], y_1[1], ..., y_k[0], y_k[1])$$

of $2k$ strings of length k each. The signature $S(X,m)$ of a message m is $x_1[m_1], ..., x_k[m_k]$ where $m = (m_1, ..., m_k)$. The verifier checks whether $H(x_i[m_i]) = y_i[m_i]$ for $1 \leq i \leq k$.

The scheme is called one-time signature because a public key can be used only for one signature: if a second message is signed that differs from the first in at least two hash bits, then an attacker can generate further valid signatures. Assume that the first message satisfies $m_1 = (0,1,1,...)$ and the second message satisfies $m_2 = (1,1,0,...)$. Then $x_1[0], x_1[1], x_2[1], x_3[0], x_3[1], ...$ are revealed, allowing anybody to sign also messages $(1,1,1,...)$ or $(0,1,0,..)$, where the values in the dots need to match one of the previously signed messages.

2.2.4 Hash Functions

When it comes to very long messages, digital signatures do not seem to be as efficient. For the aim of performance and security, it is preferred to have signatures of the same length for messages with different lengths. The solution lies in the hands of *hash functions*. Hash functions use the plaintext message as input and produce an output referred to as a message digest, fingerprint of the message, hash code, hash-result, hash-value, or simply hash.[24] The output is a short, fixed length string.

Requirements for hash function h***:***[28]

- Hash function $h(x)$ should be relatively easy to compute
- It should be a *one-way function*. For almost any given output z, it is impossible to compute input x such that $h(x) = z$.
- Given x and consequently $h(x)$, it must be impossible to find $x\prime$ such that $h(x) = h(x\prime)$.
- For the hash function h, it should be impossible to find any two pairs $(x, x\prime)$ such that $h(x) = h(x\prime)$.

Unlike other cryptographic algorithms introduced, hash functions usually do not use keys. There are definitions of hash functions available where a secret key is fed to the hash function as well as the input message itself. However, this is not the common case. Based on these definitions, hash functions can be split into two groups:[24]

- Unkeyed hash functions
- Keyed hash functions

Hash functions are mainly for digital signature schemes and message authentication codes. Figure 2.8 shows the basic protocol of hash functions used in digital signatures.[29]

Bob wants to send a signed message to Alice. Bob first calculates the hash value z of message x using the hash function $h(x)$.

$$z = h(x)$$

Then he signs the hash value z using his private key $K_{pr,B}$ and produces s.

$$s = sign_{k_{pr,B}}(z)$$

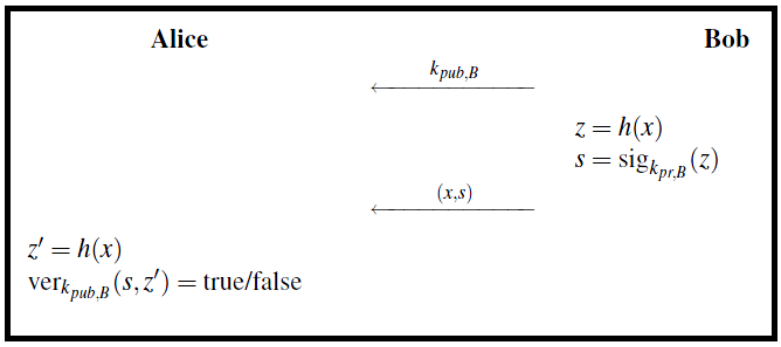

Figure 2.8. *Hash functions and digital signatures*

Note that the hash value z has a fixed length, resulting in signature s to be of a fixed length as well.

On the other side, Alice computes the hash value $z\prime$.

$$z\prime = h(x)$$

She can verify Bob's signature s with Bob's public key $K_{pub,B}$.

$$ver_{K_{pub,B}}(s, z\prime) = true/false$$

2.2.5 Authentication

Authentication is the technique used by one of the communicating parties to verify the identity of the other party. This technique is used to prevent impersonation attacked described in the previous chapter. This is mostly done by checking the correctness and integrity of the message using a secret associated by design with the genuine party.[24] Most commonly used message authentication is a *message authentication code*(MAC), also known as a keyed hash function.[35].

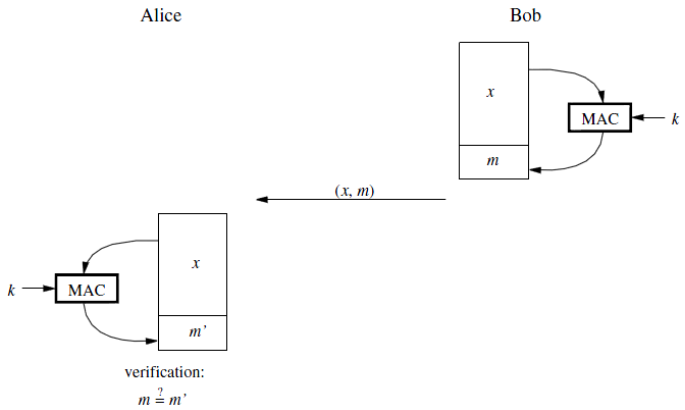

Figure 2.9. *Principle of message authentication codes*

Principle of MACs generation and verification is shown in Figure 2.9.

A MAC is a function of symmetric key k and message x, thus we have:[29]

$$m = MAC_k(x)$$

The produced MAC m can be transmitted with the protected message x to the other party. The integrity of the message can be checked by producing the MAC value of the message and comparing it with the transmitted one. An adversary wanting to alter the message cannot produce the right MAC value without knowing the secret key. The only parties in possession of the secret key can produce the appropriate MACs.

First Bob computes $m = MAC_k(x)$ and sends the message x together with the code m to Alice. Then Alice calculates $y\prime = MAC_k(x)$, compares y with $y\prime$ and thus verifies the integrity of the message. Note that Alice calculates the MAC exactly the same way Bob does.

As it can be observed, MACs have some common properties with digital signatures. They both provide message authentication and integrity, however, unlike digital signatures, message authentication codes are symmetric key schemes both for creating authentication code and for verifying it; and do not provide non-repudiation.[29] MACs are much faster than digital signatures and are based on hash functions or symmetric ciphers.

MACs from block hash functions: HMAC

The basic idea for the HMACs is to hash a secret key k with message x and consider the hash output as the authenticating tag for the message:

$$m = HMAC_k(x) = h(k||x)$$

and:

$$HMAC_k(x) = h[(k^+ \oplus opad)||h[(k^+ \oplus opad)||x]]$$

where the symbol "||" denotes concatenation, k_+ is k padded with zeros on the left so that the result is b bits in length (where b is the number of bits in a block), ipad is 00110110 repeated $b/8$ times and ipod is 01011010 repeated $b/8$ times.

MACs from block ciphers: CBC-MAC

An alternative method for using hash functions in MAC generation is to construct MACs using block ciphers. The most common approach is to use a block cipher such as AES in cipher block chaining (CBC) mode.

3. MEMORY AND WIRELESS TECHNOLOGY

This chapter shortly reviews some suitable memory technologies to be used in the wireless memory system and compares them in terms of speed, capacity and energy consumption. Then in the next section, the employed wireless technology in the system is thoroughly discussed.

3.1 Memory Technologies

Various memory technologies can be taken into account when selecting an appropriate memory to be used in the wireless memory tag. The ability to maintain information after being powered down (non-volatile memory) is the first criteria to be met by the chosen memory technology. Moreover, another critical matter that has to be considered is the power consumption of the memory, as the target of the tag design is to be powered through wireless power transfer. Other important issues to look at include capacity, speed and life span of the different memory technologies. Two applicable technologies are NAND/NOR flash and PCM.

3.1.1 Flash Memory Technology

Flash memories are a powerful and cost-effective non-volatile memory that was essentially developed from EEPROM (electrically erasable programmable read-only memory). These memories are characterized by the fact that the erasing operation should be performed at the same time on a sector or block of memory in contrast to being done cell by cell. This brings the advantage of having a competitive size for flash memories.[25] Flash memories are categorized into two dominant forms depending on the way the cells are organized: NAND and NOR flash devices.

NOR flash

The internal architecture of the NOR flash is such that the individual memory cells are connected in parallel. As a result, the memory achieves random access and thus short read times. This ability introduces nor flash as an ideal technology for low density, high speed read applications.[10] NOR flash has been typically used for code storage and direct execution in portable electronics devices.[10]

NAND flash

NAND flashes were developed to compensate for the low density and size of the NOR flash memories. IN NAND flashes random access is given up in trade-off with size. The small cell size of NAND chips results in lower cost per bit. NAND flashes achieve fast writes and erases with programming groups of data by utilizing their high density and small cell size. While an erase operation is straight forward in NAND technology, in NOR technology all bytes need to be written with zeros before being erased. Moreover writes to flash devices can only be performed only if the device has been erased before. The two previous properties result in the faster performance of NAND devices in write and erase operations.[37] Regarding the life span of the two types of the flash memories NAND flashes offer up to ten times the life span of NOR devices. NAND flash is an ideal technology for low cost, high density, high speed applications.

3.1.2 Phase-change Memory (PCM)

Phase change memory is an emerging non-volatile technology exploiting the property of chalcogenide glass. PCM is a dense technology and each PCM cell can store more than one bit. PCM offers significantly better read access time in comparison with NAND technology but has slightly lower write speed.[32] In comparison with NAND/NOR flash memories, phase-change memory offers longer life span. PCM technology is less efficient than NAND technology when the memory is accessed as relatively large blocks. NAND Flash array program/read currents for state of the art Single-Level Cell (SLC) devices are typically in the range of 25 to 50 mA, while PCM array program currents for state of the art devices are typically in the range of 50 to 70 mA and array read currents in the range of 30 to 50 mA. [20]

Table 3.1 derived from [32] shows the differences between the three technologies: NOR flash, NAND flash and PCM.

Parameter	DRAM	NAND Flash	NOR flash	PCM
Density	1X	5X	0.25X	2X-4X
Read Latency	60 ns	25 us	300 ns	200-300 ns
Write Speed	1 Gbps	2.4 MB/s	0.5 MB/s	100 MB/s

Table 3.1. *Comparison between memory technologies*

3.2 Wireless Technology

The most important concepts in the wireless technology employed in the wireless memory system are the ultra-wide band radio communication technology and the super regenerative architecture which are reviewed shortly below.

The RF unit in the tag employs a I-UWB 7.9 GHz communication and a simple On-Off-Keying (OOK) modulation. The maximum achievable data rate in this architecture will be 108.48 Mb/s.

3.2.1 Ultra-wideband Technology

UWB technology is loosely defined as any wireless transmission scheme that occupies a bandwidth of more than 25% of the center frequency, or more than 1.5GHz.[11] As it can be easily inferred from its name, UWB uses an extremely wide band of RF spectrum which consequently enables it to achieve much higher data rates than the more traditional technologies.

In 2002, the Federal Communications Commission (FCC) in the United States essentially unleashed huge "new bandwidth" (3.6 - 10.1 GHz) at the noise floor. UWB radios can use frequencies from 3.1 GHZ to 10.6 GHz, a more than 7 GHz wide band[6][41]. Each radio channel can have more than 500MHz of bandwidth in accordance with its center frequency. FCC has made severe transmit power restrictions which enables UWB to make use of such wide frequency band while not interfering with nearby devices using narrower band such as 802.11a/b/g radios. As a result UWB devices can obtain very high data throughput but only over short distances. Using UWB technology allows reuse of the spectrum meaning that a group of devices can communicate on the same channel used by another group of devices e.g. in another room.

3.2.2 Super-regenerative Architecture

Why super-regenerative receiver and ultra-wideband technology

Super-regenerative receivers were most commonly used in narrowband communications over the last decade. The main reason was the extremely low power consumption obtained with the new technologies. While the lack of frequency selectivity was a major drawback in narrowband communications, it is used as an advantage in ultra wideband communications[21]. Additionally super-regeneration relies on an unstable circuit which enables huge RF gain with extremely low power consumption which is a great condition for the power and gain trade-off in ultra wideband communication systems. Above all, super-regenerative architecture is sensitive to time domain energy concentration which is specifically proper for the ultra wideband impulse signals since these signals concentrate all the useful energy in short time duration. In contrast to conventional impulse UWB transceivers there is no need for multipath recovery over the distances below 30 cm. This decreases the requirements set for the UWB transceivers. This is used to minimize complexity and power consumption of the transceivers.[18]

Super-regenerative architecture for impulse UWB receivers

As shown in Figure 3.1, the core of the receiver is a super-regenerative oscillator, an RF oscillator that can be modeled as a frequency selective network or resonant circuit whose output is fed back through a variable gain amplifier[26]. The low

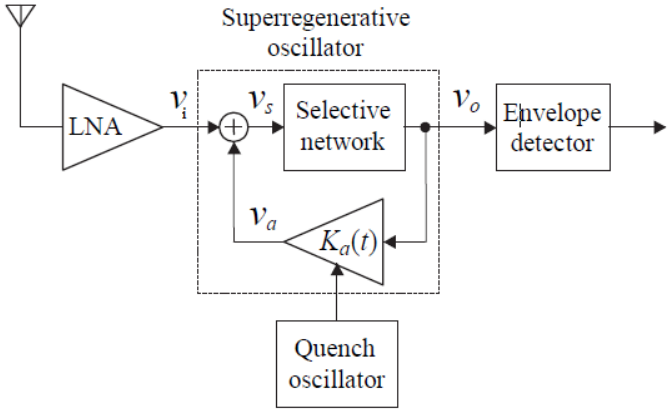

Figure 3.1. *Block diagram of a basic super-regenerative receiver*

frequency quench generator or the quench oscillator is responsible for controlling the damping factor of the oscillator with a specific command called the *quench signal* which controls growth and the cut off of the oscillation[22]. The quench signal drives the oscillator between stable and unstable states. In short, a super-regenerator uses the transient response of an oscillator to filter and amplify the signal[21]. The principle of super-regenerative architecture in pulsed communication

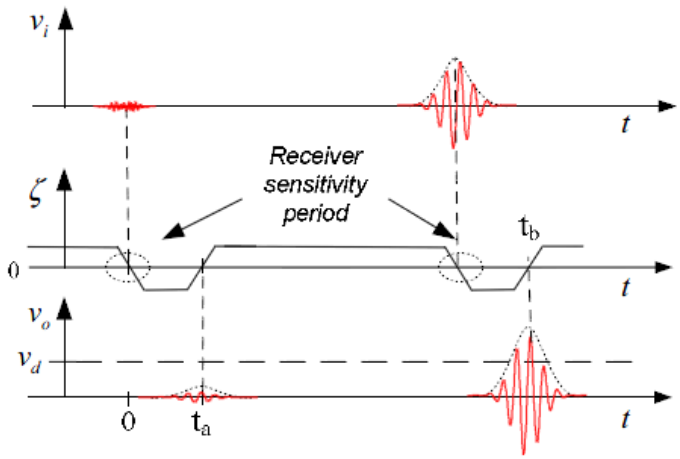

Figure 3.2. *principle of super-regenerative architecture in pulsed communication*

is illustrated in Figure 3.2. When the detector receives energy from the input $v_i(t)$, the quench signal is triggered synchronously. As a result, the damping factor $\zeta(t)$

becomes negative. The super-regenerative samples the input signal at this time and starts oscillating. The time it takes the oscillation to start depends on the amount of the energy received from the input. When the quench signal is switched off, the detector switches to the stable mode and consequently the damping factor $\zeta(t)$ becomes positive. The oscillator is damped until the next phase for input signal sampling. After each quenching, RF oscillation grows exponentially, starting from the tiny energy picked-up by the antenna plus circuit noise. Starting from noise, the amplitude of the resulting self oscillation does not exceed the detection level in the first quench period before damping of the signal by the inactivation of the quench signal at $t = t_a$. When the input signal is large enough within the sensitivity period of the receiver, the oscillation grows faster and reaches the detection level at t_b shown in Figure 3.2.[21]

Super-regenerative architecture in the reader-tag communication

Figure 3.3 shows the basic super-regenerative principle applied in the reader tag communication. As it can be seen from Figure 3.3a the reader transmits an ultra wideband impulse to the tag. The signal pulse will arrive at the tag antenna after a *Time Of Flight* (TOF). The signal received by the tag is attenuated due to propagation loss (Figure 3.3b).

In order for the oscillation to start growing, the signal should arrive on the tag side in the receiver's sensitivity period, i.e. when the damping factor is negative and a quench signal is applied to the tag oscillator (Figure 3.3c). After T_q seconds, i.e. the super-regenerative period, the quench signal is deactivated and the damping factor becomes positive so the oscillation stops growing (Figure 3.3d and 3.3e).

Precise optimizations are needed between the activation of the quench signal and the reception of the incoming pulse so that in t_{sync} when the damping factor of the oscillator becomes negative, the peak value of the incoming pulse would be received (Figure 3.3b and 3.3d). In case exact synchronization is not done, the oscillation will start due to noise. However, the amplitude of the regenerated pulse will not be large enough to be detectable.

On the contrary, if the quench signal and the incoming pulse are well synchronised, the regenerated impulse on the tag side will be detectable(Figure 3.3e). The amplitude of the impulse will be compared to a pre-defined threshold voltage in order to produce the information sent from the reader side at the tag side. Furthermore, the pulse could be sent back to the reader as an acknowledgement signal if direct connection between the tag antenna and the oscillator is obtained (Figure 3.3g).

One of the main issues in impulse ultra-wideband systems is synchronization which is due to low duty cycle and pseudo random timing of pulsed signals, and

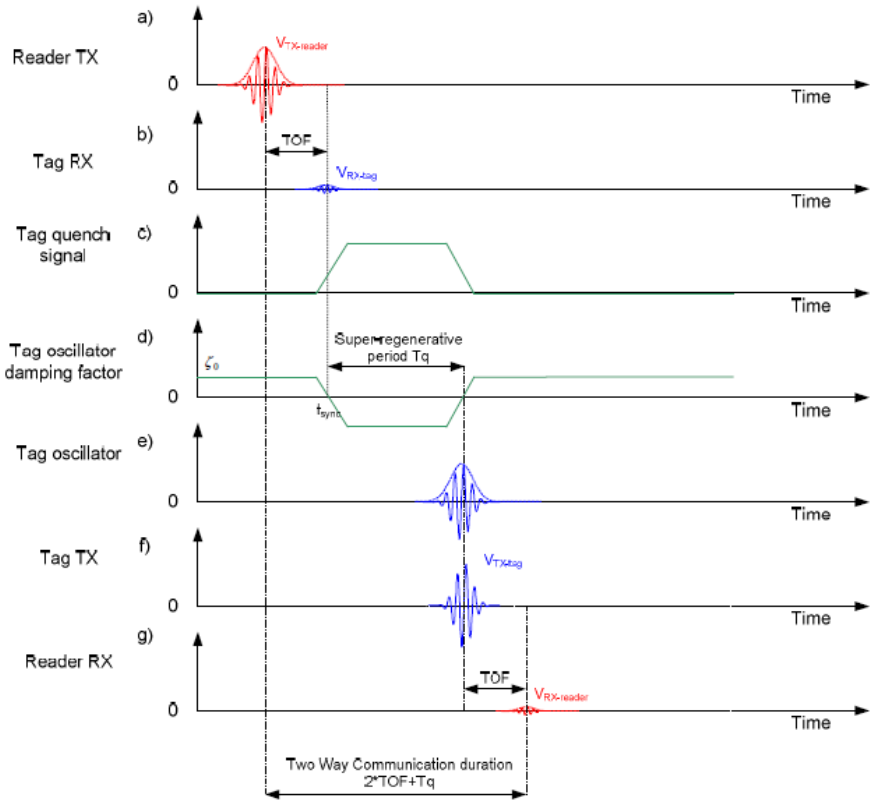

Figure 3.3. *Overview of super-regenerative principle in a reader to tag communication link*

frequency drift and differences of reference clocks between transceivers[19]. In the reader tag communication, frequency synchronization is solved using the mutual narrow band signal.

4. WIRELESS MEMORY SYSTEM: DEPLOYED SOFTWARE AND HARDWARE ARCHITECTURE

This chapter studies the hardware and software architecture of the wireless memory system both on the tag and the host side but more extensively on the tag side which is the target of this project.

4.1 Memory Tag and Host Hardware Architecture

The functional specification of the wireless memory system allows it to operate in different operating modes, i.e. standalone mode (or tag mode), reader mode, peer mode or in development mode. On the tag side, system may function in either standalone or development mode.

Regardless of the operating mode, wireless memory system is primarily composed of two basic sides: The tag side and the host side. The tag side is basically a high capacity memory device (without battery) while the host side is a wireless memory subsystem (Reader/Writer) embedded in a battery powered device such as a mobile phone. The overall architecture of both sides is shortly illustrated in Figure 4.1.

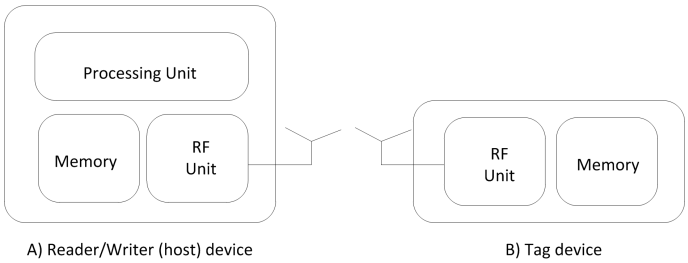

Figure 4.1. *Overall hardware architecture of the wireless memory system*

4.1.1 Tag Architecture

As illustrated in Figure 4.1 the tag side consists of two essential parts: *Radio front-end and Non-Volatile Memory Tag.*

Radio front-end: As extensively explained in Chapter 3.2, the radio unit uses a basic super regenerative transceiver and the communication is done over an impulse UWB link at 7.9 GHz centre frequency. The overall architecture of the RF module is optimized for high data rates with low power consumption within short proximity.

The used data rate is scalable and up to 108 Mb/s can be achieved. Synchronization between the two sides is done using the mutual narrowband signal which is also used as the reference clock.

Memory: Various Non-volatile memory technologies may be used in the tag or the host device. The most important criteria to take into consideration would be providing as high capacity and data rate as possible while consuming as low power as possible. Additionally, deploying a memory with a long life cycle and high number of read/write operations is of great importance. The design can take advantage of technologies like PCM, NAND/NOR flash, DRAM, etc.

As this project was based on the system operation in development mode, the architecture of the design differs from the overall architecture to some extent. Most importantly in development mode, the communication between the host and the tag is wired. A Linux machine is used as the host which communicates with the board (equipped with a cyclone II FPGA, regulators, connectors and converters) which the Non-Volatile Memory is also connected to physically. The overall view of the system's hardware architecture designed for development mode is shown in Figure 4.2.

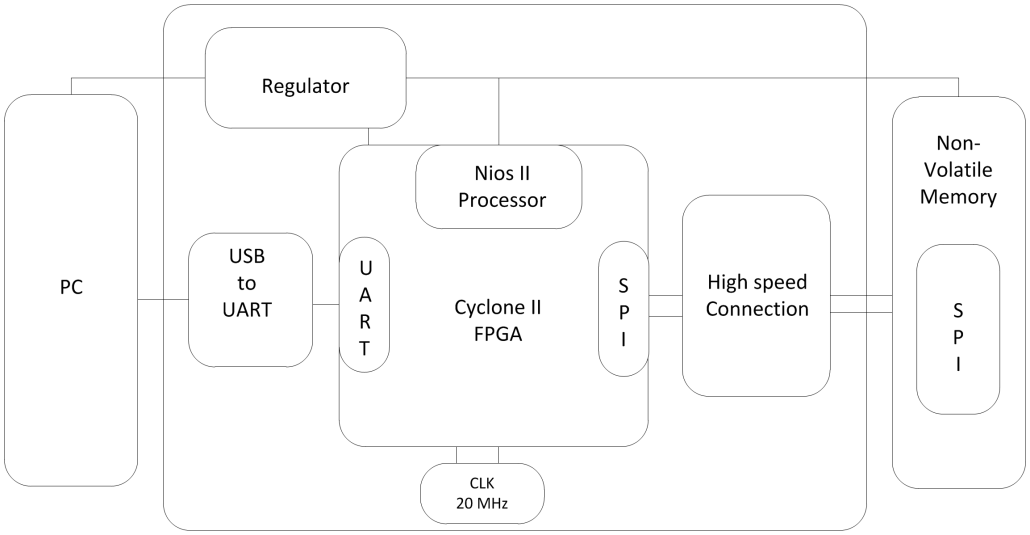

Figure 4.2. *Wireless memory system in development mode*

As shown in Figure 4.2, the host PC is connected to the board via a USB cable and then converted to UART. Then it is connected to Cyclone II FPGA chip's UART port. The Altera Cyclone II FPGA chip is equipped with a Nios II processor which handles all the control and decoding of the commands received from the host and generates appropriate responses.

However, when used in standalone mode, the tag may only take advantage of a finite state machine implemented on the tag to decode the different commands transmitted from the reader over the air.

On the other hand, the Nios II processor used in the development phase is basically responsible of running the FSM on the tag. However, in the target design the tasks of the finite state machine could be executed by just hardware which results in lower costs in terms of power and area.

While in stand-alone mode, the required power for the tag is provided wirelessly using the super-regenerative architecture, in development mode, the regulator on the board will feed the demanded power to different elements such as FPGA and NVM memory. When operating in development mode, the memory is connected to the FPGA chip via an SPI bus. In this case, the memory obtains its required power from the regulators and as a result, higher capacity memory prototypes can be used which cannot be otherwise powered through wireless power transfer.

4.1.2 Host Architecture

In addition to memory and radio unit, the host side also has a processing unit since a great part of the processes concerning the wireless memory system are done in the host device. The host is the initiator and responsible for most of the communication commands and decisions. Specifically in case of security mechanisms, the host implements the standard cryptographic functions while the tag only carries out security functions implementable within its limited processing power.

Unlike the tag, the host is embedded in another device like a mobile phone and thus it is battery powered. This allows the host to employ the more processing demanding tasks.

In the development mode of operation, host is a Linux machine sending all the command and controlling the communication. As it can be seen from Figure 4.2, the commands are sent to the tag device using a USB cable.

The host architecture will not be extensively discussed in this thesis since it was not the target of this project.

4.2 Memory Tag and Host Software Architecture

As in the hardware architecture of the wireless memory system, the software architecture also differs when operating in different modes e.g stand-alone mode and development mode. Figure 4.3 shows the overall architecture of both the tag and the host side when used in stand-alone mode.

As stated earlier in the hardware architecture chapter (chapter 4.1), the implementation of the wireless memory tag in the development mode uses the processor from the Cyclone II FPGA to decode, handle and control all the received commands. Cyclone II FPGA devices support the Nios II embedded processor that allows custom-fit processing solutions.

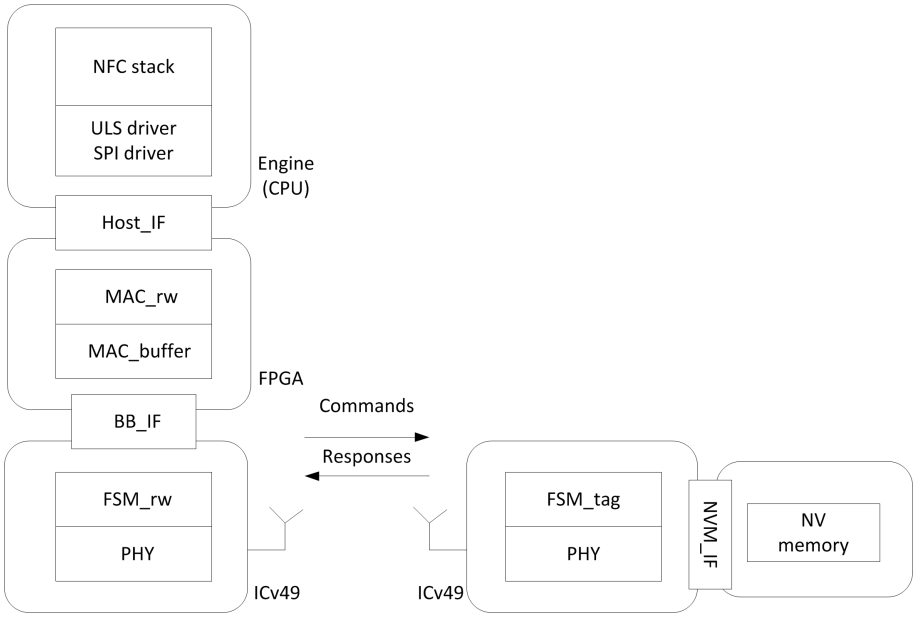

Figure 4.3. *Overall software architecture of the wireless memory system*

There are three different configurations of the Nios II processor available: fast, standard and economy. This design uses the Nios II fast core which is designed for fast performance. As a result, this core presents the most configuration options allowing to fine tune the processor for performance.[5]

Nios II's fast core performance at 100 MHz is estimated to be up to 101 DMIPS (Dhrystone MIPS). It uses about 1400-1800 logic elements. The processor is configured to utilize 4 Kbytes of instruction cache and 2 Kbytes of data cache with line size of 32 Bytes in this project. The Debug module uses level 2 debugging i.e. JTAG target connection, download software, software breakpoints, 2 hardware breakpoints and 2 data triggers and uses 800-900 logical elements.

The processor and the FPGA run on a 100 MHz clock which is generated from a 50 MHz crystal by the FPGA's internal PLL.

The Nios II processor uses two interfaces:

1. **SPI(Serial Peripheral Interface)**: The SPI interface connects the Nios II processor to the non-volatile memory. SPI runs at 1 MHz clock and this speed is due to the slow EEPROM used during the development of this project. The V49 IC is capable of 100 MHz SPI speed if a suitable memory chip is used.

2. **UART(Universal Asynchronous Receiver/Transmitter)**: The UART interface connects the Nios II processor to the PC which operates as the host during the development of the project. The UART emulates SPI interface that is used on the V49 IC and can be directly replaced with a SPI interface. The UART runs at 115200 Hz.

4.2.1 Security Software Architecture

The security functionality is located in the middle of the NVM_IF in the tag device during the development of the wireless memory security. The location of the security software in the overall tag software can be observed in Figure 4.4.

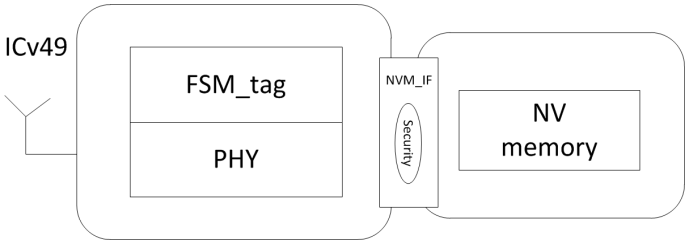

Figure 4.4. *Position of the wireless memory's security software in development mode*

However in the final design of the wireless memory (not during development), The security software would be located on the v49 IC before the NVM_IF which can be seen in Figure 4.5.

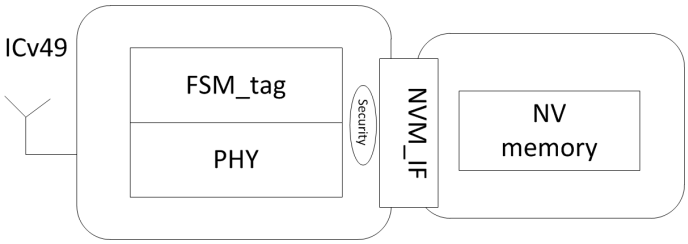

Figure 4.5. *Position of the wireless memory's security software in the final implementation*

With the reception of each command from the UART interface from the PC which acts as the host, it is parsed into a code, address and data. The first 8 bits i.e the code indicates what operation is expected to be done in the memory. The next 24 bits indicate an address in the memory and following that is the data to be written to the memory or zeros in case of a read command. The security software on the processor decides based on the code and the address which functions or operations needs to be done in the wireless memory.

Detecting the "write" code, the program decides based on the address the next set of operations which can be authentication, PIN/name/ID operations, PIN transfer or normal operation regarding "read"s or "write"s to a segment which may itself be in a specific life-cycle model. If the address belongs to the group of addresses used for registers, then actions are taken accordingly for the PINs, authentication, name and ID operations. However, in case the address belongs to the access controlled area of the memory, then the configurations of that segment are checked in the management area in order to take proper following steps.

The overall structure can be viewed in a UML diagram in Figure 4.6.

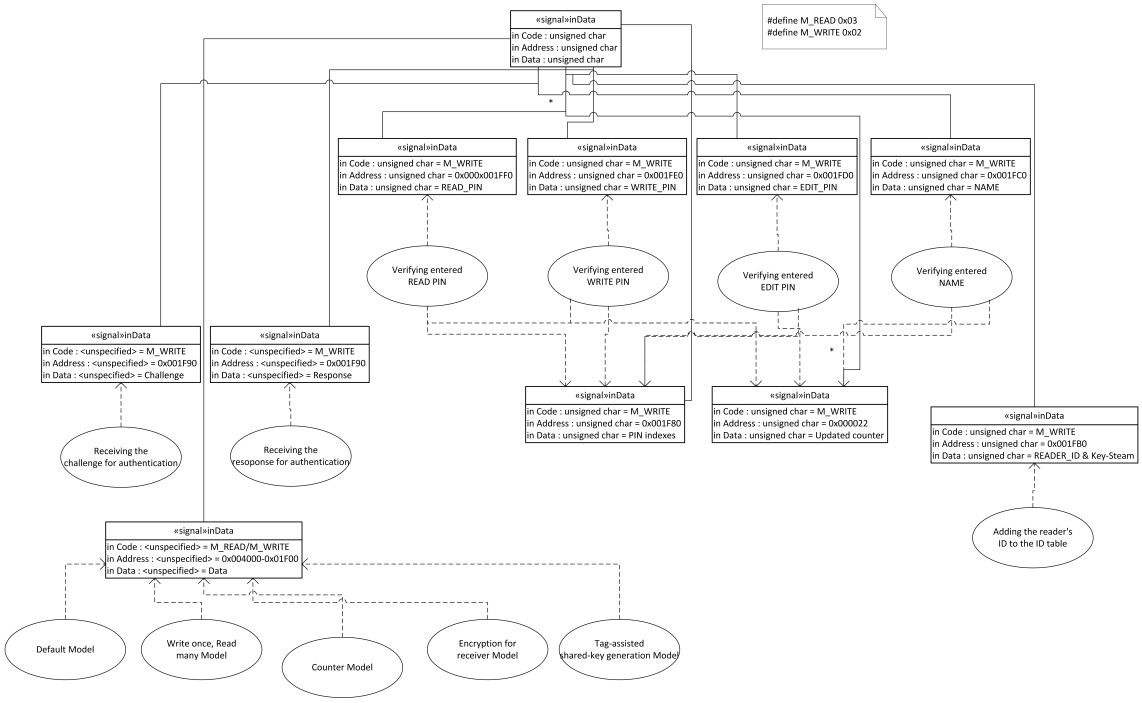

Figure 4.6. *The security function in a UML diagram*

5. THREAT ANALYSIS AND SECURITY ARCHITECTURE

Threat analysis explains how potential adversaries exploit system weaknesses to attack the system [36]. Furthermore, threat analysis determines possible threats and distinguishes mechanisms to decrease or completely remove the risk of attacks in a specific architecture. Possible security threats are extensively discussed in chapter 2 at general level. In this chapter, some of these threats are looked at and analysed in the context of the wireless communication between the tag and the host device. Following the threat analysis, the implemented security architecture for the wireless memory is explained.

5.1 Threat Analysis for a Security Memory

The information exchanged between the tag and the host device can be eavesdropped by an adversary to be used for impersonation attacks i.e the eavesdropper listens to the communication between the two sides and can later take advantage of the information achieved by eavesdropping. This information can be also used to imitate the operations of either a genuine host or a genuine tag. Information acquired by eavesdropping attacks are only useful for the attacker if the exchanged information is unencrypted. As an example an attacker may listen to the communication between a host and a tag when a host is transferring PINs to the tag to obtain access to the memory and later use the PIN to attain access to the tag. This can be prevented by encrypting the sensitive information moved between the two parties.

Impersonation happens when a malicious attacker imitates one of the parties in the tag-host communication to be able to access the information in the memory. One case can be e.g. a fake tag impersonating a genuine one and requesting PINs from a genuine host to obtain access to PIN protected parts of the memory. If the host lacks a technique to verify the genuineness of the wireless memory, it may simply disclose PINs to any requesting tag. To prevent impersonation attacks, an authentication technique must be employed in the wireless memory's security architecture. Man-In-The-Middle attacks are, as stated in chapter 2, a special case of impersonation attacks.

Denial of service threats aim to use up most of the system's available resources and leave the system with no or limited means to provide services. An example case

of denial of service attack that may occur in the tag-host communication can be randomly changing PINs when transferring PINs from host to the tag for accessing the protected areas in the tag. This section addresses the wireless memory's organization, layout and operations implemented in relation to the security of the wireless memory. The memory organization section describes how the memory is divided into different segments. The layout section explains in details the different sections layout and their purpose. The wireless memory operation section defines the control and security operations related to the wireless memory.

5.2 Wireless Memory Organization

Wireless memory security means wireless memory access control, authentication and secure life-cycle management, as well as the confidentiality and integrity of the wireless channel between the host and the wireless memory. To handle access control and other security operations on the memory, the memory is organized into different segments or areas. This segmentation of the memory area may differ from one memory to another and the information regarding these configurations is stored in the *master* area of the respective card. The overall layout of the card can be defined as follows:

Memory is logically divided into segments of equal size i.e in this project 4KB (4096 Bytes) each. Some of these segments are reserved for management purposes. These segments are defined so that the resulting overhead from these segments will not exceed 2% of the whole memory area. The rest of the memory area consists of some access controlled segment whose control data resides in the management segments and also a publicly accessible part. The exact ranges and borders of these logical partitions are fixed by card type. The overall logical structure of the memory can be observed in Figure 5.1

The access control options wherever possible reflect the fact that the data written to memory will be file systems rather than individual files. However, a few tag specific security services diverge from this rule e.g access control configurations need to be either implemented as part of the file system or as a service beside it.

5.3 Wireless Memory Layout

A more specific layout of the memory is defined as follows and illustrated in Figure 5.2.

In this project the segments are all designed to be 4KB in size.

The master segment contains user-specific data and state relating to the whole card. E.g. the division between the areas for management, access controlled data and public memory areas.

Figure 5.1. *Overall memory layout*

The PIN segment contains 256 PINs that can be used as read, write or edit PINs. The access controlled segments can be associated with any of these PINs.

The Reader ID segment holds the IDs of the devices touching the tag. A host may freely choose to give its Id to the tag or to remain anonymous. However, some (searching / listing) services can make use of a host id to list e.g. segments originating from a specific host.

The management segment(s) contain access control information for the individual 4kB segments in the access controlled area. The format of the management segments and policy is further discussed later in this chapter.

The access controlled segments are conditionally accessible based on policy defined in the management segment(s)

The public memory area is publicly readable/writeable.

An example layout for the memory used in this project's development is shown in Figure 5.3.

5.3.1 Master Area Layout

The master area is located in the first addresses of the wireless memory tag. This segment contains user-specific data and state relating to the whole card e.g. the division between the areas for management, access controlled data and public memory areas. It also holds master PINs, state flags and roll-back counter which will be explicitly discussed in the following sections.

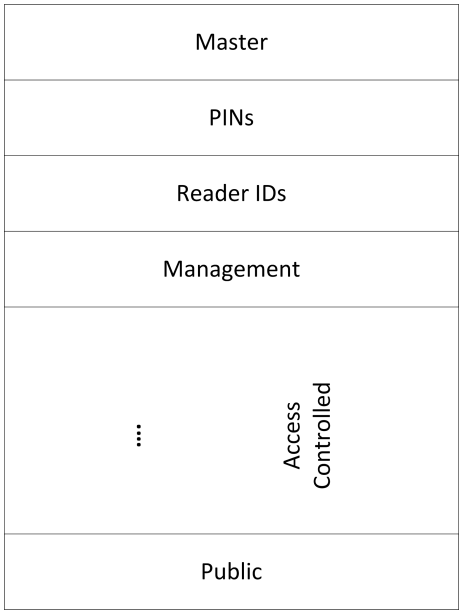

Figure 5.2. *Detailed memory layout*

For security intentions master area is divided into one readable part and one hidden part. The partitions are shown in Figure 5.4.

5.3.2 PINs Area Layout

PIN segment, like all other segments is $4K$B in size and consists of 256 PINs which can be used as read/write/edit PINs. Each PIN is designed to be 16 byte. The 256 PINs in the PIN area are indexed from 0 to 255. The PIN area is illustrated in Figure 5.5.

5.3.3 Management Area Layout

The 4 KB of available memory in the management area is mapped into units of 32 Bytes each. Each of these units corresponds to a segment of the memory in the access controlled area and contains the access control information of the respective segment. An example of management units associated with access controlled segment is illustrated in Figure 5.6. The correspondence is purely based on the addresses meaning that the unit with the lowest address position in management area provides for the first access controlled segment.

Each management unit can be separated into 5 essential parts:

Control byte contains the access control information such as read/write access for the corresponding segment.

Life-cycle model more complex control solution for some specific segments. These solutions include *"write once, read many"*, *"counter"*, *"encryption for receiver"*

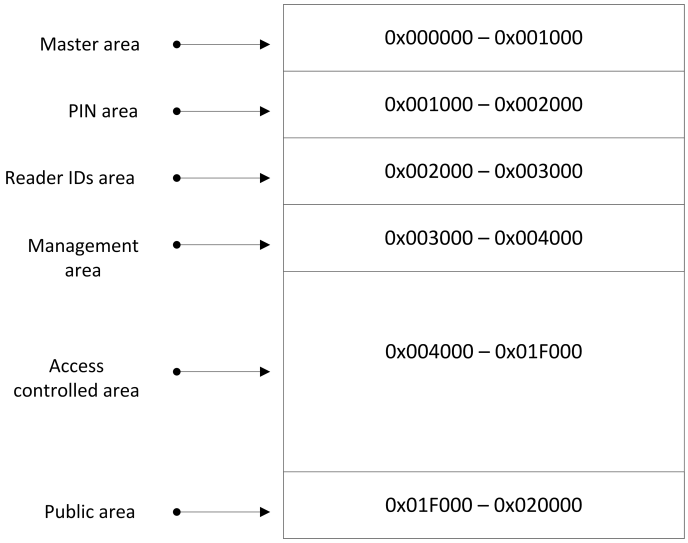

Figure 5.3. *Example memory layout*

Figure 5.4. *Master area layout*

and *"tag assisted shared key"* life-cycle models.

PIN counter keeps track of number of entries. It is used for retry protection.

PIN indexes are indexes to PINs in PIN area which represent the PINs for read-/write and management area modification (edit) protection of the corresponding segments. 2 Bytes are reserved for each index which is for future proofing since for the 256 present PINs, 1 Byte would be enough. The indexes are arranged in most significant bit order (MSB).

Name is a 16 byte field in each management unit. This 16 byte name can be used as a segment specific password for reading and writing. It is also used for file-system data management.

The overall layout of each management unit is illustrated in Figure 5.7.

Next, each individual attribute of the management unit is more explicitly discussed.

CTRL byte First byte of the control unit is the *control byte*. Control byte contains information that defines basic access control rules for the segment. The CTRL bit has the following bits defined according to Figure 5.8

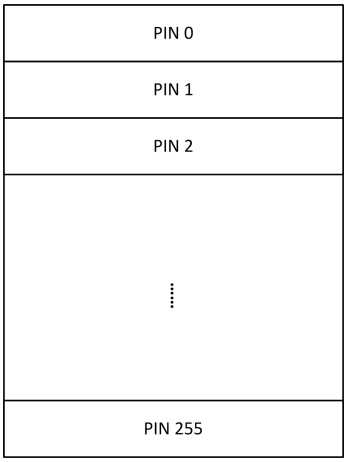

Figure 5.5. *PIN area layout*

RD : A set RD bit indicates a read enabled segment. The segment is readable unless the readability is controlled by a read PIN. Furthermore, a segment with RD bit set to zero cannot be read even after providing the right read PIN. Meaning that a set RD bit along with the right PIN (in case the segment is protected with a read PIN) are both needed to access a segment. For models (which will be described later) the read bit will also indicate whether the model is in a state where the segment currently is readable.

RD PIN : When RD PIN bit is set, read PIN is enabled. This means that reads from the segment would be protected by a 16 byte PIN and to have read access the right PIN needs to be written in the read PIN register. The *pointer to read PIN* field in the management area indicates which of the 256 PINs needs to be provided. It is important to note that even with the right read PIN, the segment would not be accessible if the RD bit is not set.

WR : A set WR bit indicates a write enabled segment. The segment is writeable unless the writability is controlled by a write PIN. Furthermore, a segment with WR bit set to zero cannot be written to even after providing the right write PIN. Meaning that a set WR bit along with the right PIN (in case the segment is protected with a write PIN) are both needed to access a segment. For models (which will be described later) the write bit will also indicate whether the model is in a state where the segment currently is readable.

WR PIN : When WR PIN bit is set, write PIN is enabled. This means that writes from the segment would be protected by a 16 byte PIN and to have write access the right PIN needs to be written in the write PIN register. The *pointer to write PIN* field in the management area indicates which of the 256 PINs needs to be provided. It is important to note that even with the right

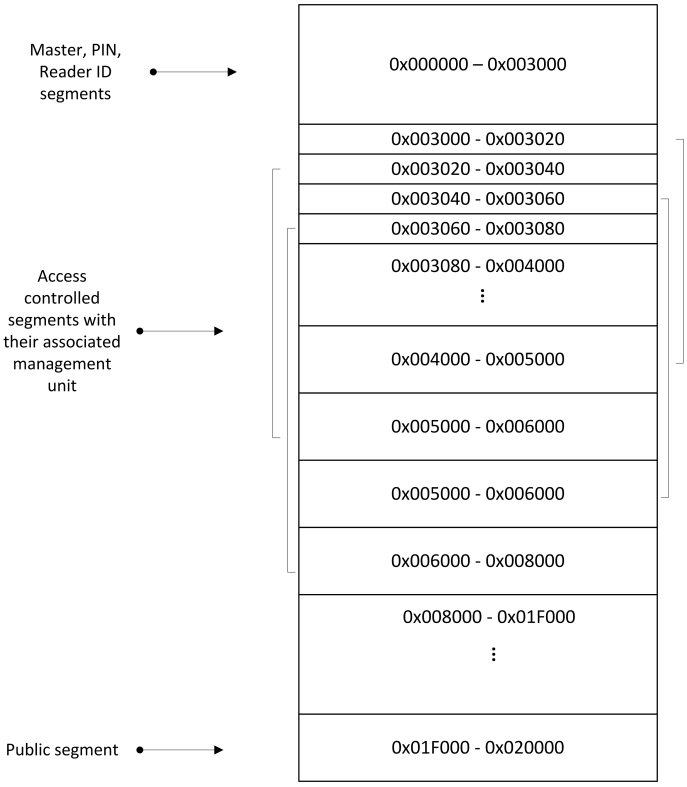

Figure 5.6. *An example of management units with corresponding access controlled segments*

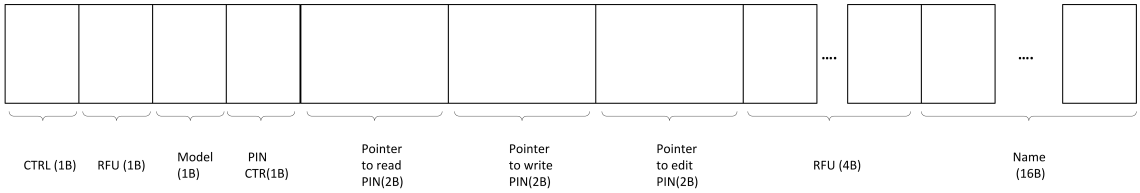

Figure 5.7. *Management area layout*

write PIN, the segment would not be accessible if the WR bit is not set.

PN : if the PN bit is set, the name will operate as a password. If enabled, the contents of the name field in the management area should be present in the name register in order to access the segment. In this case the name field in the management unit is not readable to users. This validation works in addition to any other read/write PIN protection that may be enabled for the segment. If not set, the contents of the name field will be readable to users.

nE : when nE bit is set the management unit of the corresponding segment will be locked and would not be editable even by the owner. This is a permanent condition, and is mainly used for counter implementation. It can be possible that the nE bit would be hard-wired for a couple of predefined segments which may be needed for specific applications. In that case neither the nE bit nor

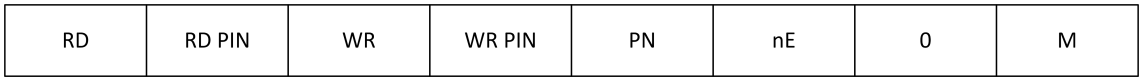

Figure 5.8. *Control byte layout*

the segment can be edited by the host or the owner.

RFU : This bit is reserved for future uses.

M : setting this bit enables the model (extension) field. There are a number of life-cycle models available that can be used for different segments and put additional constraints on the access to the segment. M bit set to 0 indicates the default model for the segment that acts purely according to the previously explained bits.

Model byte The security architecture provides a set of additional access control solutions and operations which enables use of segment in special use cases and during the memory's life-cycle. Given that the model bit (M bit) in the CTRL byte is set, this byte defines the life-cycle model operating on the respective segment. The 4 bits addressed in Figure 5.9 are currently used to indicate the model.

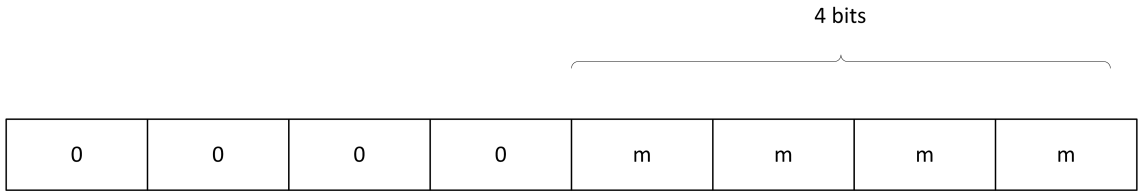

Figure 5.9. *Model byte layout*

The currently defined models are listed below. More extensive and illustrative declarations are included in the following chapters.

Default model (0)

Default model is the implicit model. When the "M" bit in "CTRL" byte is set to zero, the default model is activated.

Write once, read many model (1)

When operating in this life-cycle model, the segment can only be written to. Once written, the segment can only be accessed to be read and no write operation is allowed any more. However reading the contents of the segment is allowed as many times as desired.

In this mode, R/W bits are reflected according to model state but PR/PW/PN bits work as normally.

Counter model (2)

The counter model defines the segment to operate as a number of counters. Each counter consumes 8 Bytes and the segment can contain a certain number of counters working independent of each other. However all the counters belong to a single segment, and the configuration data is shared among all of them.

Using a segment in counter model sets the RD and WR bits in CTRL byte in management unit to allow read and write access, however, "RD PIN", "WR PIN" and "PN" bits can be configured according to application. For many use cases it may be required that the "nE" bit would be set so that modification on the counter segments becomes impossible for the user. This also brings restricted use for the owner too.

The 8 byte counter works in a manner that its value will only be updated if the new value to be written is the old value of the counter slot plus one. Other than this case writes to the counter will fail.

Encryption for receiver model (3)

This model implements a means to transfer a string to a receiver in an encrypted format such that the resulting crypto-text can be read from the tag and re-transmitted by some other means to the expected receiver. This operation is done in a few phases:

- The receiver generates a key stream for the specific segment operating in this model. This key stream is generated in a way that the host can later recreate the key stream.
- Next, this generated key stream is written to the specified segment. In this phase the segment is only writeable and not readable in the tag interface.
- Finally, the sender writes the plain-text to the segment. This write operation is a done in a specific manner by the tag. The tag *XOR*s the plain-text with the provided key stream by the receiver(which is currently written in the segment) and thus produces a crypto-text. This crypto-text is then written to the segment. In this phase the crypto-text in the segment becomes publicly readable, however, further writes to the segment will fail.

Tag-assisted shared-key generation (4)

This model creates a tag-assisted mechanism for generating shared key streams between the various hosts using the tag. This mechanism can be optionally activated by the hosts and a host not willing to use shared keys can freely use

the tag. This mechanism operates essentially like the previous model but in a higher granularity. This operation can also be divided into two phases:

- When encountering a tag for the first time, each host can write its own specific key stream to the tag.
- After each new host entry, the tag produces pair-wise key streams between all hosts using all the key streams. The produced keys will be stored in the segment and will only be readable by the hosts that know how to reconstruct their own individual key streams.

These shared keys between two individual hosts can be used to encrypt information, when transmitting it on the wireless memory between the paired hosts.

5.4 Wireless Memory Operation

5.4.1 Authentication

In order to ascertain the genuineness of a tag when it initiates its first communication with the host, the host needs to authenticate the tag. Fake tags trying to connect to the host memory will be known and denied access. Authentication mechanism is more thoroughly discussed in section 2.2.5.

The wireless memory tag uses a challenge response mechanism to authenticate genuine tags. Authenticated tags will be marked with setting the authentication flag bit which is located in the master area of the memory tag. When connecting to memory tag for the first time, the host checks the authentication flag before any other operation. If the bit for the flag is set, a second time authentication is not needed. On the other hand, in case of a cleared bit, the authentication process should take place primarily.

Overall Lamport signing operation can be seen from Figure 5.10.

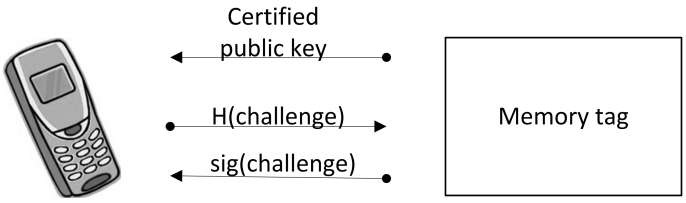

Figure 5.10. *Using Lamport signature on the tag*

For each bit in H(challenge) (shown in Figure 5.11), the corresponding hash value is selected from the memory and is returned to the host.

The host device sends 256 challenge bits to the tag and receives 32 bytes for each bit in return. In this project the Lamport signature is implemented so that in

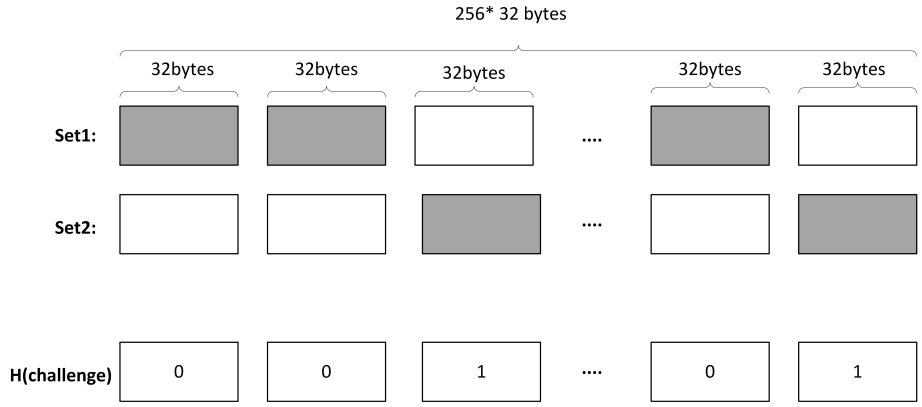

Figure 5.11. *Lamport signature operation*

each write cycle, the host device writes a one bye challenge to the tag. This write operation has to be done to a special register with a known address used for the authentication purpose. After each write, first the authentication flag in the Master area of the memory is checked to determine whether the tag is authenticated or not. In case of an unauthenticated tag, for each bit in the written 1 byte, 32 bytes will be selected from either of the 2 key sets and returned to the tag. If the bit is "0", 32 bytes will be selected from the first key set. Otherwise, in case of a "1" bit, 32 bytes are selected from the second key set. This return is then followed by erasing the corresponding 32 bits from the memory. This procedure is continued until the whole $256 * 32$ bytes are returned and erased. After the authentication process is finished, the authentication flag will be set in the master area. If the host chooses not to authenticate the tag, sending 256 bits of zero to the specified address will bypass the authentication process.

5.4.2 PIN Operation

Each access controlled segment is protected by 3 different PINs. These PINs protect segment when reading from or writing to the segment in addition to protecting the edition of a segment's management area. 256 PINs are stored in the non-readable PIN area discussed in section 5.3.2, and operations with PINs are done with help of a number of registers.

Registers : 7 registers of 16 bytes each are used to hold the PINs, Name or Reader ID. These registers use the final addresses of the master area segment. These registers are:

READ_PIN_REG_ADDR : Address used for the register holding the PIN when reading from the segment (PIN challenge-response).

WRITE_PIN_REG_ADDR : Address used for the register holding the PIN

when writing to a segment (PIN challenge-response).

EDIT_PIN_REG_ADDR : Address used for the register holding the PIN when modifying a segment's management area (PIN challenge-response).

NAME_REG_ADDR : Address used for register holding the name for a segment (PIN challenge-response).

ID_REG_ADDR : Address used for the register holding the identity of the host, if provided (this is not an encrypted transfer). In case of commit register transfers the contents of this register identifies the owner domain.

COMMIT_REG_ADDR : Commit register (encrypted under one of owner's PINs)

PA_REG : PIN access register. The structure of this register can be seen in Figure 5.12.

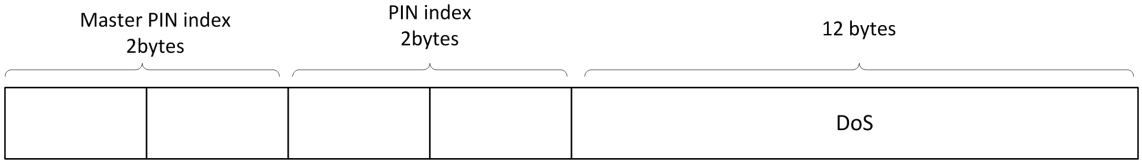

Figure 5.12. *PIN access Register (PA_REG) structure*

Roll-back counter : All registers used for PINs support encrypted transfer of the PINs to the wireless memory. This encryption is performed in *counter mode*. Transfers in counter mode employ a card-specific global *roll-back protection counter* (RBPCTR). Basically each PIN has to be encrypted with the roll-back counter before being written to the wireless memory's commit register.

The roll-back counter operates just as the other counters in the memory architecture. It consists of 8 bytes and can only be updated if the value of the counter plus one is written to it. However, the memory should provide a dedicated persistent roll-back counter for PIN transfer purposes.

The roll-back counter is additionally associated with a *usage flag*. The purpose of the usage flag is to make sure that the value of the counter is updated before being used for encryption. Whenever a write to one of the PIN registers is attempted, the usage flag is checked. Write attempt will fail in case the usage flag is set, otherwise, the usage flag will be set to 1 and the write will succeed. Each time the counter is updated the usage flag is cleared.

Master (owner) PINs There are a number of master PINs in the master area of the wireless memory. These master PINs should be employed when changing or uploading new PINs to the PIN area. Another rather important usage of master PINs is in configuring the access control system. Access controlled segments can be configured individually or in groups. Master PINs help enable editing of the management area in groups using one master PIN for all segments in the group rather than using the individual edit PIN allocated to each segment.

The main reason for having several master PINs is to support backups or emergency recovery of data. In addition to these reasons, the additional master PINs can be used for sharing tag between e.g. family members, employees of a company, etc.

Owner PINs are predefined and cannot be changed. These PINs are referred to by indexes e.g. first slot is referred by index $0x00$, second slot $0x01$, etc. These PINs may even not exist depending on the usage purpose of the tag. On the contrary, the PINs can be more complex if the manufacturer of the wireless memory wants some manufacturer-specific backup/restore mechanism.

PIN management The operations related to the PINs can be categorized into *using PINs* and *transferring PIN*. The former refers to the usage PINs when attempting to read/write/edit a PIN protected segment. The latter describes actions needed when adding or modifying a PIN in the PIN area.

Both operations are protected against passive and replay attacks. Although the authorization for accessing the segment is protected, the data transfer to or from it is not protected if the data encryption already in the host is not applied.

The PIN transferring and usage is protected using block cipher encryption function E. The encryption function E used in this project is XXTEA block cipher described in section 2.2.1. XXTEA has a key length of 16 bytes and operates on 16 byte data blocks.

Using PINs : PINs are used in an implicit challenge-response protocol based on the roll-back counter. Every time a submitted PIN is validated, it is maintained in an internal register for faster operations in case of e.g. file-systems that may share a PIN among a number of segments. The PIN is maintained in the register until a power-down occurs. All indexes are reset to zero at power-up.

The operation illustrated in Figure 5.13 can be described in 4 steps:

1. The host uses the first bytes of the PIN access register (PA_REG) to indicate which PIN is going to be used.

2. Next, the PIN in given position is encrypted using the previously updated

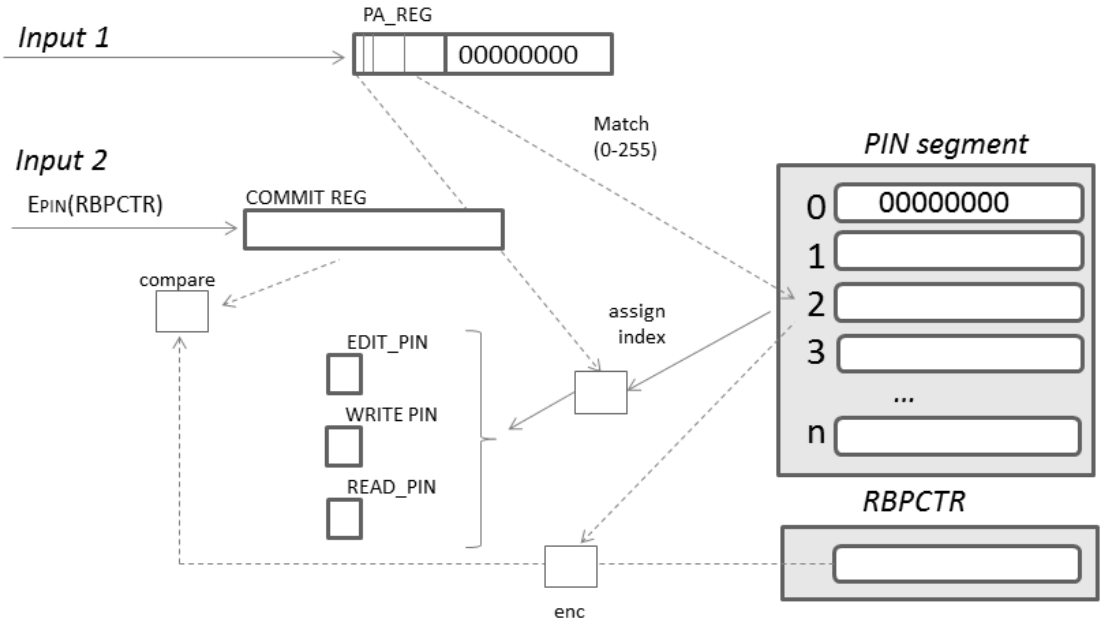

Figure 5.13. *Using PINs operation*

counter by the host ($E_{PIN}(RBPCTR)$). Meanwhile the tag calculates the encrypted hash value for the given PIN using the same encryption function.

3. Then the encrypted PIN provided by the host is compared to the one produced by the tag. If the values match, the index of the used PIN is written to the relating internal register.

4. Finally when the host attempts to access the protected segment with the aforementioned PIN, the access is granted if the index in the internal register matches the index of the PIN used for that segment. If the indexes do not match the access is denied.

PIN in location zero of the PIN area is reserved to contain all zeros. This can be used for transferring names and IDs from hosts that are not aware of the PINs for this specific card. As a result the encryption for the register will be still done, however using a globally known key.

Transferring PINs : All PINs are transferred to PIN area in an encrypted format. The operation illustrated in Figure 5.14 can be described in 3 steps:

1. The host uses the PIN access register (PA_REG) to indicate which position in PIN area the PIN is intended for, which master (owner) PIN is used for encryption and a hash value used for protection from Denial of Service (DoS). The denial of service protection field is used to make it difficult to mount attacks by which PINs are randomly changed to cause denial of service. The

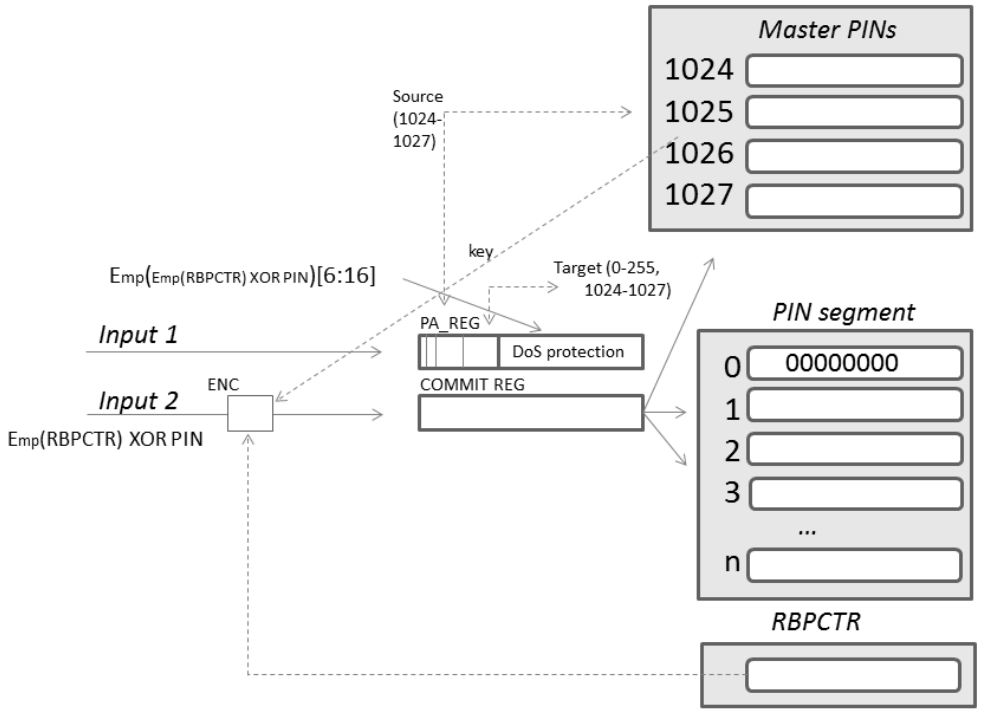

Figure 5.14. *Transporting PINs operation*

DoS value is obtained by encrypting the value written to the COMMIT_REG with the specified master PIN and choosing the last 12 bytes. Assuming encryption function E, this means

$$E_{masterpin}((E_{masterpin}(RBPCTR))XORPIN)[4..16]$$

The structure of the PA_REG can be seen in Figure 5.12.

2. Next, the specified master PIN in the PA_REG is encrypted with the counter ($E_{masterpin}(RBPCTR)$) and then the product is XORed with the PIN which is going to be written in the specified position in the PIN area. Assuming encryption function E, the host will write:

$$(E_{masterpin}(RBPCTR))XORPIN$$

to the COMMIT_REG.

3. If the DoS value matches the expected one, the values are decrypted in the wireless memory and the PIN will be written to the specified position in the PIN area.

Read PIN operation To read from a *read PIN protected segment*, the PIN whose index is found in the management area of the segment should be provided to the tag by the host. To do so, the host should first write the 16 byte read PIN to the register in the address READ_PIN_REG_ADDR and then attempt to normally read from the segment. Then If:

1. the RD bit in the segment's management area's CTRL byte is set and
2. the RD PIN bit in the segment's management area's CTRL byte is and
3. the PIN written to READ_PIN_REG_ADDR address matches the one that the pointer in the management data points to,

   the segment can be accessed for reads by the device.

Write PIN operation To write to a *write PIN protected segment*, the PIN whose index is found in the management area of the segment should be provided to the tag by the host. To do so, the host should first write the 16 byte write PIN to the register in the address WRITE_PIN_REG_ADDR and then attempt to normally write to the segment. Then If:

1. the WR bit in the segment's management area's CTRL byte is set and
2. the WR PIN bit in the segment's management area's CTRL byte is set and
3. the PIN written to WRITE_PIN_REG_ADDR address matches the one that the pointer in the management data points to,

   the segment can be accessed for writes by the device.

Edit PIN operation To edit a segment's management unit, the PIN whose index is found in the unit should be provided to the tag by the host. To do so, the host should first write the 16 byte edit PIN to the register in the address EDIT_PIN_REG_ADDR and then attempt to normally write to the management unit. Then if:

1. the nE bit in the management unit's CTRL byte is cleared and
2. the write does not exceed one unit's boundary and
3. the PIN written to EDIT_PIN_REG_ADDR address matches the one that the pointer in the management data points to

the management unit can be edited by the host.

5.4.3 Name Operation

Name field can be used as capability i.e the name located in NAME field of the management unit operating as an additional PIN for the segment. The last 16 bytes of the management unit are assigned as the name field and are hidden from the host in case the name works as capability.

The *NR* bit in the management area is what defines the name field's operation. If the NR bit is set, the name field will act as a capability. When accessing a segment with NR bit set, the name should be written to the register NAME_REG_ADDR. If the name written to the register matches the name in the segment's corresponding name field, the read/write access will be granted to the host.

5.4.4 Reader ID Operation

A full segment in the memory layout is dedicated to storing the reader IDs. This segment works as a table holding the IDs of all the hosts that have communicated with the tag.

Whenever a device touches the tag two scenarios may happen. First, if the device has never touched the memory tag before, its ID will be added to the first empty position in the reader ID area in the memory. The device can also freely choose not to add its ID to the table by sending all zeros as its ID. Second scenario is that the device has communicated with the tag before in which case the table will not be updated.

The operation related to writing the reader ID to the ID segment is done via a register located in address ID_REG_ADDR.

5.4.5 Management Area Operation

As stated in section 5.3.3, another full segment in memory is dedicated to data related to the access management of the access controlled segments. In this project there are 27 access controlled segments of 4 Kbytes. 32 byte of management data is assigned to each of the segments which are used to configure segment's access operations. The overall layout of the contents of this 32 bytes can be observed in Figure 5.7.

Editing management area is protected with *edit PINs* which area allocated to each of the 32 byte units. Other than using individual edit PIN for each unit, master PINs are also defined to enable editing of several units in management area.

To protect the information in the following management unit, no read or write operations occurring in this area can exceed a units boundary. This results in writes of maximum 32 bytes in size. The other restriction on read attempts from the management unit limits the reads to the first four bytes which includes CTRL,

RFU, MODEL and PIN CTR bytes. The remaining 18 bytes will be masked by zeros. Additionally, all writes to the management unit are accepted on the condition that the nE bit in the management unit is cleared.

To edit management area, either the PIN whose index appears in the unit or one of the master PINS should be written to the register in address EDIT_PIN_REG_ADDR. If the submitted PIN is validated, the index will be maintained in the internal register to enable editing of other segments using the same PIN. No editing attempt will be successful in case the nE bit in the CTRL byte of the management unit is set.

5.4.6 Life-cycle Models

When defined to operate in a specific life-cycle model, a segment engages additional specific behaviour to the default access settings provided. Applying a life-cycle model to a segment is done by setting the model bit in CTRL byte and selecting the appropriate model using the model byte in the segment's management area. There are 5 life-cycle models implemented on the tag architecture:

Default model: Read and write accesses are controlled according to the CTRL byte in the management area.

Write once, read many model: A segment operating in this life-cycle is initially write-only and not readable. After the first write, segment's management data alters in a way that it becomes read-only and not writeable.

Counter model: A segment operating in counter model grants all read accesses, however only writing the contents of the counter plus one is successful. The counter model only allows writes that start from the beginning address of each counter.

Encryption for receiver model: This encryption model takes advantage of *XOR* operation and thus it is also called *XOR model*. XOR model can be divided into three stages:

1. **WO1:** In this stage the segment is write only. Normal write can be done to the segment in this stage. Writes can be protected with PINs.
2. **WO2:** Like the first stage, this stage is also a write only stage. However, all data written to the wireless memory at this stage will be first XORed with the key-stream written to the same location in the previous stage. This "XOR" write can be also protected with PINs.
3. **RM:** In the third stage, the contents of the segment can be only read.

The state of a segment in the XOR model cannot be modified by editing the management data but can be changed using a specific operation.

Tag-assisted shared-key generation model: As explained in section 5.3.3, the purpose of this model is to generate shared key-streams between various hosts using the tag. These generated key streams are intended for secure transfer and integrity.

Whenever a device encounters a new wireless tag, it can submit a key-stream to it. The tag checks for ID overlapping and if the ID is detected as new, writes it to the reader ID area of the memory.

The process happening in the card after inserting a new id{x+1} and key-stream KS{x+1} can be described as:

For $i = 1$ to x:

$KTS\{i\}[x+1][id] = KTS\{i\}[x+1][id]$ XOR $id\{x+1\}$

$KTS\{x+1\}[i][id] = KTS\{x+1\}[i][id]$ XOR $id\{i\}$

$KTS\{i\}[x+1][key] = KTS\{i\}[x+1][key]$ XOR $KTS\{i\}[x+1][km]$ XOR $KTS\{x+1\}[i][km]$

$KTS\{x+1\}[i][key] = KTS\{x+1\}[i][key]$ XOR $KTS\{i\}[x+1][km]$ XOR $KTS\{x+1\}[i][km]$

$KTS\{i\}[x+1][id] = 0$

$KTS\{x+1\}[i][id] = 0$

End

where the size of one key-stream slot (KTS) is $len(id) + 2 * len(key)$.

Of the whole key-stream written by a host device, only the parts that are not yet used up for generating shared keys with other devices are readable and accessible by devices. beginning parts of the key-stream which have gone through the shared key operation are hidden from host devices.

devices can also choose to only write their IDs and not provide a key-stream for shared-key generation. The IDs of these devices are written to the reader ID area of the wireless memory from ending addresses of the segment, in contrast

with other devices with shared keys whose IDs are written to the reader ID area from the beginning addresses of the segment.

6. SECURITY ANALYSIS AND IMPLEMENTATION

6.1 Security Analysis

This section addresses the security mechanisms employed in the wireless memory's architecture and how they help solve possible security threads.

Some of the threads have been dealt with using memory life-cycle models that can be optionally activated or deactivated on specific sections of the memory. On the other hand, some mechanisms are automatically adopted in memory operations.

First thread to be prevented when encountering a wireless tag for the first time would be impersonating, that is a faulty or fake memory imitating actions of a genuine one to obtain access to sensitive data. A genuine memory can be recognized using an authentication algorithm. A host may either communicate with the tag without authenticating it or after employing an authentication algorithm. The former case occurs when the host trusts the memory, manufacturer or vendor, etc. In the latter case, the host's security architecture will only allow it to communicate with memories that are flagged as authenticated i.e. a bit in the memory's master area set to one. Otherwise, no communication or operation will be successful before the memory is authenticated.

The security architecture implemented in this project employs the Lamport signature as the means for authentication as mentioned in section 5.4.1. Lamport signature is an inefficient asymmetric signature scheme in the sense that every signing key should be used only once and public key and signatures are big in size.

One initial signature is stored in the tag when manufactured. The signature provided is a Lamport one-time signature, since the private-key operation can be performed by the wireless tag. Tag's access control architecture ascertains that the private key is used only once and erased after use. In order to enable several hosts to certify the tag, several keys should exist on the tag.

If the wireless memory fails to complete the whole challenge response in the Lamport signature, it will not be flagged as authenticated. Therefore no communication is allowed with the memory. The details of the process are explained in section 5.4.1.

To avoid eavesdropping threats, the transfer of private data (PINs) are done in an encrypted manner. No PIN can be sent to the wireless memory before being

encrypted. This results in the fact that listening to the communication channel by adversaries will only result in gaining information which is not understandable by the attacker. The encryption method adopted in this architecture uses XXTEA block cipher and a roll-back counter. A value from the roll-back counter may only be used once and should be updated before the next use. The XXTEA block cipher operation details can be found in section 2.2.1.

Sending of PINs to the memory include both using a PIN and storing a PIN to the list of PINs in PIN area. PINs sent to get access to protected areas of memory are encrypted with the roll-back counter only. However, to avoid attackers attempting to write PINs to PIN area (which they can later use to obtain access to protected segments of memory) two steps of encryption are done to the PIN:

- First an encryption with one of the few master PINs, which are only known by the authorized users.
- second Encryption with the roll-back counter.

These steps help prevent malicious attackers to store their own specific PINs in the card. Consequently, an adversary can neither overhear the PINs already in the PIN area while being transmitted to the memory, nor can it add PINs to the PIN area. As a result, PIN protected areas cannot be accessed by an attacker.

To avoid denial of service attacks by randomly changing PINs, a few bytes are also allocated to DoS protection which are a part of the resulting bytes from the aforementioned 2 step encryption done to the PIN being transferred.

To be able to move sensitive data to a specified receiver in a way that the data can be only comprehended by that specific receiver, a life-cycle model is adopted in the wireless memory's security. This life-cycle model protects the data in the memory from anyone other than the intended receiver.

A segment in memory operating in this life-cycle model will be only readable after being encrypted hence over-writes from potential attackers is are prevented. For the data to be readable by the receiver, the host has to first generate a key stream that it can regenerate later. At this stage the segment is not readable to protect the key stream from adversaries. Later, in the sender side, the sender writes the data to that specific segment and the security architecture implemented in the memory takes care of encrypting the data with the written key-stream. Therefore, the data in the segment can be only readable by the owner of the key-stream. Details of the operation of segments in this life-cycle can be found in section 5.4.6.

To implement the encryption for receiver but in a coarser version, another life-cycle model is designed named "tag-assisted shared key". This model allows different hosts to generate shared keys with each other to be used later for sharing data that should be understandable just by the two paired sides. The aforementioned shared

key is generated by the wireless memory using the two hosts' IDs and part of a key-stream that each host writes to the segment operating in this life-cycle model. Part of the key-stream that is used up for creating a pairwise key with a certain host cannot be read from the memory. Only the pair of hosts that were involved in creating that shared key can reconstruct their own individual key from the shared key. Using this key, sensitive data can be moved to and from a pair of host in encrypted manner. To mitigate eavesdropping threats on this data transfer, a host may choose to write several IDs to a given tag.

The security mechanism implemented using life-cycle models are associated with specific segments. The architecture should have a way to make it impossible, if needed, for hosts to change the operating model for a segment. First mechanism to provide this would be edit PINs which can be allocated to all segments and thus protect the segment's management area from unauthorized changes. Next step protection should be applied in case even users knowing the PINs should not be allowed to change the segment's operation settings. This is done using a bit in each segment's management area (not editable bit) explained in section 5.3.3. Setting this bit will make a segment with permanent configuration in management area hence no change can be done to that segment.

6.2 Implementation

To achieve an overview of the resulting architecture for the security of the wireless memory, some measurements from the final design of the architecture, sample codes and tests cases are introduced in this section.

6.2.1 Measurements

Some essential criteria that should be considered in the overall design of the wireless memory's security architecture would be:

- the timing of the different operations
- the size (area on the FPGA board) of the design
- and the amount of memory needed for the code

The three factors mentioned above are monitored using Quartus II and Nios II tools and the results are as follows:

Timing Here the performance of the designed architecture is studied in the context of the number of clock cycles each operation takes to be completed. Total execution time of these functions depends on the system's operating frequency. The frequency of the Nios II processor's clock is set to 100 MHz during these measurement.

Table 6.1 shows some of the employed operations in the system and number of clock cycles it takes them to be executed completely.

Operation	Number of clock cycles
Authentication	5,498,172
Shared Key	7,970,780
Counter Update	18,198
Read PIN	1,676,095
Write PIN	1,631,559
EDIT PIN	1,662,211
Name	1,368,767

Table 6.1. *Execution clock cycles*

Area on FPGA Looking at Quartus II synthesis results, number of the logic elements utilized on the used FPGA board can be estimated to be 8246 elements. This number can vary when using different families of FPGA boards. In this synthesis Cyclon II EP2C70F672C6 is used. More detailed numbers on the used elements on the FPGA can be found in table 6.2

Element	Number
Total combinational functions	6,141
Dedicated logic registers	5,673
Total logic elements	8,246
Total registers	5,673
Total pins	10
Embedded multiplier	4
Total PLLs	1

Table 6.2. *FPGA measurements*

Memory The code for the security architecture which is run on the FPGA's Nios II processor is stored in a RAM memory. The size of the final code used for the architecture is approximately 25 KBytes (25360 bytes). Table 6.3 shows the code size in different phases of implementation

6.2.2 Testing

A series of test codes were designed to validate the functionality of the architecture. Using a Linux machine as the host, the test codes imitate the possible requests by a host and verifies the responses from the wireless memory. Some of the test cases are introduced in this section.

Phase	Code size (in bytes)
Simple communication with memory	5,120
PIN & name protection added	8,232
Lamport signature added	11,980
Counter life-cycle model added	14,840
XXTEA added	15,512
XOR life-cycle model added	21,252
Shared key life-cycle model added	25,360

Table 6.3. *Code size in different steps of implementation*

Test case 1: Access control to a segment protected by write PIN

Steps included in this test case will validate the memory's access control capabilities. In these steps first the corresponding management area of a segment is configured to be protected with write PINs and readable without PINs. Then show how first write to such segment fails without the right PIN and after providing the right PIN, a successful write operation can be confirmed by reading the contents of the segment.

Step 1: The management data of the segment should be changed to readable and protected writes. To edit management area, the roll-back counter should be updated and the relating edit PIN for that segment should be transferred to the memory in an encrypted format.

```
// updating counter
address = 0x000022      // adress of the RBCNT in memory
indata = 0x0000000000000001
write_bytes

// The PA_REG
address = 0x001F80      // address of register used for PA\_REG
indata = 0x00000000     // the index to the edit PIN for the segment
write_bytes

// giving encrypted edit  PIN number 0
address = 0x001FD0      // address of register used for edit PINs
indata = 0x357a91e1f6378158acfd77e0059678ee
write_bytes

// changing the management data for second segment
address = 0x003020      // adress of the management data for the second segment
indata = 0xb0 // setting WR PIN bit & RD bit
write_bytes
```

Step 2: In this step, a write is done to the segment without providing the PIN and then read from the same address. It can be seen that the contents have not changed.

```
// Writing to PIN protected segment
address = 0x005000      // address of the second segment
indata = 0xaabbccddaabbccdd
write_bytes
```

```
// reading from segment
address = 0x005000
read_8_bytes
outdata = 0x0000000000000000     // the contents of the segment have not changed
```

Step 3: In the final step, the right write PIN is provided and then, the successful write is verified by reading the contents of the segment.

```
// updating counter
address = 0x000022
indata = 0x00dd000000000002
write_bytes

// The PA_REG
address = 0x001F80
indata = 0x00000004      // the index to the write PIN for the segment
write_bytes

// give encrypted write PIN for position 4
address = 0x001FE0       // address of register used for write PINs
indata = 0x9808b391341a7a2a83121944957169f5
write_bytes

// Writing to PIN protected segment
address = 0x005000       // address of the second segment
indata = 0xaabbccddaabbccdd
write_bytes

// reading from segment
address = 0x005000
read_8_bytes
outdata = 0xaabbccddaabbccdd
// the contents of the segment have changed after providing the right PIN
```

Test case 2: Encryption for receiver model

This test case aims to verify the operation of segments in a specific life-cycle model: encryption for receiver model. To achieve this goal, first a segment is configured to operate in this model, then the segment is tested during the different states of its life-cycle.

Step 1: The management data of the segment should be changed to operate in the encryption for receiver model. To edit management area, the roll-back counter should be updated and the relating edit PIN for that segment should be transferred to the memory in an encrypted format.

```
// updating counter
address = 0x000022       // adress of the RBCNT in memory
indata = 0x0000000000000003
write_bytes

// The PA_REG
address = 0x001F80       // address of register used for PA\_REG
indata = 0x00000000      // the index to the edit PIN for the segment
```

```
write_bytes

// giving encrypted edit  PIN number 0
address = 0x001FD0      // address of register used for edit PINs
indata = 0x357a91e1f6378158acfd77e0059678ee
write_bytes

// changing the management data for second segment
address = 0x003020      // adress of the management data for the second segment
indata = 0x210004       // setting model to 4
write_bytes
```

Step 2: In this step, a key-stream is written to the segment, and in this stage the segment should not be readable. Then some bytes of data are written to the segment which will be XORed before being written to the segment by the tag.

```
// writing 2*32B to the segment, WO1 stage
// 0x008000 = 32768
for((i=32768;i<=32800;i+=32))   // writing 2*32 bytes beginning from address 0x8000
do
 indata = 0x11111111111111111111111111111111111111111111111111111111111111111111
 write_bytes

// reading from the segment
address = 0x8000
read_32_bytes
outdata = 0x0000000000000000000000000000000000000000000000000000000000000000
// reading failed, only zeros were returned as expected

// writing 2*32B in the segment, WO2 stage
// 008000 = 32768
for((i=32768;i<=32832;i+=32))   // writing 2*32 bytes beginning from address 0x8000
do
 indata = 0x3333333333333333333333333333333333333333333333333333333333333333
 write_bytes
```

Step 3: After this stage the segment can be read and checked to validate if the XOR operation has been done successfully.

```
// Reading the XORed segment, RM stage
// 008000 = 32768
for((i=32768;i<=32832;i+=32))
do
 read_32_bytes
outdata = 0x2222222222222222222222222222222222222222222222222222222222222222
outdata = 0x2222222222222222222222222222222222222222222222222222222222222222
```

Test case 3: Authentication

The target of this test case is to verify if the authentication algorithm implemented on the tag functions as expected. This test attempts to communicate with the tag before and during the process and thus validates that these attempts fail. After completing the authentication process, the tag is again tested to prove that it works normally then.

Step 1: To make sure that the tag is not authenticated before, the authentication flag should be cleared. This operation will not be possible in the final security architecture, however during testing, this option is available to enable multiple runs of the test.

```
/* Writing 1 to address 0x20 to make the tag authenticated
This option will not be possible in the real version
and is only available for testing reasons */
address=000020
indata=01
write_bytes
```

Step 2: Now two sets of private keys are required in the memory to be used in the Lamport signing operation. This step is also done only for testing purposes.

```
// address
#01B000 == 110592
// writing dummy data to two of the segments
// to be used as the two sets of private keys
for((i=110592;i<=110720;i+=32))
do
indata=1223344556677889900112233445566778899001122334455667788990 01AABB
write_bytes
done
```

Step 3: Now before authenticating the tag, an attempt to communicate with the tag is done. This attempt fails as the tag is still not ensured to be genuine.

```
// reading from a random segment
address=002000
read_8_bytes
outdata = 0x0000000000000000
```

It can be seen that all zeros were returned which shows that the read attempt has been unsuccessful.

Step 4: In this step challenges are written to the tag and the corresponding responses are read which will result in the tag to be authenticated.

```
// writing challenges (this step is repeated 256 times)
address=001F90
indata=00
write_bytes

// Reading responses (this step is repeated 256 times)
address=001F90
read_32_bytes
```

Step 5: In this step the same segment that was read before authentication is read and the real contents of the segment should be shown as the authentication is done successfully.

```
// reading from the random segment
address=002000
```

```
read_8_bytes
outdata = 0x1122334411223344
```

Step 6: Now the segments containing the Lamport keys are read to check if they are erased as they are supposed to.

```
address=01B000
read_32_bytes
outdata = 0x00000000000000000000000000000000000000000000000000000000000000000000000000000000
```

It can be seen that all zeros are returned showing that the segment does not contain the private keys any more.

7. CONCLUSION

In this thesis, some mechanisms to implement access control, authentication and secure life-cycle management in the tag side of a wireless memory system were developed and analysed. Furthermore, algorithms to ensure confidentiality and integrity of the wireless channel between the host and the tag were introduced.

The security architecture is built up on the fact that the memory is divided into different segments, each segment operating in a certain manner. Each segment is configured using that segment's corresponding management area. Setting some bits in segment's management area can define the segment as access controlled, separately for "read"s and "write"s. Such segments would require a 16 byte PIN to allow the host to do changes to the segment's data or read the information. To ensure further security, modifying the management area of a segment can be optionally also protected by PINs.

In addition to the PINs, each segment has an associated 16 byte name which is stored in the segment's management area. This name can be used for file-system data management or act as an additional PIN for the segment. As a result each access controlled segment can be protected by three different PINs.

A few number of owner (master) PINs are also stored in the memory which are predefined and cannot be changed. The main purpose of these PINs is to support backups or emergency recovery of data. However, they can also be used for sharing the tag between family members, employees of a company, and other similar purposes.

Another extra protection added is the use of an option which, if enabled, will make any further changes impossible to a segment and its management area. This will be a permanent condition for a segment and locks it in a certain mode. This option can be useful in some specific applications where no changes by the user should be allowed and the security may be endangered by changing the operation of a part of memory.

Operations which involve moving PINs (using PINs and transferring PINs) are protected against attacks in the employed architecture. This was implemented using block cipher encryption function XXTEA which has key length of 16 bytes and operates on 16 byte data blocks. XXTEA protects the transmission of keys against a passive attacker that listens to the communication over the air interface between

the reader and the wireless memory tag, but does not insert his own messages into that interface. When encrypting the PINs, a few bytes are also added to avoid denial of service attacks by randomly changing PINs.

Additionally, the master host may choose to encrypt some of the data before it stores it on the wireless memory. This encryption is performed in the master host using the AES block cipher.

It is required that before the first initialization by the "owner" of the tag, the host checks that the wireless memory tag has been certified by its manufacturer. This check, if successful, is a kind of guarantee that the tag behaves as specified.

A challenge-response technique, which is commonly used in authentication protocols, on the air interface was employed in tag's security architecture. This technique will guarantee the genuineness of the tag before allowing any communication with it. The employed mechanism is the Lamport signature. Once the authentication process is done, the tag will be marked as authenticated and further communication with the tag is allowed.

Some other security functions are implemented as life-cycles for different segments. These life-cycles include : 1. write once, read many 2. counter 3. encryption for receiver 4. tag-assisted shared key generation. These functions provide more specific secured transfer of data and can be enabled using a segment's management area.

During this project, the described security architecture was implemented and demonstrated. The correct and safe behaviour of the wireless tag has been ensured through intensive testing of the tag operations.

The measurements from the number of clock cycles taken by each of the security operations show promising results in the sense that it can be seen that these operations can be done within an acceptable time and with low latency. Additionally, the amount of memory used by the security software is no more than 25 Kbytes which is an extremely low amount in comparison with the available space.

The access control methods implemented in this project provide partial protection for the data stored in the wireless memory tag in the case that the tag is lost or stolen. Other protective measures include making a backup copy of the data, and tamper-proofing the tag: an attempt to open the cover will destruct the data. These additional measures are not specified further in this project.

The aforementioned features guarantee the secure life-cycle of the wireless tag and can be employed using the limited processing available in the tag.

In the future, when more processing capabilities in the wireless memory are available, the wireless memory security features may be expanded. The protocols developed in this project will remain essentially the same when the block cipher component is upgraded to a more sophisticated one (like AES) in the future. The

advantage of the more sophisticated block cipher is that it will better protect the key transmission against an active attacker.

Three patents have been filed from the wireless memory system project at Nokia Research Center. Also, the architecture is used as input material in standardization (Jedec TG649_1 Wireless Memory Standard).

REFERENCES

[1] Corrected block tea (XXTEA)[www]. Available at: `http://en.wikipedia.org/wiki/XXTEA`.

[2] M. Bellare and P. Rogaway. Introduction to modern cryptography. May 2005, Department of Computer Science and Engineering, University of California at San Diego. Lecture notes. 283 p.

[3] H. Bidgoli. *Handbook of information security, Threats, Vulnerabilities, prevention, detection, and management, Vol. 3.* John Wiley & Sons, Inc., California State University, 2006.

[4] Altera corporation. Cyclone II device handbook, vol. 1, San Jose, CA, February 2007. 470 p.

[5] Altera Corporation. Nios II processor reference handbook, San Jose, CA, May 2011. 284 p.

[6] Intel Corporation. Ultra-wideband (UWB technology), enabling high-speed wireless personal area networks (white paper), 2004.

[7] Renesas Electronics Corporation. Serial peripheral interface (SPI) & inter-IC (IC2) (SPI_I2C). September 2003.

[8] PennWell Corporation[WWW]. Solid-state technology. Available at: `http://www.electroiq.com/articles/sst/2012/04/jedec-non-volatile-wireless-memory-subcommittee-nokia-micron-\samsung-semiconductor.html`.

[9] M. Domenico, A. Calandriello, G. Calandriello, and A. Lioy. Dependability in wireless networks: Can we rely on wifi? *Security & Privacy, IEEE Vol. 5 , no. 1, pp. 23-29*, January-February. 2007.

[10] Toshiba America electronic components Inc. Nand vs. nor flash memory technology overview. Irvine, CA, 2006. Available at: http://umcs.maine.edu/ cmeadow/courses/cos335/Toshiba

[11] J. Foerster, E. Green, S. Somayazulu, and D. Leeper. Ultra-wideband technology for short- or medium-range wireless communications. *Intel Technology Journal, Vol. 5, pp. 1-11*, May 2001.

[12] H. Guo and B. Jin. Forensic analysis of skimming devices for credit fraud detection. *Information and Financial Engineering (ICIFE), Chongqing, 17-19 September 2010. pp. 542-546.*

[13] A. Gurtov. *Host identity protocol(HIP): towards the secure mobile internet.* John Wiley & Sons Ltd, Helsinki Institute for Information Technology, 2008.

[14] B. Guttman and E. Roback. *An Introduction to Computer Security: The NIST Handbook.* U.S. DEPARTMENT OF COMMERCE, Technology Administration National Institute of Standards and Technology, 1995.

[15] Network World Inc. Security's inseparable couple, February 2005.

[16] TecSec Incorporated. Information security and cryptography, Herndon, VA, November 2008. 10 p.

[17] Texas Instruments Incorporated. Universal asynchronous receiver/transmitter (UART) user's guid. Dallas, Texas, November 2010.

[18] I. Jantunen, J. Hamalainen, T. Korhonen, H. Kaaja, J. Jantunen, and S. Boldyrev. System architecture for mobile-phone-readable RF memory tags. *UBICOMM 2010, The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies , Florence, October 25, 2010. pp. 310-316.*

[19] I. Jantunen, J. Jantunen, H. Kaaja, S. Boldyrev, L. Wang, and J. Hämäläinen. System architecture for high-speed close-proximity low-power RF memory tags and wireless internet access. *International Journal On Advances in Telecommunicationsnternational, Vol. 4, no. 3&4*, April 2011.

[20] J. Jantunen, S. Boldyrev, M. Voutilainen, J. Arponen, and E. Confalonieri. Dual-interface memory for RF memory tag systems. *Memory Workshop (IMW), 2012 4th IEEE International, Milan, 20-23 May 2012. pp. 1-4.*

[21] J. Jantunen, H. Kaajaa, and S. Boldyrev. Super-regenerative architecture for UWB pulse detection: From theory to RF front-end design. *IEEE Transactions on circuits and systems I: Regular papers, Vol.56, no. 7, pp. 1500-1512*, July 2009.

[22] J. Jantunen, A. Lappetelainen, J. Arponen, A. Parssinen, M. Pelissier, B. Gomez, and J. Keignart. A new symmetric transceiver architecture for pulsed short-range communication. *IEEE Global Telecommunications Conference, New Orleans, LO, November 30 2008-December 4 2008. pp. 1-5.*

[23] D. Kim and M. Solomon. *Fundamentals of Information Systems Security.* Jones & Bartlett Learning, United States of America, 2010.

[24] J. Menezes, P. C. Oorschot, and S. A. Vanstone. *Handbook Of applied cryptography.* CRC-Press, Massachusetts Institute of Technology, 1996.

[25] R. Micheloni, G. Campardo, and P. Olivo. *Memories in wireless systems.* Springer, Italy, 2008.

[26] F. X. Moncunill-Geniz, P. Pala-Schonwalder, J. Bonet-Dalmau, F. del Aguila-Lopez, and R. Giralt-Mas. *Ultra Wideband Impulse Radio Superregenerative Reception, Ultra Wideband Communications: Novel Trends - System, Architecture and Implementation.* InTech, July 2011.

[27] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou. A survey on jamming attacks and countermeasures in wsns. *Communications Surveys & Tutorials, IEEE Vol. 11, no. 4, pp. 42-56*, Fourth Quarter 2009.

[28] C. Paar. Applied cryptography and data security. January 2005, Ruhr-Universitat Bochum. Lecture notes. 198 p.

[29] C. Paar and J. Pelzl. *Understanding Cryptography, A textbook for students and practitioners.* Springer, Bochum, 2009.

[30] M. Pavlin. Encryption using low cost microcontrollers.

[31] C. M. Pinteaa and P. C. Pop. Sensitive ants for denial jamming attack on wireless sensor network. *International Joint Conference SOCO13-CISIS13-ICEUTE13 Advances in Intelligent Systems and Computing Vol. 239, pp. 409-418*, September 2012.

[32] M. K. Qureshi, V. Srinivasan, and J. A. Rivers. Scalable high performance main memory system using phase-change memory technology. In *Proceedings of the International Symposium on Computer Architecture (ISCA), Austin, Texas*, 2009.

[33] A. Salomaa. *Public-key cryptography, 2nd edition.* Springer, Turku, 1996.

[34] CERT Coordination Center Software Engineering Institute. Denial of service attacks. June 2001, Carnegie Mellon University. Available at: `http://www.cert.org/tech_tips/denial_of_service.html`.

[35] W. Stallings. *Cryptography and network security, Principles and practice, fifth edition.* Pearsons Education, Inc., 2011.

[36] A. Stango and D. M. Kyriazanos. A threat analysis methodology for security evaluation and enhancement planning. In *Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems and Technologies*, Washington, DC, USA, June 2009. IEEE Computer Society.

[37] A. Tal. Two technologies compared: NOR vs. NAND (white paper). *M-Systems Flash Disk Pioneers Ltd.*, July 2003.

[38] H. V. Tilborg and S. Jajodia, editors. *Encyclopedia of cryptography and security, Volume 2.* Springer, Eindhoven University of Technology.

[39] T.Kasper, D. Oswald, and C. Paar. Wireless security threats: Eavesdropping and detecting of active RFIDs and remote controls in the wild. *Software, Telecommunications and Computer Networks (SoftCOM), Split, 15-17 September 2011. pp. 1-6.*

[40] JEDEC[WWW]. Non-volatile wireless memory. Available at: `http://www.jedec.org/news/pressreleases/jedec-announces-plans-standardize-non-volatile-wireless-memory`.

[41] L. Yang and G. B. Giannakis. Ultra-wideband communications, an idea whose time has come. *Signal Processing Magazine, IEEE, Vol. 21, no. 6, pp. 26-54*, November 2004.

[42] E. Yarrkov. Cryptanalysis of xxtea. Cryptology ePrint Archive, Report 2010/254, May 2010. Available at: http://eprint.iacr.org/2010/254.

[43] M. V. Zelkowits, editor. *Advances in computer science, Volume 74: Software developments.* Elsevier Inc., University of Maryland, 2008.

A. APENDICES

A.1 XXTEA Reference Code

The XXTEA algorithm used as the encryption function in this thesis is shown below[1].

```
#include <stdint.h>
#define DELTA 0x9e3779b9
#define MX (((z>>5^y<<2)+(y>>3^z<<4))^((sum^y) + (key[(p&3)^e]^z)))

void btea(uint32_t *v, int n, uint32_t const key[4]) {
  uint32_t y, z, sum;
  unsigned p, rounds, e;
  if (n > 1) {          /* Coding Part */
    rounds = 6 + 52/n;
    sum = 0;
    z = v[n-1];
    do {
      sum += DELTA;
      e = (sum >> 2) & 3;
      for (p=0; p<n-1; p++) {
        y = v[p+1];
        z = v[p] += MX;
      }
      y = v[0];
      z = v[n-1] += MX;
    } while (--rounds);
  }
```

A.2 Code samples

A.2.1 Authentication

In this section parts of the code related to the authentication process is presented. This represent only one challenge-responses out of the 256 challenges and responses in the actual process.

```
// **** receiving the challenge for one 32 Byte out of 256 32 Bytes ****
if(((data[4]) & (0x01)) == 0 )
{
// select from the first set and send back to base band
        R_CMDBUF[0] = M_READ;
        R_CMDBUF[1] = (unsigned char)((((KEY_SET1_ADDR)) >> 16) & 0xFF);
        R_CMDBUF[2] = (unsigned char)((((KEY_SET1_ADDR)) >> 8) & 0xFF);
        R_CMDBUF[3] = (unsigned char)((((KEY_SET1_ADDR)) >> 0) & 0xFF);

        send_to_FLASH(R_CMDBUF,36);
        read_from_FLASH(R_DATABUF,36);

        for(i=0 ; i<36 ; i++)
                R_KEY[i] = R_DATABUF[i];
        alt_printf("returned from first set\n");
}

if(((data[4]) & (0x01)) != 0 )
{
        // select from the second set and sending back to base-band
        R_CMDBUF[0] = M_READ;
        R_CMDBUF[1] = (unsigned char)((((KEY_SET2_ADDR)) >> 16) & 0xFF);
        R_CMDBUF[2] = (unsigned char)((((KEY_SET2_ADDR)) >> 8) & 0xFF);
        R_CMDBUF[3] = (unsigned char)((((KEY_SET2_ADDR)) >> 0) & 0xFF);
        send_to_FLASH(R_CMDBUF,36);
        read_from_FLASH(R_DATABUF,36);

        for(i=0 ; i<36 ; i++)
        R_KEY[i] = R_DATABUF[i];
        alt_printf("returned from second set\n");
}

// erasing used 32 Bytes from first key set
R_CMDBUF[0] = M_WRITE;
R_CMDBUF[1] = (unsigned char)((((KEY_SET1_ADDR)) >> 16) & 0xFF);
R_CMDBUF[2] = (unsigned char)((((KEY_SET1_ADDR)) >> 8) & 0xFF);
R_CMDBUF[3] = (unsigned char)((((KEY_SET1_ADDR)) >> 0) & 0xFF);
for(i = 4 ; i <36 ; i++)
        R_CMDBUF[i] = 0;
send_to_FLASH(R_CMDBUF,36);
read_from_FLASH(R_DATABUF,36);

// erasing the used 32 Bytes from second key set
R_CMDBUF[0] = M_WRITE;
R_CMDBUF[1] = (unsigned char)((((KEY_SET2_ADDR)) >> 16) & 0xFF);
R_CMDBUF[2] = (unsigned char)((((KEY_SET2_ADDR)) >> 8) & 0xFF);
R_CMDBUF[3] = (unsigned char)((((KEY_SET2_ADDR)) >> 0) & 0xFF);
for(i = 4 ; i <36 ; i++)
        R_CMDBUF[i] = 0;
send_to_FLASH(R_CMDBUF,36);
read_from_FLASH(R_DATABUF,36);
```

```
// **** receiving the response to challenge for one 32 Byte out of 256 32 Bytes ****
if((R_CODE == M_READ) && (R_ADDR == 0x001F90) && (authen_f == 0))
{
        send_to_BB(R_KEY,36);
}
write_authenticated:

        R_CMDBUF[0] = M_WRITE;
        R_CMDBUF[1] = 0x0;
        R_CMDBUF[2] = 0x0;
        R_CMDBUF[3] = 0x20;
        R_CMDBUF[4] = 0x01;

        send_to_FLASH(R_CMDBUF,5);
        read_from_FLASH(R_DATABUF,5);
```

A.2.2 Shared-key Generation Model

In this section parts of the tag-assisted shared key generation life-cycle model implementation is shown:

```
// starting froom the  second new ID the sharing proccess starts
if(ID_ADDR_PTR >= 0x002020)
{
        READER_NUM = ( ( ( ID_ADDR_PTR) - 0x002010 ) / 0x10 );

        for(k=0 ; k<READER_NUM ; k++){

        // Reading IDs of previous readers
                R_CMDBUF[0] = M_READ;
                R_CMDBUF[1] = (unsigned char)(((ID_ADDR_PTR) >> 16) & 0xFF);
                R_CMDBUF[2] = (unsigned char)(((ID_ADDR_PTR) >>  8) & 0xFF);
                R_CMDBUF[3] = (unsigned char)(((ID_ADDR_PTR) >>  0) & 0xFF) -
                (0x10 *(READER_NUM - k));
                send_to_FLASH(R_CMDBUF,20);
                read_from_FLASH(R_DATABUF,20);

                for(i=0; i<16 ; i++)
                        R_KEY[i+(k*48)] = R_KEY[i+(k*48)]^ R_DATABUF[i+4];

                // reading keys of the previous devices
                R_CMDBUF[0] = M_READ;
                R_CMDBUF[1] = (unsigned char)(((ID_ADDR_PTR1) >> 16) & 0xFF);
                R_CMDBUF[2] = (unsigned char)(((ID_ADDR_PTR1) >>  8) & 0xFF);
                R_CMDBUF[3] = (unsigned char)(((ID_ADDR_PTR1) >>  0) & 0xFF);
                send_to_FLASH(R_CMDBUF,52);
                read_from_FLASH(R_DATABUF,52);

                for(i=0 ; i<16 ; i++)
                        R_CMDBUF[i+4] = R_READER_ID[i]^R_DATABUF[i+4];
                for(i=16 ; i<32 ; i++)
                        R_CMDBUF[i+4] =
                        R_DATABUF[i+4]^R_DATABUF[i+20]^R_KEY[i+16+(k*48)];
                for(i=32 ; i<48 ; i++)
                        R_CMDBUF[i+4] = 0;

                // writing the shared key to the previous devices

                R_CMDBUF[0] = M_WRITE;
                R_CMDBUF[1] = (unsigned char)(((ID_ADDR_PTR1) >> 16) & 0xFF);
                R_CMDBUF[2] = (unsigned char)(((ID_ADDR_PTR1) >>  8) & 0xFF);
```

```
                R_CMDBUF[3] = (unsigned char)(((ID_ADDR_PTR1) >>  0) & 0xFF);
                send_to_FLASH(R_CMDBUF,68);
                read_from_FLASH(R_CMDBUF,68);

                for(i=16; i<32 ; i++)
                        R_KEY[i+(k*48)] =
                        R_KEY[i+(k*48)]^R_KEY[i+16+(k*48)]^R_DATABUF[i+20];
                for(i=32; i<48 ; i++)
                        R_KEY[i+(k*48)] = 0;
        }

        // writing the keys

        R_CMDBUF[0] = M_WRITE;
        R_CMDBUF[1] = (unsigned char)(((ID_ADDR_PTR1) >> 16) & 0xFF);
        R_CMDBUF[2] = (unsigned char)(((ID_ADDR_PTR1) >>  8) & 0xFF);
        R_CMDBUF[3] = (unsigned char)(((ID_ADDR_PTR1) >>  0) & 0xFF);

        for(i=0; i<48 ; i++)
                R_CMDBUF[i+4] = R_KEY[i];
        for(i=48 ; i<64 ; i++)
                R_CMDBUF[i+4] = 0;
        for(i=48; i<96 ; i++)
                R_CMDBUF[i+20] = R_KEY[i];
        for(i=112 ; i<128 ; i++)
                R_CMDBUF[i+4] = 0;

        send_to_FLASH(R_CMDBUF,132);
        read_from_FLASH(R_DATABUF,132);

        R_CMDBUF[0] = M_WRITE;
        R_CMDBUF[1] = (unsigned char)(((ID_ADDR_PTR1) >> 16) & 0xFF);
        R_CMDBUF[2] = (unsigned char)(((ID_ADDR_PTR1) >>  8) & 0xFF);
        R_CMDBUF[3] = (unsigned char)(((ID_ADDR_PTR1) >>  0) & 0xFF);

        for(i=0; i<48 ; i++)
                R_CMDBUF[i+4] = R_KEY[i+96];
        for(i=48 ; i<64 ; i++)
                R_CMDBUF[i+4] = 0;
        for(i=48; i<96 ; i++)
                R_CMDBUF[i+20] = R_KEY[i+96];
        for(i=112 ; i<128 ; i++)
                R_CMDBUF[i+4] = 0;

        send_to_FLASH(R_CMDBUF,132);
        read_from_FLASH(R_DATABUF,132);
}
```