MUTUAL AUTHENTICATION SCHEME FOR MOBILE RFID SYSTEMS

by

Hisham Khalaf Allahem

Submitted in partial fulfilment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
March 2013

DALHOUSIE UNIVERSITY

Faculty of Computer Science

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "MUTUAL AUTHENTICATION SCHEME FOR MOBILE RFID SYSTEMS" by Hisham Khalaf Allahem in partial fulfilment of the requirements for the degree of Master of Computer Science.

Dated: March 28, 2013

Supervisor: _____

Readers: _____

_____

DALHOUSIE UNIVERSITY

DATE: March 28, 2013

AUTHOR: Hisham Khalaf Allahem

TITLE: MUTUAL AUTHENTICATION SCHEME FOR MOBILE RFID SYSTEMS

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: MCSc. CONVOCATION: May YEAR: 2013

_____

Signature of Author

Dedication Page

This work is dedicated to my father and mother for all the sacrifices they offered for me to be successful.

Table of Contents

List of Tables

List of Figures

Abstract

Radio Frequency Identification (RFID) systems are rapidly becoming popular in a variety of applications such as supply chain management, storage management and healthcare. Such a system consists of a tag with a unique identifier, a tag reader and a backend server. Due to the system's limited computational resources, it can be subject to various types of attacks. This can exacerbate when the reader itself is mobile. The objective of this thesis is to propose a mutual authentication scheme for mobile RFID systems. Our proposed scheme uses a shared encryption key generated and updated by the mobile reader to authenticate the system entities. The encryption keys are updated at the end of each authentication session. Experimental results show that the proposed scheme meets the security and privacy goals, and resists known attacks on mobile RFID systems.

List of Abbreviations and Symbols Used

Abbreviations

RFID	Radio Frequency Identifier
CRC	Cyclic Redundancy Code
OR	Logical or operation
AND	Logical and operation
XOR	Logical exclusive or operation
DoS	Denial of Serves
MITM	Man in The Middle
PRNG	Pseudorandom Number Generator
ID	Identification Number
SD	System Dependent
Desynch.	Desynchronization
Eavesdr.	Eavesdropping

Symbols	**Meaning**
k	Encryption Key
ck	Current Encryption Key
nk	New Encryption Key
ok	Old Encryption Key
$Cref$	Current Tag Reference Number
$Oref$	Old Tag Reference Number
$Nref$	New Tag Reference Number
$Trnd$	The Tag Pseudorandom Number
$Rrnd$	The Reader Pseudorandom Number
$Srnd$	The Server Pseudorandom Number
M	The Message to be Encrypted

EM	Encrypted Message
RID	The Mobile Reader ID
SID	The Server ID
$E_k(M)$	The Encryption Hash Function to Encrypt M using key k
$D_k(EM)$	The Decryption Hash Function to decrypt EM using key k

Acknowledgement

I would like to acknowledge my supervisor Dr. Srinivas Sampalli for his guidance and recommendations for me to complete this work. I would like also to acknowledge Dr. N. Zincir-Heywood and Dr. Q. Ye for their feedback to improve my thesis. Finally, I would like to acknowledge my father, my mother, my teachers, the research team of Dr. Srinivas Sampalli, and my friends for all their support.

Chapter 1: Introduction

1.1 Overview

Radio Frequency Identification (RFID) is an important emerging wireless technology. It uses a tag with microchip that contains a unique identifier used to identify or trace the tag. The cheap components and ease of set up features have increased the spread of using RFID systems in many areas such as supply chain management and control, health care, and point of sales. RFID is also a major component of the new emerging technology know as “Internet of things”. Moreover, RFID chips have been integrated in the new smartphones [9]. It is stated that RFID is the future trend of ubiquitous technology [6][7][10]. As with any wireless network, RFID systems are vulnerable to different types of attacks such as DoS, MITM, eavesdropping, or tracking attacks. Therefore, security and privacy are two important considerations for RFID systems. This demand becomes a challenge to researchers because of the limited computational resources of RFID systems.

1.1.1 RFID System Structure

RFID systems consist of four main components, namely, tag, reader, middleware, and backend server. Figure 1.1 shows RFID system components. In general, the reader sends a request to the tag that sends data back to the reader. The data consists of a unique identifier called the electronic product code (EPC). In addition, other information such as authentication data can also be sent. The reader then sends the data to the backend server through the middleware. The server then identifies the tag and executes the appropriate operations.

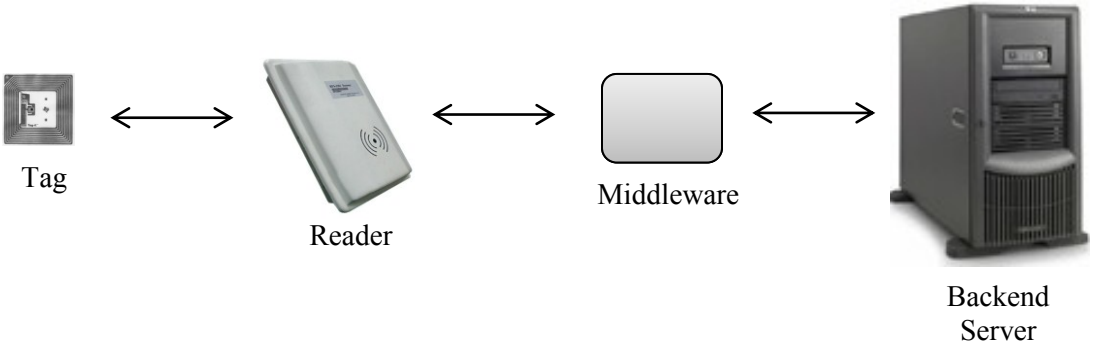

Figure 1.1 RFID system components

1.1.2 Categories of RFID Tags

An RFID tag is a device with a microchip that can store a unique identification number that identifies it. Tags based on battery power are classified into three categories: passive tags, semi-passive tags, and active tags [1].

A passive tag does not have a power source but is powered by the reader. When the reader wants to communicate with a passive tag, it sends an electromagnetic signal that powers up the tag. The tag then responds with the data to the reader. On the other hand, the active tag has a built-in power source and can communicate with the reader on its own. Finally, the semi-passive tag has a built-in power source to perform computations but needs the reader for communication.

Since it does not have a built-in power source that needs to be replaced after a while, the lifetime on the passive tags is unlimited. It is fast to read from and to due to the simple operations that it is capable of performing. Moreover, since it does not require a built-in power source to operate, it is cost-effective and can be deployed in large RFID systems. Active and semi-passive tags, however, do not have a long lifetime since the lifetime is limited by their power source. For that reason, are more expensive to be manufactured and are deployed only in small RFID

systems [11].

RFID tags can also be categorized based on their computational power into four classes [12]. The first class is the full-fledged class, which can perform complex cryptographic functions such as symmetric encryption or the public key algorithms. The second class is the simple class, which is capable of generating random numbers and one-way hashing. The third class is lightweight class, which can generate random numbers and simple functions like Cyclic Redundancy Code (CRC) checksum but cannot perform hashing. The last class is the ultra-lightweight class, which is the simplest class that can perform simple bit-wise operations such as: XOR, AND, and OR [22].

1.1.3 Categories of RFID Readers

RFID readers can read tags different ranges, from a few millimeters up to several meters [4]. It can also have the capability to query millions of tags in a very short time. RFID readers can be categorized into fixed readers and mobile readers [13]. Fixed readers, as the name illustrates, are placed and fixed in one place. Usually it communicates to the RFID backend server on a wireline network. On the other hand, mobile readers are not fixed and are usually handheld. They communicate with the backend server over a wireless network [13][9].

1.1.4 Security Challenges of RFID Systems

RFID systems are vulnerable to different types of attacks such as DoS, cloning, desynchronization, replay, tracking or eavesdropping attacks [2][8][23]. Due to the characteristics of RFID systems such as the limited computational power of the tag or the wireless communication between RFID components, these attacks can be easily launched. For example, an attacker can launch a desynchronization attack by simply block the communication between RFID system's entities on a

certain time to prevent information from updating. The wireless environment RFID uses to communicate makes it easier for the attacker to launch the attack. Another example is the cloning attack where the tag's information is copied bit-by-bit to another tag. The original tag cannot prevent its information from being physically copied and the cloned tag can communicate with the RFID system as a valid tag. Reader and backend server cannot differentiate between valid and cloned tag [14].

Many authentication protocols were proposed to achieve the security and privacy goals. Most of these protocols used a compensation of symmetric encryption keys generating and updating. Others have used additional techniques such as permutation [8], timestamp [10], or a third party to do the authentication [6]. These solutions managed to achieve some of the goals, however, none of them have proposed a solution to prevent or recover from desynchronization attack.

1.1.5 Objective of the Thesis

The objective of this thesis is to implement a new scheme for a mutual authentication for mobile RFID systems. The new scheme uses a shared encryption key generated and updated by the mobile reader to authenticate the system entities. The encryption keys are updated at the end of each authentication session. The goal of this scheme is to achieve security and privacy for RFID systems. Our work aims to achieve all the goals including a recovery mechanism from desynchronization attack.

1.2 Report Organization

The rest of the thesis is organized as follows. Chapter 2 discuses the background and the related works. Chapter 3 presents the focus and motivation of this research. Chapter 4 describes the proposed scheme. Chapter 5 presents the evaluation methodology and experimental results. Chapter 6 provides concluding remarks and scope for future work.

Chapter 2: Background and Related Work

2.1 Background

2.1.1 RFID Security

Security and privacy are two main issues arising with the use of RFID systems. The use of wireless communication between RFID tags and reader increases the need to protect the data from any attack during the authentication process. Moreover, when an RFID mobile reader is used, the channel between the reader and the backend server also uses wireless communications. This adds more security and privacy demands than the case where the channel between the reader and the backend server is a wired communication. Finally, as the RFID readers are mobile, they are being carried by the user when used, which increases the privacy concerns to the user's information [7][23].

Due to the low computational capabilities of RFID tags [8], complex cryptographic schemes cannot be applied when designing RFID authentication systems. One of the solutions to this issue is the use of ultra-lightweight authentication protocols that do not require a heavy computation [1]. Such solutions should guarantee forward security for RFID systems. That means an attacker at time t cannot trace the previous transactions at t', where $t > t'$ [9][14].

2.1.2 Attacks on RFID Systems

The nature of RFID systems makes them vulnerable to various types of attacks. The attacks are categorized into eight categories:

1. DoS Attack:

   In this attack, the attacker can send a large number of tags IDs to the reader [9]. The attacker can also flood the backend server with fakes requests, which may cause the server to crash [7].

2. Cloning Attack:

   The attacker reads the tag's information and copies it to another tag to impersonate the original tag. The backend server in this case cannot recognize the original tag from the fake one [14][9]. This attack is typically a physical layer attack and that makes it hard to prevent [15].

3. Desynchronization Attack:

   This attack can prevent the information from reaching the reader or the tag when the update information is sent from the backend server [9][16]. On the next session, the backend server cannot authenticate the reader or the tag or both. The challenge on this attack is that it does not need to decrypt the data in order to make a successful attack. It needs only to block, for example, the last session where the update is performed then the attack will be successful. This attack can lead to economical losses [21]. The proposed solutions so far cannot prevent the attack from being launched, so they detect the attack after it occurs and then applies a recovery approach to continue with the next session.

4. Replay Attack:

   The attacker can listen to the session and replay the query to the tag, the reader, or the backend server, resulting in the impersonating of the valid tag [14] [9].

5. Eavesdropping Attack:

   The attacker can listen to the communication channel between the tag, the reader, and the backend server to gather information [9].

6. Man-In-The-Middle Attack (MITM):

   In this attack the attacker starts to listen to the communication

between the tag, the reader, and the backend server, gather information, and finally manipulate it or redirect it [9].

7. Location Tracking Attack:

   The attacker tracks the location of the tag or the reader by listening and analyzing the communication between RFID system entities [14]. The challenge of this attack is that the attacker does not need to decrypt encrypted messages. He/she can analyze the encrypted messages to find a pattern that could be used to track the RFID entity.

In addition to mitigating the above attacks, a secure RFID system should also ensure forward and backward secrecy. This means that if an attacker manages to get the encryption key for the i^{th} session, then he/she should not be able to decrypt data from earlier sessions, and vice versa.

2.2 Related Work

The low computation capabilities RFID passive tags have made the RFID system in general vulnerable to the above-mentioned attacks. Because of that, complex authentication schemes are not applicable to RFID systems based on passive tags. Lightweight and ultra-lightweight authentication mechanisms, on the other hand, are more applicable in these systems. Moreover, the mobility of the reader adds more attention to the privacy issue since the reader is handheld.

The proposed protocols that use lightweight and ultra-lightweight authentication schemes apply a combination of symmetric key updating and synchronizing approach to solve the weakness of using lightweight and ultra-lightweight authentication schemes. We will review and analyze some of the proposed protocols that use lightweight and ultra-lightweight authentication schemes in their design.

The proposed solutions or protocols are based on two methods [14]:

1. Physical Methods: These solutions are based on changing the physical behavior of the tag so it does not respond anymore to any request from any RFID reader. These methods may include kill command, active jamming, or a blocker tag.
2. Cryptographic Methods: These approaches are based on using software to protect the tag from any possible attack. Examples of methods used in this approach are a hash function or pseudorandom number generator (PRNG).

We will review and analyze six proposed protocols that use the cryptographic methods to solve or prevent attacks on RFID systems with mobile readers and passive tags:

The first protocol was proposed by Sandhya and Rangaswamy [9]. The protocol is based on a hash function and a shared key between the reader and the backend server, and another key shared between the tags and the backend server. When the reader queries the tag, it generates a pseudorandom number *r* and sends it to the tag. The tag then computes a hash value with its ID and sends it to the reader. The reader then computes its own hash value that contains the reader's ID and sends it to backend server along with the value sent by the tag. The backend server then verifies both values from the reader and the tag. The backend server then updates the shared key with the tags to the new key, performs XOR operation of the new key with the data from the tag along with the detailed information of the tag, and sends it to the reader. The reader then obtains the tag's information, performs XOR operation to the data from the backend server along with *r,* and then forwards it to the tag. The tag finally verifies the reader, decrypts the data from the backend server, and then updates the encryption key.

According to the authors, the proposed protocol can prevent illegal access, eavesdropping, tracking, cloning and replay attacks. Moreover, this protocol cannot resist the desynchronization for it does not have any counter approach to recover for the attack. A tracking attack can also be done since the encryption key does not

update between the mobile reader and the backend server.

The second proposed protocol was designed by Chia-Hui, Min-Shiang, and Augustin [14]. It is a hash function based protocol for mutual authentication and data security between the tag and backend database. It uses a secret value *S*, random number *N*, and hash function *h()* as both static and dynamic lockers. When the mobile reader queries the tag, it sends a random number to the tag. The tag then generates its own random number, hashes it with the secret key that is shared with the backend server, and forwards it to the reader. The reader then hash its ID along with the random number it generated before and forwards the value with the data from the tag to the backend server. The backend server then verifies the reader using the old random number from the reader (if they match then the authentication fails). After that, the backend server verifies the tag, generates the new secret key by hashing the tag's ID along with its own the random numbers and the tag's random number, stores the old secret key, and send the hash value of the tag's ID and its random number. The backend server's random number is also sent to the tag as plain text. Finally, the tag verifies the data from the backend server and generates the new secret key the same way the backend server did.

The authors state that the proposed protocol is secure against tracking attack, cloning attack, replay attack, forward security and DoS attack. However, we can notice that the backend server does not authenticate the reader. Even though the reader sends a random number every time, it does not prevent a rogue reader from penetrating the backend server and act as a trusted reader. Once the attacker has the reader ID, he/she can easily communicate with the backend server by simply generating new random numbers and hash it with the reader ID. Moreover, the proposed protocol does not prevent or recover from the desynchronization attack. Finally, eavesdropping and MITM attacks are not applicable since the secret key is changing between the tag and the backend server.

The third protocol was proposed by Allen, Dwen-Ren, Chang-Lung, and Yong-Jiang [11]. It combines a mobile communication device (such as PDAs or Phones)

and an RFID credit card for a trading mechanism. When the reader starts the connection with the tag, it generates a random number, hashes it with the reader secret key, and queries the tag. The tag then verifies the reader, generates a random number and a timestamp, hashes it together with the tag ID and secret key, and sends it to the reader. The reader then forwards it along with values it generated previously. The backend server then verifies the data from the tag and the reader. The backend server then creates a random number along with a timestamp, generates the new tag key, sends the new key to the tag, and updates the key on the backend server. The tag then verifies the data and updates the new key.

The authors mention that the proposed protocol achieves both privacy and authentication for the user and the system in terms of tag tracking (by using random numbers and timestamps) and cloning but it does not resist desynchronization attacks. However, we can see that the reader does not update its secret key. This will allow the attacker to break the key by brute force since the used key is a lightweight.

The fourth protocol was designed by Sun and Ting called Gen2$^+$ [4]. It provides mutual authentication between the tags and the readers by randomizing each session. For authentication, Gen2$^+$ uses only PRNG and CRC-16 functions. This protocol aims to improve the original Gen2 protocol [5].

The authors state that their protocol is able to resist tracing and cloning attacks. The authors, however, do not analyze the protocol against other attacks.

The fifth protocol was proposed by Enjian, Huayong, Kejia, and Wen [6]. It is a key management protocol based on a Trust-Third-Party to secure the session between the tags, the readers, and the backend server. This protocol aims at the following: first, the reader communicates with backend server via an anonymous communication that provides security for the reader; second, it protects the user from being tracked by using the dynamic tag identity; third, it secures the information between the tag, the reader, and the backend server using the secured session key construction; finally, the privacy-policy-based access control

mechanism in the system guarantees the system's performance since the owner of the tag can previously specify what tag information the reader can and cannot access.

The proposed protocol, according to the authors, resists replay, eavesdropping, and tracking attacks. Forward security is also a feature on this protocol. However, there is no analysis done by the authors on DoS, desynchronization, cloning, and MITM attacks.

The sixth proposed protocol was designed by Lee and Yi [7]. This protocol uses the mobile device (i.e. smartphone) computation capabilities to perform heavy calculations and then transfers only the results to the tag. It uses the passive tags as an active tags and achieve the same security as in using active tags. The reader unlocks the tag at the beginning of the authentication and locks the tag after the last step of the authentication process.

The protocol based on the authors' analysis resists cloning, location tracking, and eavesdropping attacks. However, no analysis was conducted for the DoS, desynchronization, replay, and MITM attacks. Forward security is applied in this protocol.

In common, these protocols are using lightweight and ultra-lightweight passive tags to apply a symmetric key authentication mechanism where each entity has a private key. In addition, they assume the communication channel between the reader and the backend server is secure, so they consider these two entities as one entity.

2.3 Summary

In this chapter, we discussed the background of the mobile RFID security and what authentication schemes are best to use. We also categorized and defined the security threats and attacks on mobile RFID systems. Finally, we presented some proposed protocols that solve these issues with mobile RFID and analyzed each protocol's advantages and disadvantages .In the next chapter, we will present the motivation and the objectives of our work.

Chapter 3: Research Focus and Proposed Scheme

3.1 Motivation

Typically, in any RFID system, the goal is for the tag and backend server to communicate and exchange information with each other via the reader. In most of these systems, the communication between the tag and the reader is wireless. The reader then communicates and exchanges information with the backend server using either wired or wireless communication (in mobile RFID systems). Moreover, RFID systems' tags have limited computational capabilities. The information that is being exchanged between RFID entities are the encryption keys shared among them, their unique IDs, and any updating information (for example, new encryption key or acknowledgment).

This process should be conducted in a secure and private manner. Most of the current RFID systems, however, lack security and privacy in their design [3]. These RFID systems also consider the connection between the reader and the backend server to be secure so they design their protocols on the assumption that the backend server is the central entity. Therefore, the backend server is responsible for generating and updating the encryption keys to the tag and/or to the reader. Encryption keys in the authentication process can be used between the tags and the backend server, the reader and the backend server, or among all entities. In addition, information exchange on mobile RFID systems between the tag, the mobile reader, and the backend server is wireless. This increases the security and privacy concerns in two ways: firstly the information is exchanged on air in all stages which means they can be easily captured; secondly, the mobile reader is handheld which adds a privacy concerns to the private information like location (which is not an issue with fixed-reader RFID systems). We can conclude that any loss or modification of the exchanged information between RFID systems entities should be prevented [21].

3.2 Objectives

Motivated by the limitations of RFID systems that we mentioned previously, the objective of this thesis is to design a mutual authentication protocol between the tag, the reader, and the backend server, which focuses on:

1. Symmetric key system where all the RFID system's entities will have the same encryption key.
2. The reader is mobile and responsible for generating and updating encryption keys in parallel to both the tag and the backend server.
3. Encryption keys will be updated after each authentication session.
4. Encryption keys are randomly generated with no link to the previous encryption key.
5. Each entity will authenticate the other entity independently at each step of the authentication process.
6. Minimization of the desynchronization attack by using the mobile reader to update the encryption keys.
7. Fulfillment of the following security goals: confidentiality, data integrity, authentication, non-repudiation, and forward security.

3.3 Proposed Scheme

In this section, we discuss the proposed mutual authentication scheme between the tag, the mobile reader, and the backend server. The novelty of this work is that the mobile reader is the central entity that generates and updates the encryption keys for the tags and backend server. The entities will authenticate each other on every step of the authentication session. Finally, the mobile reader will update the encryption keys in parallel for both the tag and the backend server.

3.3.1 Assumptions

The following assumptions are important for the proposed scheme to be

successfully implemented:

- The system's owner or enterprise preloads RFID tags with the data. The data is a unique reference number to the tag ID so the backend server can match it with the actual tag ID.
- The tag has a pseudorandom number generator to generate a 32-bit random number.
- The reader is mobile and responsible of generating and updating the encryption keys since it is handheld.
- The tag, the mobile reader, and the backend server will have a preloaded 128-bit encryption key (k) that will be used to encrypt and decrypt messages during the mutual authentication.
- The mobile reader and the backend server will store the old reference number ($Oref$) and the old encryption key (ok) in order to continue with the mutual authentication if a desynchronization attack was launched on the previous mutual authentication session.

3.3.2 Overview of the Scheme

As we discussed in Chapter 1, an RFID system consists of a tag with a unique identifier to an object, a reader that transmits the data from the tag to the backend server, and a backend server that uses the tag ID to perform a task. In our proposed scheme, the mobile reader and the backend server store the same information related to each tag. The tag information on the mobile reader and backend server is represented by the following equation:

$$t_i = \{Cref_i, ck_i, Oref_i, ok_i, ID_i\} \quad (1)$$

where:

$Cref_i$: the current reference number of the i^{th} tag.

ck_i : the current encryption key for the i^{th} tag.

$Oref_i$: the old reference number of the i^{th} tag ID.

ok_i : the old encryption key for the i^{th} tag.

ID_i : the i^{th} tag's ID.

For a set of *n* tags each one represented by equation (1), the stored data on both the mobile reader and backend server can be represented as the following equation:

$$T = \{t_1, t_2, t_3, \ldots, t_n\} \quad (2)$$

The data on the tag will contain only two elements represented as the following equation:

$$t_i = \{Cref_i, ck_i\} \quad (3)$$

The normal process of the proposed scheme consists of three authentication phases. Figure 3.1 shows design summary of our proposed scheme phases. The phases are categorized as the following:

- Phase1: between the mobile reader and the tag, and consists of:
  - Phase1.1: the mobile reader communicates with the tag.
  - Phase1.2: the tag responds to the reader.
- Phase 2: between the mobile reader and the backend server, and consists of:
  - Phase 2.1: the mobile reader communicates with the backend server.
  - Phase 2.2: the backend server responds to the mobile reader.
- Phase 3: between the mobile reader and both the tag and backend server.

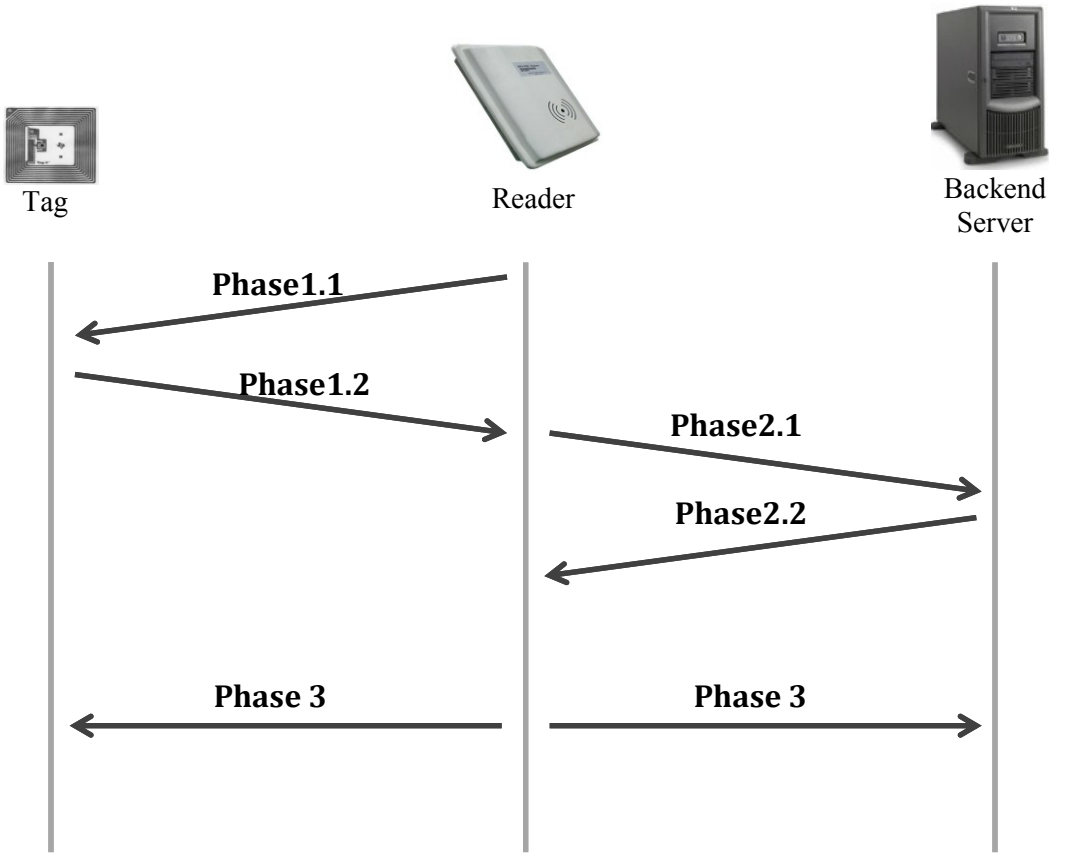

Figure 3.1 Design summary of the proposed scheme phases

The authentication process will be as follows: the mobile reader will query tag *i* with a hello message (Phase 1.1). The tag then generates a pseudorandom number $Trnd$, encrypts it with the current encryption key ck_i, and sends the encrypted message EM_i along with the tag current reference $Cref_i$ in plaintext: $EM_i;Cref_i$ (Phase 1.2). When the mobile reader receives EM_i, it gets the tag $Cref_i$, fetches its database for the correspond tag ID, gets the correspond current encryption key ck_i , and decrypts EM_i to get M_i and verify the tag. When the tag is verified, the mobile reader generates a pseudorandom number $Rrnd$, concatenates the retrieved message M_i from the tag with $Rrnd$ and the mobile reader ID: M_{inew} = ($Rrnd,RID,M_i$) where M_i contains "$Trnd$". The mobile reader then encrypts the concatenated message with ck_i to get EM_i and sends it along with $Cref_i$ to the

backend server: $EM, Cref_i$ (Phase 2.1).

After receiving the data from the mobile reader, the backend server retrieves $Cref_i$, fetches its database for the correspond tag ID, gets the correspond current encryption key ck_i , decrypts EM_i to get M_i and verify the mobile reader. Notice that the tag is automatically authenticated in this Phase for two reasons: firstly, the mobile reader will not contact the backend server unless it verifies the tag; and secondly the backend server cannot find the corresponding decryption key unless it exists on its database. After that, the backend server generates a pseudorandom number $Srnd$, concatenates the retrieved message M_i from the mobile reader with $Srnd$ and the backend server ID: M_{inew} = ($Srnd, SID, M_i$) where M_i contains $(Rrnd, RID, Trnd)$. The backend server then encrypts the concatenated message with ck_i to get EM_i and sends it along with $Cref_i$ to the mobile reader: $EM; Cref_i$ (Phase 2.2).

When the mobile reader receives EM_i from the backend server, it gets $Cref_i$, fetches its database for the correspond tag ID, gets the correspond current encryption key ck_i , and decrypts EM_i to get M_i and verify the tag and the backend server. Finally, the mobile reader generates a pseudorandom number $Rrnd$, the new encryption key nk_i key, and the new tag reference number $Nref_i$. The mobile reader then concatenates them together: ($Rrnd$, nk_i , $Nref_i$, RID) , encrypts them with ck_i to get EM_i, and sends EM_i to the tag and (EM_i, $Cref_i$) the backend server at the same time.

The mobile reader then updates the encryption keys and the tag's reference numbers on its database by replacing the old encryption key ok_i with the current encryption key ck_i and the current encryption key ck_i with the new encryption key nk_i . The old reference number $Oref_i$ will be replaced with the current reference number $Cref_i$ and the current reference number $Cref_i$ will be replaced with the new reference number $Nref_i$. For the backend server, the same process is done after verifying EM_i from the mobile reader. The following demonstrates the updating process on both the mobile reader and the backend server:

$ok_i = ck_i$

$ck_i = nk_i$

$Oref_i = Cref_i$

$Cref_i = Nref_i$

On the tag side, the tag will verify EM_i from the mobile reader and then updates its current encryption key ck_i with the new nk_i, and its current reference number $Cref_i$ to the new reference number $Nref_i$ (Phase3) . This ends the authentication process.

3.3.3 Encryption Keys and Reference Numbers Management

3.3.3.1 Encryption Mechanism

The proposed encryption scheme is based on a simple mechanism. The encryption key for the authentication session is a 128-bit key shared between the mobile RFID entities. The goal of the encryption mechanism is to guarantee the security and privacy of the mobile RFID entities from any attack or breach. This encryption mechanism is used among all the mobile RFID entities. It uses logical exclusive-OR operation (XOR) to perform the encryption mechanism. Table 3.1 illustrates the encryption mechanism. We can put the encryption mechanism on the following equation:

Encrypted Message (EM)=$E_{ki}(M) = M \oplus ki$ where: M is the message to encrypt (4)

Table 3.1 Illustration of the encryption mechanism

Algorithm	**Encryption**
Input:	Message (M) Encryption Key (k_i)
Output:	Encrypted Message (EM)
Process:	$EM = M \oplus ki$

3.3.3.2 New Encryption Key Generation

In the last phase of the authentication session (Phase3), the mobile reader is responsible for creating the new 128-bit encryption key (nk_i) and sending it to both the tag and the backend server. Table 3.2 shows the new encryption keys generation process.

The 128-bit encryption key was chosen to guarantee a strong encryption mechanism that can resist any attempt to break the encrypted message by any means such as brute force. Moreover, the 128-bit can be stored on the tag's limited memory. The 128-bit encryption key consists of 32 characters. Each character is randomly chosen from a pool that contains the small letters from a to f and numbers from 0 to 9. This process guarantees that the generated encryption key will be unique and random. The total number of possible keys is 32^{16}, which is hard to break by brute forcing for example.

Table 3.2 New encryption keys generating process

Algorithm	**New Encryption Key**
Input:	String Pool = "0123456789abcdefghijklmnopqrstuvwxyz"
Output:	New Encryption key

Process:	For char :=1 : 32 {
	Choose a random character form Pool
	Update the new encryption key with the chosen character
	}
	return the new encryption

3.3.3.3 Encryption Keys Updates and Management

The proposed scheme encryption mechanism manages the encryption keys as follows:

- The tag has one encryption key ki .
- The mobile reader and backend server both store the current and old encryption keys (ck_i and ok_i).

The mobile reader is responsible for generating the new encryption key. It is also responsible for updating the new encryption key for the tag and the backend server. After generating the new encryption key nk, the mobile reader encrypts nk with the session's current encryption key and sends it to the tag and the reader to update their encryption key/keys. The backend server mobile reader then updates its encryption keys. As explained on the earlier, both the tag and the backend server authenticate the last message from the mobile reader (Phase3). If the authentication is successful, then the tag will replace the current encryption key ck with the new encryption key nk. For the backend server, after a successful authentication of Phase3, it will replace the old encryption key ok with the current encryption key ck and the current encryption key ck with the new encryption key nk. The same process of encryption keys update on the backend server is performed on the mobile reader.

3.3.3.4 New Tag Reference number Generation and Management

The idea of the tag reference number is to protect the actual tag ID from the public to ensure the security privacy of the tag. The reference number is a six-digit 24-bit number points to the actual tag ID on the mobile reader and backend server databases. Using it, both the mobile reader and backend server can extract information about the tag to, for example, authenticate it or perform a specific task related to the tag. The reference number is sent in a plaintext to the mobile reader and the backend server. In order to prevent tracking attacks on the tag; the tag

reference number is always updated on all the mobile RFID entities. The new reference number (*Nref*) generated by the mobile reader is then sent encrypted to both the tag and backend server.

The new reference number (*Nref*) is generated by choosing a random number from 1 to 1000000. Each *Nref* is checked if it exists on the database of the mobile reader or not. If so, then a new *Nref* is generated, and so on. Table 3.3 shows this process of generating new reference number.

When the tag receives the last message for the mobile reader (Phase3) that contains the new reference number (*Nref*), it authenticates it, gets *Nref* , and replaces the current reference number (*Cref*) with *Nref* . The backend server will perform the same process of authenticating the last message from the mobile reader (Phase3), gets *Nref* , replaces the old reference number *Oref* with the current reference number *Cref* and the current reference number *Cref* with the new reference number (*Nref*). The same updating process on the backend server is performed with the mobile reader.

Table 3.3 New reference number generating process

Algorithm	**New Reference Number**
Input:	1 to 1000000
Output:	New Reference Number
Process:	Choose a random number form 1 to 1000000 return the new reference number

3.3.4 Authentication Process

3.3.4.1 Tag Response the Mobile Reader's Hello Message (Phase1.1)

As we explained before, the authentication process is divided into three phases with two sub phases on the first and second phase. Table 3.4 shows the process of the tag response to the mobile reader's hello message. The first message in the authentication process starts from the mobile reader to the tag, called Phase1.1. The mobile reader queries the tag with a hello message. The tag then generates a pseudorandom number ($Trnd_i$), encrypts it with the current encryption key ck_i , and sends the encrypted message (EM_i) to the mobile reader along with the tag current reference number $Cref_i$. Phase1.1 ends here. Figure 3.2 illustrates the flowchart of the tag response to the mobile reader hello message.

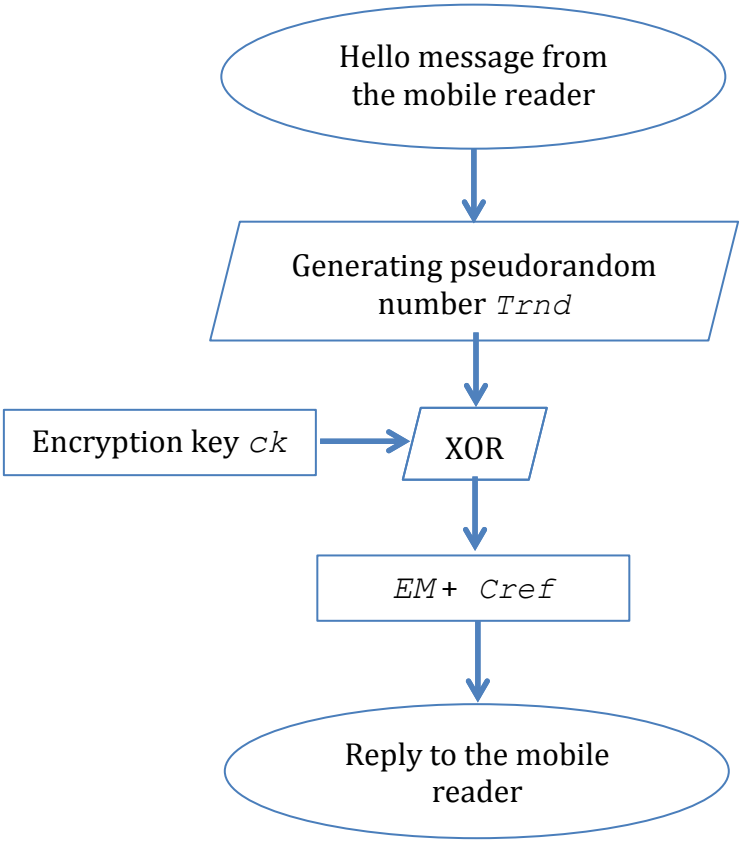

Figure 3.2 Flowchart of the tag's response to the mobile reader's hello message

Table 3.4 Tag response to the mobile reader's hello message

Algorithm	The Tag Response to the Mobile Reader
Input:	Hello Message
Output:	• Encrypted Message EM • Current tag reference number Cref
Process:	Generate Trnd EM = E_k(Trnd) = Trnd ⊕ k M = EM+Cref Return M

3.3.4.2 Tag Authentication by the Mobile Reader (Phase1.2)

When the mobile reader receives the reply from the tag, we have three cases:

1. The tag was not desynchronized during the last session or this is the first time to query the tag.
2. The tag was desynchronized during the last session.
3. The tag is unknown.

At first, the mobile reader will retrieve the current encryption key (ck_i) using the tag reference number ($Cref_i$) then tries to authenticate the tag by decrypting EM_i. If EM_i was successfully decrypted, then the tag is authenticated. If not, then the mobile reader will give a desynchronization attack warning, retrieves the old encryption message (ok_i), and tries to authenticate the tag by decrypting EM. If EM was successfully decrypted, then the tag is authenticated. If not, then the tag is unknown and the session will be terminated. If the tag was authenticated, then the mobile reader will generate a pseudorandom number ($Rrnd_i$), concatenates it with mobile reader ID (RID) and the decrypted message M_i , encrypts it with ck_i (if the

first decryption attempt was successful) or with ok_i (if the second decryption attempt was successful) as the following: $E_k(Rrnd_i, \text{RID}, M_i)$ where $M_i = Trnd_i$ and $k = ck_i \text{ or } ok_i$, and sends it to the backend server along with the $Cref_i$ as a plaintext, Phase1.2 ends here. Figure 3.3 illustrates the flowchart of the mobile reader authenticating the tag. Table 3.5 shows the tag authentication by the mobile reader.

3.3.4.3 Mobile Reader Authentication by the Backend Server (Phase2.1)

When the backend server receives the message from the mobile reader, we will have three cases:

1. The backend server was not desynchronized during the last session.
2. The backend server was desynchronized during the last session.
3. The mobile reader is unknown.

At first, the backend server will retrieve the current encryption key (ck_i) using the tag reference number ($Cref_i$) then tries to authenticate the mobile reader by decrypting EM_i. If EM_i was successfully decrypted, then the mobile reader is authenticated. If not, then the backend server will retrieve the old encryption message (ok_i) and tries to authenticate the mobile reader by decrypting EM. If EM was successfully decrypted, then the mobile reader is authenticated. If not, then the mobile reader is unknown and the session will be terminated. If the mobile reader was authenticated, then the backend server will generate a pseudorandom number ($Srnd_i$), concatenates it with backend server ID (SID) and the decrypted message M_i , encrypts it with ck_i (if the first decryption attempt was successful) or with ok_i (if the second decryption attempt was successful) as follows: $E_k(Srnd_i; \text{SID}; M_i)$ where M_i = ($Trnd_i$, SID, $Trnd_i$) and $k = ck_i \text{ or } ok_i$, and sends it to the backend server along with the $Cref_i$ as a plaintext .Phase2.1 ends here. Figure 3.4 illustrates the flowchart of the backend server authenticating the mobile reader. Table 3.6 shows mobile reader authentication by backend server.

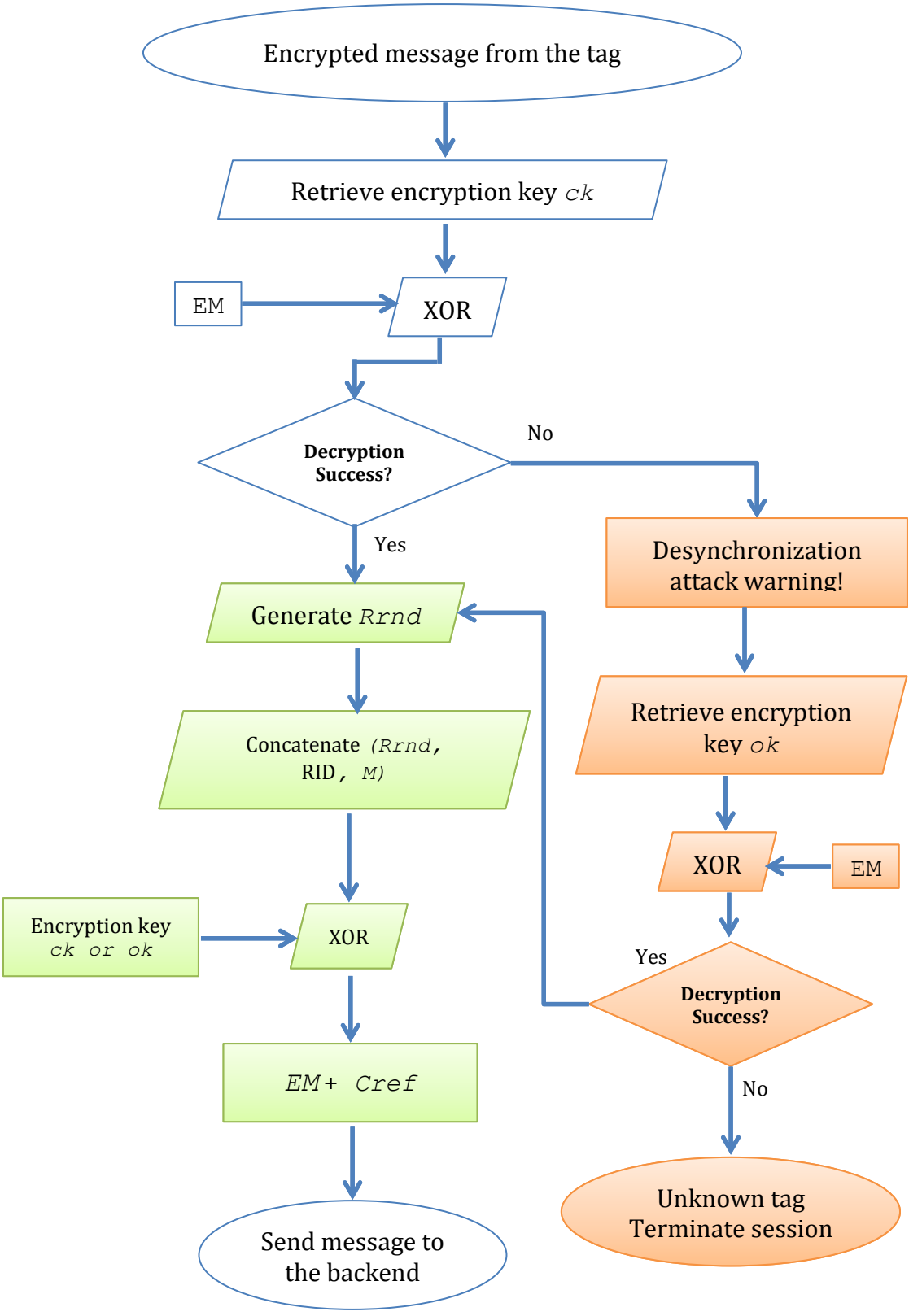

Table 3.5 Tag authentication process by the mobile reader

Figure 3.3 Flowchart of the mobile reader authenticating the tag

Algorithm **Tag Authentication by the Mobile Reader**

Input:	Response Message from the Tag
Output:	Boolean: • True if tag is authenticated + Message to the Backend Server • False if the tag not authenticated

Process:	Retrieve k using Cref list
	$M = D_k(EM) = EM \oplus k$
	if(Decryption success)
	{
	Generate Rrnd
	$EM = E_k(Rrnd+RID+M) = "Rrnd+RID+M" \oplus k$
	Send EM+Cref to Backend Server
	}
	else
	{
	Retrieve k using Oref list
	$M = D_k(EM) = EM \oplus k$
	if(Decryption success)
	{
	Generate Rrnd
	$EM=E_k(Rrnd+RID+M) = "Rrnd+RID+M" \oplus k$
	Send EM+Oref to Backend Server
	}
	else Return False

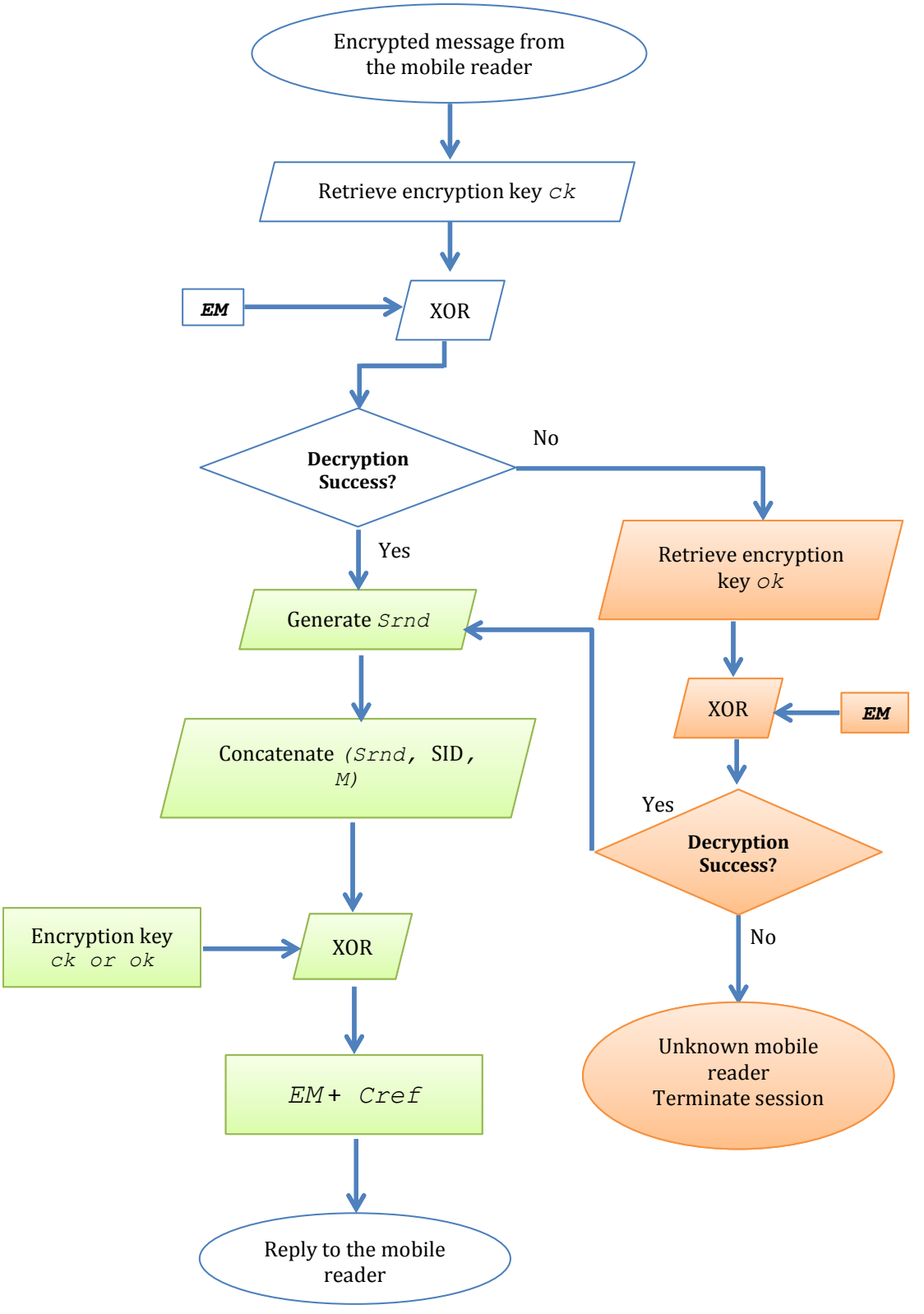

Figure 3.4 Flowchart of the backend server authenticating the mobile reader

Table 3.6 Mobile reader authentication process by the backend server

Algorithm	**Mobile Reader Authentication by the Backend Server**
Input:	EM from the Mobile Reader
Output:	Boolean: • True if Mobile Reader is authenticated + Reply to the Mobile Reader • False if Mobile Reader is not authenticated

Process:	Retrieve ck using Cref list
	Decrypt EM using ck
	M = D_{ck}(EM) = EM ⊕ ck
	if(Decryption success)
	{
	Generate Srnd
	Retrieve SID
	Concatenate Srnd+SID+M
	Encrypt them with ck
	EM=E_{ck}(Srnd+SID+M)="Srnd+SID+M" ⊕ ck
	Reply EM+ Cref to Backend Server
	}
	else
	{
	Retrieve ok using Oref
	Decrypt EM using ok
	M = D_{ok} EM) = EM ⊕ ok
	if(Decryption success)
	{
	Generate Srnd
	Retrieve SID

Concatenate Srnd+SID+M

Encrypt them with ok

EM=E_{ok}(Srnd+SID+M)=”Srnd+SID+M” ⊕ ok

Reply EM+Oref to Backend Server

}

else Return False

3.3.4.4 Backend Server Authentication by the Mobile Reader (Phase2.2)

Because of the possible impact of the desynchronization attack on updating the encryption keys, we may have the following situations when the backend server replies in Phase2.1:

1. There was no desynchronization attack or this is the first session after deploying the system. The authentication process then will be successful.
2. There was a desynchronization attack. In this case we have three possible scenarios (more explanation on the security evaluation is given later):
    a. Only the tag was desynchronized.
    b. Only the backend server was desynchronized.
    c. Both the tag and the backend server were desynchronized.

Notice that on case 2(b), both the tag and the mobile reader will authenticate each other using the new encryption key updated from last session, however, the backend server will not authenticate the mobile reader since it did not update to the new encryption key from the last session. The backend server then will terminate the session and the authentication will fail. Based on that, we will explain Phase 2.2 based on two possibilities.

Table 3.7 shows backend server authentication by mobile reader. The first possibility is when the backend server replies to the mobile reader (scenario a and c). In this case, the mobile reader will authenticate the backend server similar to the tag authentication on Phase1.2. However, the authentication will be 100% successful because the backend server and the mobile reader will have a common encryption key (either the current key ck or the old key ok).

When the mobile reader receives the reply from the backend server, the mobile reader will retrieve the current encryption key (ck_i) using the tag reference number ($Cref_i$) then tries to authenticate the mobile reader by decrypting EM. If

EM was successfully decrypted, then the backend server is authenticated. If not, the mobile reader will retrieve the old encryption message (ok_i), and tries to authenticate the backend server by decrypting *EM*. *EM* then should be successfully decrypted and the backend server will be authenticated. After authenticating the backend server, then Phase 2.2 ends and Phase 3 will start immediately. Figure 3.5 illustrates the flowchart of the mobile reader authenticating the backend server (first possibility).

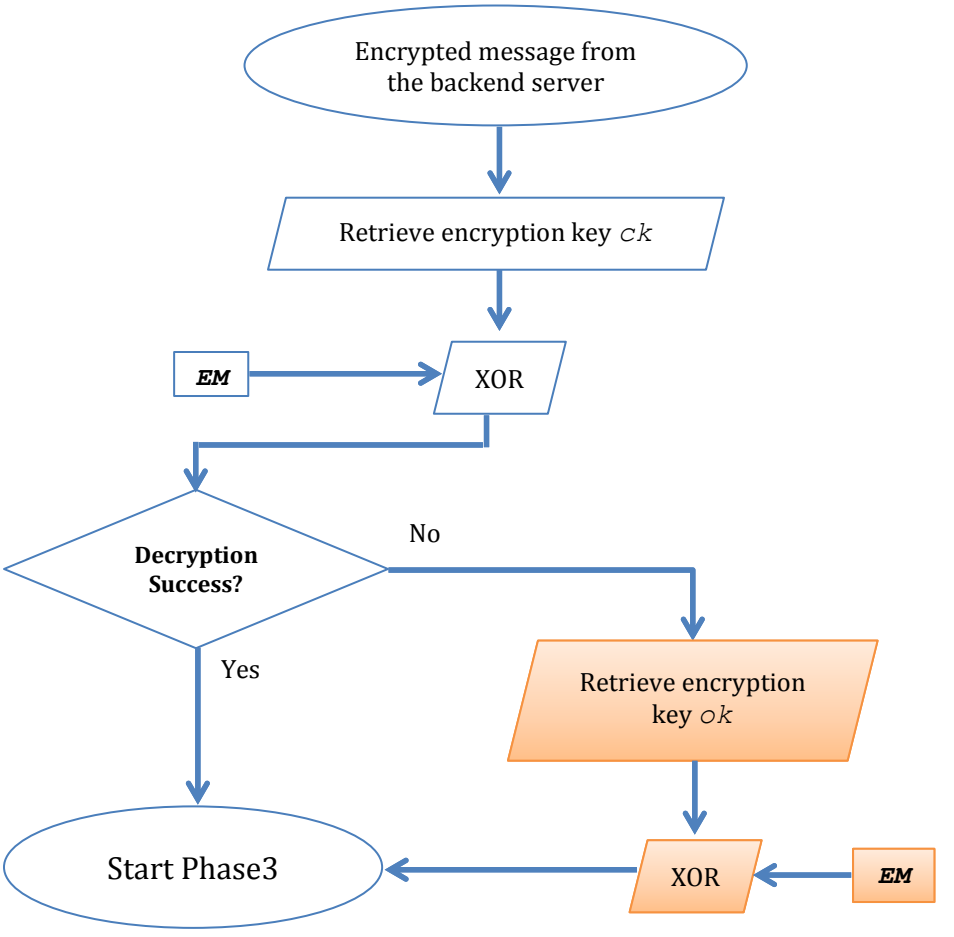

Figure 3.5 Flowchart of the mobile reader authenticating the backend server (first possibility)

The second possibility is when the backend server does not reply to the mobile reader (Scenario 2(b)). This is due to the failure of updating the encryption

keys on the backend server during the last session. The mobile reader will contact the backend server with a message encrypted with the mobile reader current encryption key. This current encryption key was the new encryption key on the last session and was not delivered to the backend server. The backend server does not have it and so it will reject the connection with the mobile reader and terminate the session. In this case, the mobile reader will restart the session again from the point when the last message was sent to the backend server (phase 2.1). However, the message will be encrypted with the old encryption key, which is currently equal to the backend server current key. The backend server will then reply (phase 2.2) and the mobile reader will authenticate the backend server the same way in the first possibility. Figure 3.6 illustrates the flowchart of the mobile reader authenticating the backend server (second possibility).

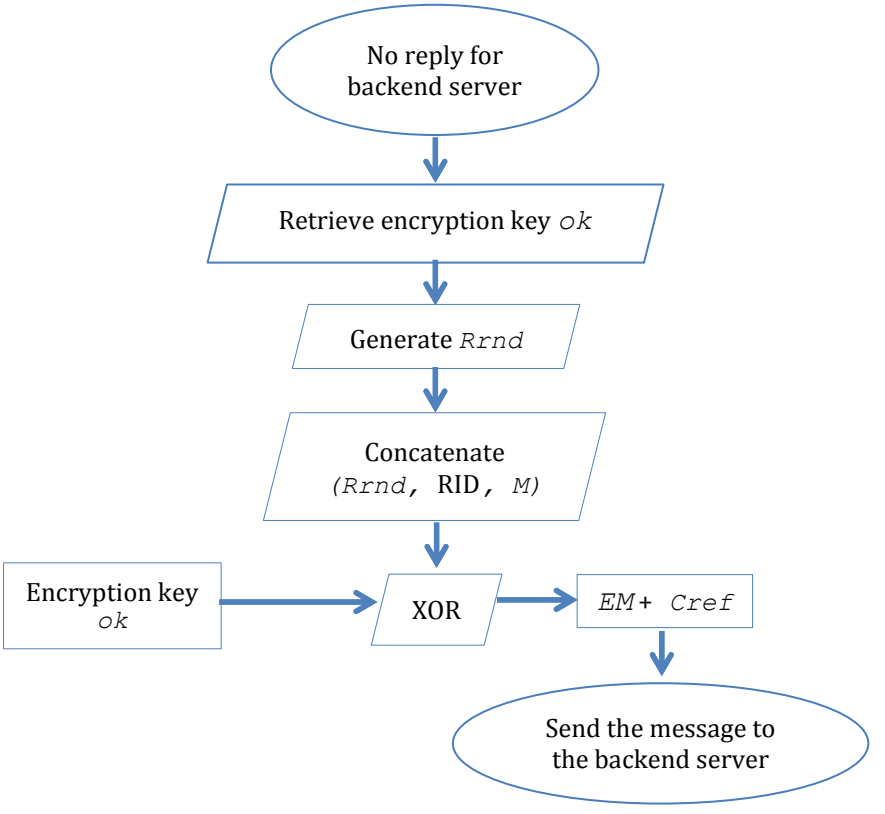

Figure 3.6 Flowchart of the mobile reader authenticating the backend server (second possibility)

Table 3.7 Backend server authentication process by the mobile reader

Algorithm	**Backend Server Authentication by the Mobile Reader**
Input:	EM from the Backend Server
Output:	Boolean: • True if Backend Server is authenticated • False if Backend Server is not authenticated

Process:	Retrieve k using Cref list
	$M = D_k(EM) = EM \oplus k$
	if(Decryption success)
	{
	Generate Srnd, Nref, and nk
	$EM=E_k$(Srnd+SID+nk+Nref)=”Srnd+SID+nk+Nref”⊕k
	Send EM+Cref to Backend Server and EM tag
	}
	else
	{
	Retrieve k using Oref
	$M = D_k(EM) = EM \oplus k$
	if(Decryption success)
	{
	Generate Srnd, Nref, and nk
	$EM=E_k$(Srnd+SID+nk+Nref)=
	”Srnd+SID+nk+Nref”⊕k
	Send EM+Cref to Backend Server and EM tag
	}
	else Return False

3.3.4.5 Updating the New Encryption keys and Reference Number (Phase3)

When Phase 2.2 is finished successfully, the mobile reader will generate the new encryption keys (nk_i), the new reference number ($Nref_i$), and a pseudorandom number ($Rrnd_i$). The mobile reader then encrypts them together to get EM_i, sends EM_i+Cref_i (or $Oref_i$ depends on the previous response from the backend server), and sends EM_i to the tag. The tag will decrypt EM_i, retrieve nk_i and $Nref_i$, and updates ck_i to nk_i and $Cref_i$ to $Nref_i$. The backend server will decrypt EM_i, retrieve nk_i and $Nref_i$, updates ok_i to ck_i, updates ck_i to nk_i, $Oref_i$ to $Cref_i$, and $Cref_i$ to $Nref_i$. Finally, the mobile reader updates ok_i to ck_i, updates ck_i to nk_i, $Oref_i$ to $Cref_i$, and $Cref_i$ to $Nref_i$. Figure 3.7 illustrates the flowchart of the updating mechanism.

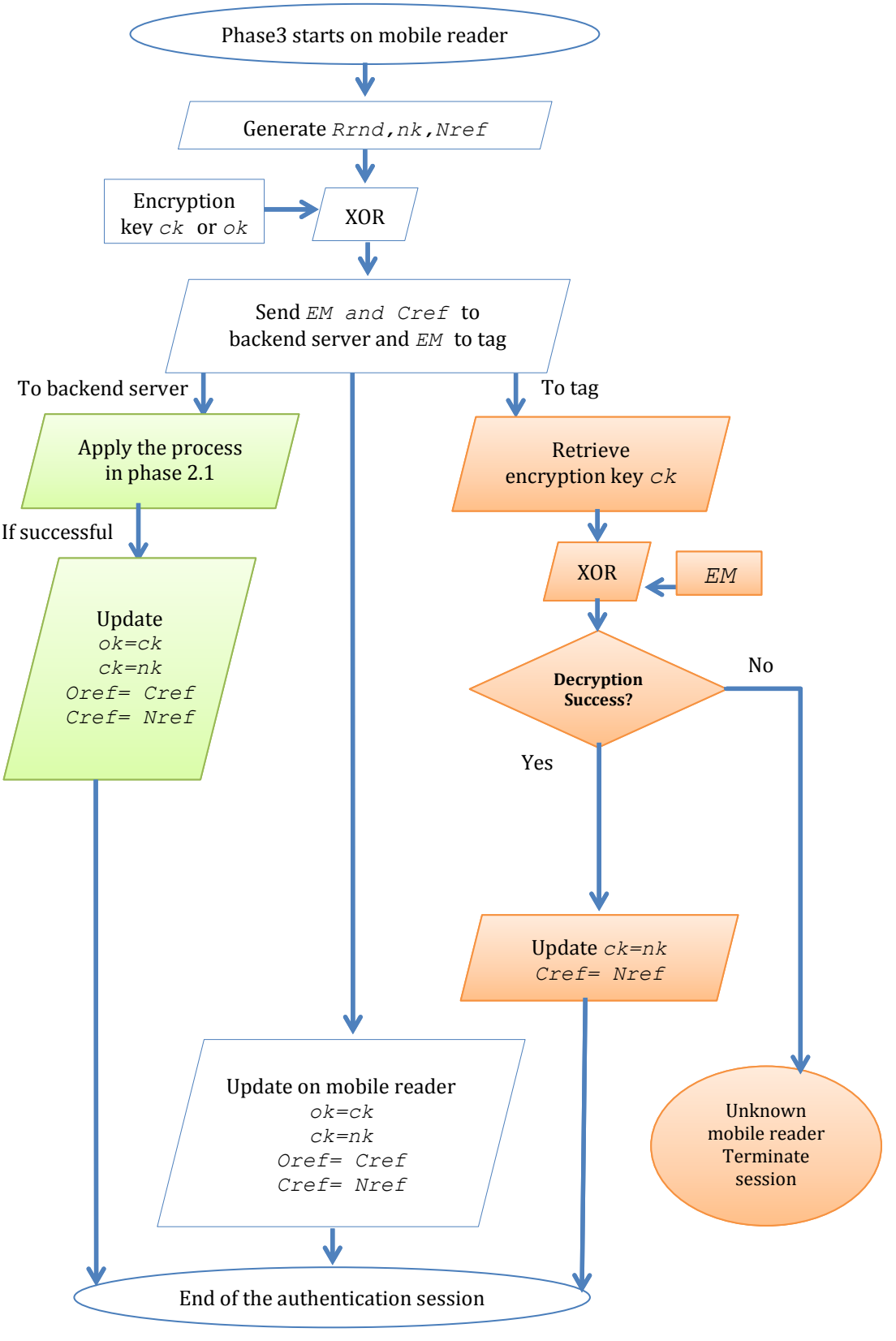

Figure 3.7 Flowchart of the updating mechanism

3.4 Summary

In this chapter, we presented our proposed scheme that is based on a mutual authentication between the tag, the mobile reader, and the backend server. The mobile reader is the central entity, which is responsible for generating and updating the new encryption keys. The encryption keys are shared among all the entities and updated after each authentication session. In the next chapter, we will discuss the evaluation methodology of our proposed scheme.

Chapter 4: Evaluation Methodology and Experimental Results

In this chapter, we will present evaluation methodology and the experimental results of the proposed scheme along with the discussion. We implemented the proposed scheme as a proof of concept using Java. The following analysis was performed:

1) Energy consumption and memory requirement analysis.
2) Security analysis.

We implemented one server and one reader for testing our implemented scheme. Approximately, 30 tags were tested to completely fill out the current and old values of reference numbers and encryption keys on the database. Next, we launched the desynchronization attack 5 times, first on the tags, and then on the backend server. The system behavior against the desynchronization attack was tested each time. In total, 10 desynchronization attacks were launched on the system.

4.1 Energy Consumption and Memory Requirement Analysis Methodology

In this section, we will analyze the proposed scheme resources consumption. We will measure the number of arithmetic and logical operations along with the memory consumption. This measurement is used to estimate the resources' computational the proposed scheme would consume.

4.2 Security Analysis Methodology

As mentioned earlier, the proposed scheme should fulfill the following criteria: confidentiality, data integrity, authentication, non-repudiation, and forward security. Confidentiality means that the data is exchanged safely without being exposed to any unknown entity. Data integrity means that the data is sent by a

legitimate entity. Authentication is when each entity can prove that it is a legitimate entity. Non-repudiation means the source cannot deny a message was sent from it. Forward security refers to the property of the encrypted data where it is not possible to decrypt it without having the encryption key.

Moreover, the proposed scheme should be secured against the following attacks (explained in Chapter 2):

1. DoS Attack.
2. Cloning Attack.
3. Desynchronization Attack.
4. Replay Attack.
5. Eavesdropping Attack.
6. Man-In-The-Middle Attack (MITM).
7. Location Tracking.

4.3 Experimental Results

We will discuss the results of the energy consumption and memory requirement analysis along with the security analysis of the proposed scheme.

4.3.1 Energy Consumption and Memory Requirement Analysis

The proposed scheme uses the following computational functions for authentication:

- Pseudorandom Number Generator (PRNG).
- The logical XOR function for encryption ($E_k(M)$) and decryption ($D_k(EM)$) of the exchanged messages.

We will analyze each entity for both computational performance and memory requirement. Notice that the computational analysis depends on the authentication phases and if the previous authentication session had an attack or not.

4.3.1.1 The Tag

During the normal authentication session (i.e. the tag was authenticated by the mobile reader), the tag will use its computational resources twice (n = 2; where n refers to the number of operations). In the situation where the mobile reader will not authenticate the tag during Phase 1.2, the tag will use its computational resources only once (n = 1) since the mobile reader will terminate the session. Table 4.1 shows the computational overhead of the tag.

Table 4.1 Computational analysis for the tag

Computational Operations	**The Tag Response**
PRNG	1
XOR	1 x n
Total	1+n

Table 4.2 shows the memory requirement for the tag. The tag stores only one 24-bit reference number (*Cref*) and one 128-bit encryption key (k).

Table 4.2 Memory requirement analysis for the tag

The Data	**Memory Requirement (bits)**
Cref	24
k	128
Total	152

4.3.1.2 The Mobile Reader

Since the mobile reader is the central entity, it is responsible for generating the new encryption keys and reference number. This feature gives us the advantage

of reducing the probability of the desynchronization attack to be successfully launched because if one entity was attacked (i.e. the tag or the backend server) the other one will still be updated. However, by giving this feature to the mobile reader, we will increase the memory requirement of the mobile reader, and thus, be limited to the memory size of the mobile reader.

As the authentication process depends on the success of each phase, the mobile reader will have different scenarios in using its computational resources. Before explaining these scenarios, we will define three variables related to these scenarios:

- m : refers to the number operations.
- m' : refers to the two extra usage of PRNG to generate the new reference number and new encryption key. The purpose we are using this variable is because these two extra usages will be performed only on Phase 3; which mean they won't be used before if Phase 3 was not reached.
- m'' : refers to the one time that the decryption algorithm will not be used. This is applied when the backend server does not reply to the mobile reader on Phase 2.1. The mobile reader will then repeat Phase 2.1, however; one decryption (decryption of *EM* from Phase 1.2) will not be done since *EM* is already decrypted.

The scenarios are:

1) If the tag did not authenticate the mobile reader, then the tag will terminate the authentication session and the mobile reader will not use its computational resources ($m = 0$, $m' = 0$, $m'' = 0$; where m refers to the total number of time the mobile reader will use its resources, m' to the number of PRNG).
2) If the tag authenticated the mobile reader, and the backend server authenticated the mobile reader from the first time ($m = 2$, $m' = 1$, $m'' = 1$).
3) If the backend server authenticated the mobile reader from the second time. In this case, the mobile reader will use its computational resources again to

communicate with the backend server using the old reference number (*Oref*) ($m = 3, m' = 1, m'' = 1$).

4) If the backend server did not authenticated the mobile reader on the two times ($m = 2, m' = 0, m'' = 1$).

Notice that during Phase 3, the mobile reader will use the pseudorandom number generator three times to generate the random number, the new encryption key, and the new reference number. Table 4.3 shows the computational overhead of the mobile reader.

Table 4.3 Computational analysis for the mobile reader

Computational Operations	**The mobile Reader Response**
PRNG	$1 \times m + 2 \times m'$
XOR	$2 \times m - m''$
Total	$3m + 2m' - m''$

Table 4.4 shows the memory requirement for the mobile reader. The mobile reader will contain N number (i.e. number of tags) of following data:

- Two encryption keys (the current *ck* and the old *ok*). The key size is 128 bits per key.
- Two reference numbers (the current *Cref* and the old *Oref*). The key size is 24 bits per number.
- The actual tags IDs. The size of the keys will depend on the type of passive tags the corporation or system owner will use.

Table 4.4 Memory requirement analysis for the mobile reader

The Data	Memory Requirement (bits)
Cref and *Oref*	24 x 2 x N
Ck and *ok*	128 x 2 x N
Tag actual ID	System Dependent (SD)
Total	(304 + SD) x N

4.3.1.3 The Backend Server

The authentication process utilizes the backend server resources twice, on Phase 2 and Phase 3. Table 4.5 shows the computational overhead of the backend server. The best scenario will be when the backend server authenticates the mobile reader in both phases using the current encryption key (n = 1), and the worst case is when the backend server authenticates the mobile reader in both phases using the old encryption key (n = 2).

Table 4.5 Computational analysis for the backend server

Computational Operations	The Backend Server Response
PRNG	1
XOR	2 x $n + 1$
Total	$2n + 2$

The data stored on the backend server is an exact copy of the data on the mobile reader, so the same memory requirement analysis of the mobile reader applies to the backend server. The backend server is also responsible for updating new mobile readers introduced to the system or mobile readers that do not have the updated information of any tag that was updated by another mobile reader.

4.3.2 Security Analysis

We will present in this section the security analysis of our proposed scheme against the security goals.

4.3.2.1 Security Goals Analysis

- Confidentiality: The tag will only send the reference number in plaintext and it is updated every session, which guarantees confidentiality.
- Data Integrity: When the messages are exchanged between the three entities, only the entity with the shared key can decrypt the message. The pseudorandom number generated by each entity can also works as a unique signature for each entity.
- Authentication: All the entities are authenticating each other using the shared encryption key.
- Non-repudiation: The random number generated by the entity works as a signature to it.
- Forward security: XOR guarantees that the data will not be decrypted without the appropriate encryption key. The encryption keys are also updated each session to guarantee no risks come from cracking older encryption keys.

4.3.2.2 Analysis of the Proposed Scheme Resistance to the Attacks

1. DoS Attack: The proposed scheme can resist DoS attack. This is due to the fact that any entity will terminate the authentication session if the encrypted data was not successfully authenticated.
2. Cloning Attack: This is usually a physical attack the proposed scheme cannot prevent it from being launched. However, it can be detected if one of the two identical tags attempts to authenticate itself after the other one. Notice that if one of the two tags was successfully authenticated twice, the other tag will not be authenticated since the

current and old encryption keys have been changed. This may result to the original tag to be an invalid tag to the system.

3. Desynchronization Attack: The proposed scheme is able to recover from this attack by using the previous reference number and encryption key. The proposed scheme can detect if the attack was launched on the tag or the backend server. It also lowers the risk of this attack by sending the updates of the reference number and the encryption key in parallel through the mobile reader. In this case, if the tag or the backend server was desynchronized, the other one will not be affected by the attack.
4. Replay Attack: The proposed scheme is protected against this attack because the encryption keys are updated after the end of each session. That means any replayed message will be ignored since the encryption key was changed.
5. Eavesdropping Attack: This attack is useless against the proposed scheme since the data is sent encrypted. Even if the attacker captured the reference number, on the next session the reference number will be changed and not useful to the attacker to use.
6. Man-In-The-Middle Attack (MITM): Similar to the eavesdropping, the attacker will not get any information since it is encrypted.
7. Location Tracking: The attacker cannot track any of the system's entities because each entity generates its own pseudorandom number that changed the pattern of the data transmission every time a message is being transmitted. Moreover, if the attacker captured the reference number, it will not be useful because as soon as the session ends, the reference number will be updated and changed.

In comparison to the previously discussed protocols in Chapter 2, our proposed scheme is similar to these protocols in two ways: first, using symmetric encryption keys system; second, using private keys for the system's entities. Notice that our proposed scheme is similar to Sandhya and Rangaswamy's proposed

protocol since all entities have an encryption key, however, the encryption key on the reader is not updated where our scheme updates encryption keys at the end of each session.

Table 4.6 shows a comparison of the proposed scheme against the protocols discussed on Chapter 2. The protocols on the table are arranged in sequence with the order of on Chapter 2. The comparison is based on the attacks the protocol is protected against or not. Notice that all the protocols do not resist desynchronization attack.

Table 4.6 A comparison of the proposed scheme against other protocols

	Protocol 1 [9]	Protocol 2 [14]	Protocol 3 [11]	Protocol 4 [4]	Protocol 5 [6]	Protocol 6 [7]	Proposed Scheme
DoS	•	•		•			•
Cloning	•	•	•	•	•	•	•
Desynch.							•
Reply	•	•	•	•	•		•
Eavesdr.	•	•		•		•	•
MITM	•	•					•
Location tracking		•	•	•	•	•	•

4.3.3 Implementation Snapshots

We show snapshots of the proposed scheme implementation. The snapshots show three scenarios:

1. Normal authentication process, shown by figure 4.1.
2. Authentication process with tag desynchronized during previous session, shown by figure 4.2.
3. Authentication process with backend server desynchronized during previous session, shown by figure 4.3.

```
On reader
+++++++
Reading...
First hello message from the reader...    (phase 1.1)

On tag
+++++++
Tag responding to the hello message with reference number: 679613
message to be encrypted: 75801726     (phase 1.2)

On reader
+++++++
ID from tag: 679613
Message Back: 75801726
end of phase 1

message to be sent from the reader to the server (phase 2.1): 776248

On server
+++++++
Message Back: 77624834;1;75801726
message to be sent from the server to the reader (phase 2.2): 860734

On reader
+++++++
Message Back: 860734;1;77624834;1;75801726
(end of phase 2)

message to be sent to both the tag and the server(phase 3): 4675880;

On tag
+++++++
Message Back: 4675880;ffe9ad7f6bf4812744fef04ad222bfde;966;1
The key has been updated on the tag.

On reader
+++++++
The tag-reader update is done.

On server
+++++++
Message Back: 4675880;ffe9ad7f6bf4812744fef04ad222bfde;966;1
The key has been updated on the server.

On reader
+++++++
The server-reader update is done.
The reader is updated
end of phase 3
```

Figure 4.1 Normal authentication session

```
Tag responding to the hello message with reference number: 602247
message to be encrypted: 95240211     (phase 1.2)

On reader
+++++++
ID from tag: 602247
Message Back: 95240211
Warning: This tag was not updated on the previous session process. Possible desync
message to be sent from the reader to the server (phase 2.1): 72440115;1

On server
+++++++
Message Back: 72440115;1
message to be sent from the server to the reader (phase 2.2): 27342865;1;72440115;

On reader
+++++++
(end of phase 2)

message to be sent from the reader to both the tag and the server (phase 3): 81880

On tag
+++++++
Message Back: 81880664;f9f6ea13bae5425058afdfa16202bf08;817;1
The key has been updated on the tag.

On reader
+++++++
The tag-reader update is done.
```

Figure 4.2 Tag authentication with a desynchronization attack during last session

```
message to be sent from the reader to the server (phase 2.1): 3365992;1;92779097

On server
+++++++
The reader is unknown!

On server
+++++++
Message Back: 92779097
message to be sent from the server to the reader (phase 2.2): 19387512;1;92779097

On reader
+++++++
(end of phase 2)

Warning: The Server was not updated on the previous authentication session. Possible desynchronization attack!
message to be sent from the reader to both the tag and the server (phase 3): 14337644;2e22e6a6b80747d71c46c955d
```

Figure 4.3 Backend server authentication with a desynchronization attack during last session

4.4 Discussion

In this section, we discuss the experimental results of the proposed scheme stated earlier in this chapter. Following that, we present the advantages and challenges of the proposed scheme.

In this proposed scheme, we designed an ultra-lightweight mutual mobile RFID authentication protocol. The reader on the proposed scheme is mobile, and acts as the central entity between the tag and the backend server. Each entity will authenticate the other two entities using a shared 128-bit encryption key. The encryption key is updated at the end of each session. The proposed scheme design showed that it met the security goals and resists the known RFID attacks.

4.4.1 The Computational Performance and Memory Requirements

As the design of the proposed scheme is an ultra-lightweight design, the overhead of the computational performance of the design rests mainly on the tag since it has the lowest computational resources of the three entities. As shown on the tag computational performance results, the worst case will be 3 operations, which is low.

On the mobile reader and backend server side, their computational resources are higher than the tag, so the computational resource consumption will not be an issue. However, optimizing the resource consumption will improve the system overall performance. The mobile reader has an additional concern on the resource consumption, which is power consumption since its mobility features requires a battery-powered resource. As shown on the mobile reader computational performance results, the worst case will be 10 operations, which is still low. For the backend server, the worst case will be 6, which is low. The overall computational performance of the proposed scheme on the worst case is 18 operations, which is still low considering that the majority of these operations is done on the mobile

reader which has a high computational resources.

In order to solve the desynchronization problem, our scheme requires the addition of the old reference number and the old encryption key to encrypt and decrypt data. On the other hand, memory requirement has been increased on the mobile reader and the backend server sides since we are storing the old reference number and old encryption key of each tag.

4.4.2 Modeling the Proposed Scheme

In order to make the proposed scheme an ultra-lightweight, only pseudorandom number generator and XOR operations were used. Pseudorandom number generator is used to generate the encryption keys, the reference numbers, and the random numbers.

The encryption key size choice of 128-bit is to add complexity to the encryption process even with the limited resources on the tag. The encryption key size guarantees that the produced encryption key is very hard and unlikely to predict, which means that brute force attempts to get the encryption key will be very hard to the attacker. The number of encryption key possibilities is 32^{36}. Moreover, the encryption key changes at the end of each session, which adds another level of complexity to the attacker to deal with.

The reference number was design to protect the actual tag ID. The updating mechanism guarantees that the changing of the reference number will prevent tracking attacks on the tag while the tag actual ID is safe and not changed.

4.4.3 Generalizing of the Proposed Scheme

The proposed scheme was designed to work with mobile RFID systems that have a mobile reader. It is an ultra-lightweight scheme, delivers the RFID security

goals in general, and resists the known attacks on RFID systems. We can say that the proposed scheme module is applicable to any RFID environments since the proposed scheme authenticates each entity individually and needs low resources in order to be implemented.

As an application, our scheme can be applied to various areas such as NFC-based smartphones that have built-in readers.

4.4.4 Scalability of the Proposed Scheme

The proposed scheme depends on the mobile reader to be the central entity. This could be an issue if the number of tags in a big one (some RFID systems has millions of tags). Storing these tags information including two encryption keys, two reference numbers, and tags IDs could increase the use of the mobile reader resources and, as a result; reduce the productivity of the system. The same issue is applicable to the backend server since it has a copy of the mobile reader data.

4.4.5 Advantages of the Proposed Scheme

The following are the advantages the proposed scheme address:

- **Mutual authentication:** the proposed scheme, unlike proposed protocols, considers the three mobile RFID entities as individual entities. The other schemes conceder the reader and the backend server as one entity so the backend server is the main entity. That means if any attack was done on the backend server, the whole system will be affected. On our proposed scheme, if the attack was done on the backend server, the system will not be completely affected.
- **Enhanced Security and Privacy:** our proposed scheme can resist all known attacks on RFID systems. Its symmetric encryption key mechanism gives it the advantage over other scheme to maintain a secure and private authentication session.

- **Low Resources Consumption:** as an ultra-lightweight scheme, our proposed scheme does not require a high resources.

4.4.6 Challenges of the Proposed Scheme

As any RFID system, the proposed scheme faces the following challenges:

- The proposed scheme may have a scalability issue when it is deployed in an n RFID system with large tags number. The memory requirement on both the mobile reader and the backend server will increase and have some effects on the system performance.
- EPCglobal standards [19] limit the use of pseudorandom number generators and the cryptographic hash functions [17][18]. These limitations could affect the authentication process. A custom made tags can be manufactured to solve this issue. As for the previously proposed protocols, protocols number 3 [11], 4 [4], and 6 [7] are compatible with EPCglobal standards. Protocols number 1 [9], 2 [14], and 5 [6], however, did not provide any information on their compatibility with the EPCglobal standards.

4.5 Summary

In this chapter, we discussed the results of our proposed scheme. We presented the advantages and challenges of our proposed scheme. The next chapter will be the conclusion and future work.

Chapter 5: Conclusion and Future Work

5.1 Conclusion

In this work, we presented a mutual authentication scheme for mobile RFID systems. The tag, the mobile reader, and the backend server will authenticate each other individually. The mobile reader is the central entity that generates and updates the encryption keys and reference numbers. The proposed scheme uses a symmetric key encryption where the encryption key is shared among the tag, the mobile reader, and the backend server. The encryption key and reference number is updated at the end of each authentication session.

The proposed scheme meets the following security and privacy standards: confidentiality, data integrity, authentication, non-repudiation, and forward security. It also resists the known RFID attack such as: DoS attack, cloning attack, desynchronization attack, replay attack, eavesdropping attack, man-in-the-middle attack (MITM), and location tracking.

5.2 Future Work

The current proposed scheme has a potential for optimization and enhancements. One optimization area is to analyze the stored data on both the mobile reader and the backend server to reduce the data size in order to optimize the system performance and scalability. Using the Trivium passive tag that was used in the Trivium algorithm can make an enhancement. The new tag is suitable for RFID environments and needs a small number of gates, yet it is very fast in processing [20]. Finally, we can analyze the current proposed scheme for new attacks on RFID systems such as Tango attack [8].

References:

[1] Pateriya, R.K. and Sharma, S., "The Evolution of RFID Security and Privacy: A Research Survey", *2011 International Conference on Communication Systems and Network Technologies (CSNT),* vol., no., pp.115-119, 3-5 June 2011 doi:10.1109/CSNT.2011.31

[2] Venkataramani, G. and Gopalan, S., "Mobile Phone Based RFID Architecture for Secure Electronic Payments using RFID Credit Cards" *The Second International Conference on Availability, Reliability and Security, 2007. ARES 2007*., vol., no., pp.610,620, 10-13 April 2007 doi: 10.1109/ARES.2007.105

[3] Hu, J., Wang, D., Ding, Y., Zhang, J., and Tan, H. "Design and Implementation of Intelligent RFID Security Authentication System", *2010 IEEE International Conference on RFID-Technology and Applications (RFID-TA), vol., no.,* pp.17–19. 2010

[4] Sun, H.-M. and Ting, W.-C. "A Gen2-Based RFID Authentication Protocol for Security and Privacy". *IEEE Transactions on Mobile Computing. vol.8, no.8,* pp.1052,1062 2009

[5] D.N. Duc, J. Park, H. Lee, and K. Kim, "Enhancing Security of EPCglobal Gen-2 RFID Tag against Traceability and Cloning," *Proc. 2006 Symp. Cryptography and Information Security*, 2006.

[6] Bai Enjian, Ge Huayong, Wu Kejia, and Zhang Wen. "A trust-third-party based key management protocol for secure mobile RFID service". *Wireless Communications, Networking and Mobile Computing, WiCom '09. 5th International Conference*, pp. 1-5. 2009

[7] Hyeong-Chan Lee and Jeong Hyun Yi. “Development Of Privacy-Preserving RFID Authentication System Using Mobile Devices”. *ICT Convergence (ICTC), International Conference*, pp. 760-765. 2011

[8] Tian, Y., Chen, G. and Li, J. “A New Ultra-lightweight RFID Authentication Protocol with Permutation”. *IEEE Communications Letters, vol. 16, no. 5*, pp. 702–705. doi:10.1109/LCOMM.2012.031212.120237, 2012

[9] Sandhya, M. and Rangaswamy, T. R. “A Forward Secured Authentication Protocol For Mobile”, *International Journal of Information Technology and Knowledge Management vol. 4, no. 2,* pp. 549–553. 2011

[10] Morshed. M., Atkins, A. and Yu, H. “An Efficient and Secure Authentication Protocol for RFID Systems”, *Automation and Computing (ICAC), 2011 17th International Conference on , vol., no.*, pp.51-56, 2011

[11] Chang, A. Y., Tsai, D.-R., Tsai, C.-L. and Lin, Y.-J. “An improved certificate mechanism for transactions using radio frequency identification enabled mobile phone”. *43rd Annual 2009 International Carnahan Conference on Security Technology*, pp. 36–40. doi:10.1109/CCST.2009.5335567. 2009

[12] H.-Y. Chien, “SASI: a new ultra-lightweight RFID authentication protocol providing strong authentication and strong integrity,” *IEEE Trans. Dependable and Secure Computing, vol. 4, no. 4,* pp. 337–340, 2007.

[13] Lee, D., Kim, S., Kim, H., & Park, N. “Mobile Platform for Networked RFID Applications”. *2010 Seventh International Conference on Information Technology: New Generations,* pp. 625–630. doi:10.1109/ITNG.2010.188, 2010

[14] Chia-Hui Wei, Min-Shiang Hwang, and Augustin Yeh-hao Chin. “A Mutual Authentication Protocol for RFID”. *IT Professional*, pp. 20-24, 2011.

[15] M. Lethonen, D. Ostojic, A. Ilic, and F. Michahelles," Securing RFID systems by detecting tag cloning," *In 7th International Conference on Pervasive Computing – Pervasive 2009, volume 5538 of LNCS*, pp. 291–308., 2009.

[16] Ahmadian, Z., Salmasizadeh, M., and Aref, M. R. (n.d.). "Desynchronization Attack on RAPP Ultra-lightweight Authentication Protocol", *Information processing letters, vol. 113, no. 7*, pp. 205. 2013

[17] Habibi, M. H. and Gardeshi, M. "Cryptanalysis and improvement on a new RFID mutual authentication protocol compatible with EPC standard". *2011 8th International ISC Conference on Information Security and Cryptology,* pp. 49–54. doi:10.1109/ISCISC.2011.6062337. 2011

[18] Doss, R., Zhou, W., Yu, S., and Gao, L. "A novel mutual authentication scheme with minimum disclosure for RFID systems". *2011 Seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing,* pp. 544–549. doi:10.1109/ISSNIP.2011.6146526. 2011

[19] GS1 EPCglobal. EPCglobal Tag Data Standards Version 1.4, http://www.epcglobalinc.org/standards/

[20] Gong, J., Chen, G., Li, L., and Li, J. "A Secure Authentication Protocol for RFID Based on Trivium", *2011 International Conference on Computer Science and Service System (CSSS), vol., no.,* pp.107-109, 2011

[21] Sun, H., Ting, W., and Wang, K. "On the security of Chien's ultra-lightweight RFID authentication protocol ", *IEEE Transactions on, vol. 8, no. 2,* 2011

[22] D'Arco, P. and De Santis, a. "On Ultra-lightweight RFID Authentication Protocols". IEEE Transactions on Dependable and Secure Computing, *vol. 28, no.* 4, pp. 548–563. doi:10.1109/TDSC.2010.75, 2011

[23] Mubarak, M. F., Manan, J. A., and Yahya, S. “A critical review on RFID system towards security, trust, and privacy (STP)”. *2011 IEEE 7th International Colloquium on Signal Processing and its Applications,* pp.39–44. doi:10.1109/CSPA.2011.5759839, 2011