SECURITY AND PRIVACY ANALYSIS OF AUTHENTICATION PROTOCOLS IN RFID SYSTEMS

by

İmran Ergüler

B.S., in Electrical-Electronics Engineering, Boğaziçi University, 2003

M.Sc., in Electrical-Electronics Engineering, Boğaziçi University, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Electrical-Electronics Engineering

Boğaziçi University

2011

SECURITY AND PRIVACY ANALYSIS OF AUTHENTICATION PROTOCOLS IN RFID SYSTEMS

APPROVED BY:

Prof. Emin Anarım (Thesis Supervisor)	.....
Prof. Mehmet Ufuk Çağlayan	.....
Prof. Mustafa Kemal Cılız	.....
Assoc. Prof. Albert Levi	.....
Assist. Prof. Gökay Saldamlı	.....

DATE OF APPROVAL: 14.06.2011

ACKNOWLEDGEMENTS

There are numerous people I would like to thank due to their support and encouragement in completion of this thesis. First of all, all my gratitude goes to my supervisor Prof. Emin Anarım. He has guided my research during both my undergraduate and graduate research at Boğaziçi University for about eight years and also supervised my senior project and my master thesis. Over these years, he has truly inspired me during my research, provided helpful hints and always supported me through the good and the bad times. So, it has been a pleasure to work with him. I also would like to thank Dr. Gökay Saldamlı for always being helpful and encouraging to me. My thanks also go to my thesis jury members Prof. Ufuk Çağlayan, Dr. Albert Levi and Prof. Kemal Cılız for spending their rare available time on examining my thesis. Additionally, I would like to thank Dr. Orhun Kara for providing valuable discussion during my study.

I am indebted to my institution TUBITAK-BILGEM for motivating its employees to pursue an academic career and funding research expenditures. Also, my huge thanks go to my project manager Gökhan Vıcıl and my project coordinator Dr. Murat Erat for giving me an opportunity to complete my Ph.D. study, consistently encouraging me to go on working and giving valuable advices. My thanks also go to my friends and colleagues Tayyar Güzel, Özgür Ören, Yunus Karamavuş, Yusuf Çeri and Orhan Üçtepe for the pleasant and productive environment and their long lasting friendship. A special thank goes to my friend Fatih Kara for his notable effort in organizing sport activities.

Last but not least I would like to mention my parents, my brother and my sister for their unconditional support. They have always encouraged me when I took important decisions and always been with me when I needed them. Finally, my deepest thanks go to my wife, Saime and my lovely daughter Ceyda for their love and their enduring support. Without their patience and encouragement, this thesis would never have been written.

ABSTRACT

SECURITY AND PRIVACY ANALYSIS OF AUTHENTICATION PROTOCOLS IN RFID SYSTEMS

Radio Frequency IDentification (RFID) technology continues to flourish as an inherent part of virtually every ubiquitous environment. However, it became clear that the public— implying the industry— seriously needs mechanisms emerging the security and privacy issues for increasing RFID applications.

This thesis examines security and privacy of RFID authentication protocols and presents three main contributions. First, we show that RFID protocols having unbalanced states for which tag identification is performed in different order of computational complexities are subject to side-channel attacks and do not preserve the RFID privacy.

Second, we introduce a timing attack such that if the database querying in tag identification is performed through a static process, RFID protocol is vulnerable to the proposed attack that could easily jeopardize the system's untraceability criteria. We formulate success probability of our attack and demonstrate its success on some well known protocols.

Finally, we analyze security of RFID delegation systems and present an unnoticed security flaw that makes tag impersonation attack possible. To overcome these weakness, we give some possible countermeasures.

ÖZET

RFID SİSTEMLERİNDE KİMLİK DOĞRULAMA PROTOKOLLERİNİN GÜVENLİK VE GİZLİLİK ANALİZİ

Radyo Frekansı ile Tanımlama (RFID) teknolojisi günlük hayatın önemli bir parçası olmaya devam etmektedir. RFID uygulama alanlarının genişlemesi güvenlik ve gizlilik mekanizmalarına olan ihtiyacı ortaya çıkarmaktadır. Bu tez, RFID kimlik doğrulama protokollerinin güvenliğini ve gizliliğini inceler ve başlıca üç başlık altında katkı sağlar. Birincisi RFID etiket tanımlama işlemini farklı durumlar için farklı karmaşıklık derecelerinde gerçekleştiren protokollerin bu durumlarına dengesiz durumlar diye adlandırıp, bu tür durumlara sahip protokollerin yan kanal saldırılarına açık olduğunu ve gizliliği sağlayamadıklarını gösteriyoruz.

İkinci olarak, eğer RFID protokolünde etiket tanımlama işlemi veri tabanında sabit bir sorgulama mekanizması ile gerçekleşiyorsa, önerdiğimiz zaman saldırısına bu sistemin korumasız olduğunu belirtiyoruz. Ayrıca saldırının başarı olasılığını formüle edip bazı bilinen protokoller üzerinde uyguluyoruz.

Son olarak, RFID görevlendirme protokollerinin güvenliğini inceleyip, önceden farkedilmemiş bir güvenlik problemini belirtiyoruz. Bu güvenlik açığından dolayı etiketlerin sistemde taklit edilebilir olduğunu gösteriyoruz. Ayrıca, bu zayıflıklardan korunmak için bazı olası önlemlerden bahsediyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathcal{A}$	Adversary
$\mathcal{DB}$	The back-end database
EPC	Electronic product code
$\mathbf{Exp}_{\mathcal{S}}^{priv}$	Privacy experiment for system $\mathcal{S}$
$H()$	One-way hash function
HID	Hashed value of ID
ID	Identity of a tag
K_P	Public key
N	Number of tags in the database
$O()$	Computational complexity
PID	Previous ID of a tag
PRF	Pseudo random function
$\mathcal{R}$	RFID reader
$\mathcal{R}^+$	Online RFID reader
$\mathcal{R}^-$	Off-line or delegated RFID reader
r_R	Random nonce generated by reader
r_T	Random nonce generated by tag
$SecReq$	Secret update request message
$\mathcal{T}$	RFID tag
Δ	Elapsed time between 2nd and 3rd message flow
γ	The attacker's precision in timing attack
$\|\|$	Concatenation operator
$\oplus$	XOR operator
$\vee$	Bitwise OR operator
$\wedge$	Bitwise AND operator
$\ll$	Left rotation operator

LIST OF ABBREVIATIONS

BAP	Basic authentication protocol
CHT	Cheon Hong Tsudik
CRC	Cyclic redundancy code
CW	Chang and Wu
DPLK	Duc Park Lee Kim
HM	Henrici and Muller
IFF	Identify friend or foe
LAS	Linear-time authentication system
LK	Lim and Kwon
OSK	Ohkubo Suzuki Kinoshita
PRNG	Pseudo random number generator
RF	Radio frequency
RFID	Radio frequency identification
RKKW	Rhee Kwak Kim Won
RNG	Random number generator
SLAS	Static linear-time authentication system
SM	Song Mitchell
SMD	Song Mitchell delegation
SSM	Scalable Song Mitchell
UAP	Unbalanced authentication protocol

1. INTRODUCTION

1.1. Motivation

Radio Frequency IDentification (RFID) technology with an increasing popularity in manufacturing, supply chain management, inventory control, etc. is the main drive behind the pervasive computing. Because of its low production costs and tiny size, RFID gadgets are considered as a replacement technology for barcodes and other means of traditional identification tools.

Typically, an RFID system incorporates three components: Tags, one or more readers, and a back-end server. On top this hardware, a set of networking rules including the authentication protocols reside. Despite their advantages, RFID systems have some computational constraints mostly driven by the cost concerns of low-cost RFID tags. It hampers use of public key cryptography and only permits to have security schemes based on symmetric key primitives at the tag side. Indeed, this fact makes design of a fully privacy-preserving authentication protocol as a challenging task. Solving this delicate task has attracted the interest of the security community, and many authentication protocols have been proposed recently. However, it is shown that majority of these proposals like in [1–10] do not provide security and privacy requirements.

Meanwhile, security community appreciated the so called identification schemes based on the symmetric cryptography that are more realizable on the tag nodes. Nevertheless, in a system having a large number of tags, these schemes take the burden from the tags and place it on the server nodes. To identify or authenticate a responder tag, the server has to perform an exhaustive search over the list of all tag entries in the database. For instance; in a system having N tags, symmetric cryptography based protocols require $O(N)$ decryption operations. Such a tedious search procedure brings up the so called scalability problem that many systems including [5, 11] have suffered from. To reduce the cost of exhaustive search and enhance the scalability of the sys-

tem, new security protocols such as [9, 10, 12–20] have been proposed, but all of these protocols possess serious security or privacy drawbacks.

As discussed above, there still exists some security, privacy or scalability related problems in present RFID authentication protocols. Consequently, in this thesis we aim to address some common weaknesses in RFID protocols and provide relevant countermeasures to facilitate development of stronger protocols.

1.2. Contributions

In this dissertation, we investigate security and privacy of RFID authentication protocols and make the following major contributions:

- *Introduction of Unbalanced States:* We prove that any RFID authentication protocol involving unbalanced states for which the server performs identification of the tag in different order of computational complexities are traceable and this violates RFID privacy requirements.

- *Exploiting Security Flaws in Recent Schemes:* Recently, new security protocols have been proposed to reduce the computational load either at the server side or at the tag side. We show however that these protocols suffer from different types of attacks while aiming to achieve an optimal fully secure privacy-preserving design.

- *Timing Attack on Database Search Mechanism:* We discover a new privacy threat for RFID protocols and introduce a timing attack. We show that if the database querying in tag identification phase of an RFID protocol is performed through a static process, the system is vulnerable to the proposed attack. Thus, it cannot provide untraceability criteria, although the protocol itself might satisfy the necessary security requirements of the RFID system specifications. Moreover, we outline a countermeasure by precisely describing the database query mechanism in order to resist such an attack.

- *An Inherent Security Flaw in Delegation Protocols:* We review some RFID delegation protocols and present a common security flaw that threatens system security. Also, we propose a delegation scheme to counteract such a weakness.

1.3. Organization of the Thesis and Publications

The rest of this thesis is organized as follows:

In Chapter 2, we give an overview of RFID systems, illustrating the main components and describing typical applications.

Chapter 3 describes some possible attack types and privacy threats relevant to RFID systems. Also, it notes some design requirements for a secure, privacy-preserving and efficient RFID system and explores the adversary's capabilities for different scenarios.

In Chapter 4, we review some recent related work and briefly state security weaknesses or other shortcomings in these schemes.

In Chapter 5, we introduce the unbalanced states notion in RFID protocols and show that protocols having unbalanced states cannot provide RFID privacy requirements. To illustrate this brutal security flaw, we conduct our analysis on different RFID protocols. The results of this chapter are published in [21, 22].

In Chapter 6, we present subtle flaws of some recent RFID protocols that are designed to mitigate computational load either at the server side or at the tag side. Also, we draw some lessons to be learned from these security vulnerabilities. Some parts of this chapter are based on [23–25] and some parts of this chapter are published in [22, 26].

In Chapter 7, we propose a timing attack by which an adversary can break the privacy of the system, if the database querying in tag identification is done through a static process. We demonstrate our attack on some well known protocols and outline a countermeasure by precisely describing the database query mechanism. Hence, we would like to call our attention to the importance of database querying mechanism in tag identification process. Furthermore, we analyze the success probability of the attack in terms of the system parameters such as the number of tags, number of cryptographic operations has to be carried out and server's computational power. Parts of this chapter are published in [27].

In Chapter 8, we first give an overview of RFID delegation system. Then, we present an inherent security flaw for RFID delegation protocols and define some security requirements that a basic RFID delegation protocol should take into account to avoid such a weakness. Parts of this chapter are published in [28].

Finally, Chapter 9 summarizes the contributions of this thesis and draws the main conclusions. It also points out future research lines.

As mentioned above, the main results presented in this thesis have been submitted or published in [21–28].

2. RFID SYSTEMS

Radio Frequency Identification (RFID), originated from military IFF (Identify Friend or Foe) systems in World War II, enables identification of objects that are labeled with small tags, capable of wireless data transmission, in a wide variety of environments without any physical contact and need for line-of-sight alignment. Indeed, this feature allows objects to be identified in large numbers without the need of physical contact. This technology is the main drive behind the pervasive computing and has been applied in various fields such as supply-chain management, inventory monitoring, payment systems, automobile immobilizers, and medical management. Because of its low production costs and tiny size, RFID gadgets are considered as a replacement technology for barcodes and other means of traditional identification tools. In this chapter, we give an overview of RFID systems and their applications.

2.1. RFID System Components

A typical RFID system consists of mainly three components: RFID tags, one or more RFID readers, and a back-end server. On top this hardware, a set of networking rules including the authentication protocols reside. In the following parts, we describe these components in detail.

2.1.1. RFID Tags

RFID tag is a small microchip and an antenna providing wireless data transmission [29]. In an RFID system, each object is labeled with a tag which transmits object related data through the radio frequency (RF) interface upon interrogation by an RFID reader. Tags can be classified based on four main criteria: source of power, type of memory, computational power, and functionality, but the most common criteria is the source of power. According to source of power, tags can be categorized into three types as passive, semi-active and active RFID tags [30]. Figure 2.1 illustrates some examples of RFID tags.

Source: *http://www.directindustry.com*

Figure 2.1. RFID tag examples.

Passive Tags: Powerless or passive RFID tags are low-cost devices with limited memory and limited computational capabilities. These tags do not possess any internal power source and cannot initiate communication with RFID readers [30]. The required energy of such a tag is provided from the electromagnetic signals transmitted by the reader. Hence, this type of tags have the shortest communication range and can only operate in presence of an RFID reader.

Semi–Active Tags: These tags have battery on their board to power microchips. However, they obtain needed energy for communication from the signal transmitted by the reader. Compared to passive tags, these tags have longer communication range [30, 31].

Active Tags: These tags are equipped with a power source such as a battery. Therefore, they can provide power for communication and microchips without the readers' transmitted signal, i.e. they can work even if absence of the reader. Compared to passive and semi-active tags, they allow longer communication range, but costs higher than the former types [30, 32].

2.1.2. RFID Readers

An RFID reader or transceiver is a device used to read information and possibly also write information into RFID tags [30]. It is basically incorporates an RF module, a control unit, and a coupling element to interrogate electronic tags via RF communication [31]. A reader is typically connected to a back-end database for further processing of information received from a tag. Also, in this case, if necessary, it delivers the back-end server messages to the tag through air interface. Figure 2.2 shows some examples of RFID readers.

Source: *http://www.directindustry.com*

Figure 2.2. RFID reader examples.

2.1.3. Back-end Server

The back-end server manages data related to the tags in the system [30]. It stores related information of the RFID tags in the system such as product and price information etc. and also their secrets. It receives tag response from the reader and processes this response as a key to retrieve the detailed information of the object to which the tag is attached. Also, it can provide certain security services like secret key update and authentication of the tags. In general, it is assumed that RFID reader is directly connected to the back-end server with a secure channel.

2.2. RFID Applications

2.2.1. Advantages of RFID over Barcode

As a result of their low production costs and tiny size, RFID tags are considered as the replacement technology for barcodes and other means of traditional identification tools. The followings summarize major advantages of RFID tags over barcodes:

First, a barcode identifies the type of object on which it is labeled. For instance, a printed barcode printed on a pen can indicate "This is a black pen of ABC brand". Note that it only identifies the type of object among many millions of identically manufactured objects [29], but not the object itself. In other words, all black pens of the same brand may have the same serial number that the barcode emits. On the other hand, RFID systems identify individual items attached with a tag, e.g. each black pen of the same brand gives a unique information that is meaningful in database.

Second, barcode needs line of sight in optical scanning of an object. Thus, except in the most rigorously controlled environments, barcode scanning requires human intervention [29]. Unlike barcodes, in identification of objects, RFID systems do not require line of sight and careful physical positioning of the tagged objects. Moreover, thanks to RFID technology RFID tagged objects can be identified by an RFID reader at rates of hundreds per second. Also, because RFID technology has a greater read range than barcodes, the security problems related to thievery and counterfeiting of products and goods can be eliminated [33].

Third, in contrast to barcodes, RFID tags can realize on-chip computation. Hence, cryptographic protocols for authentication can be implemented and security and privacy demands can be satisfied [34].

Last, barcodes are usually printed on paper products or hard metals that make them vulnerable to harsh environments [33]. Barcodes do not function properly when

they have been torn, disfigured, or detached from their objects. Hence, compared to RFID tags barcodes are weak in terms of durability.

2.2.2. Example Applications of RFID Technology

RFID technology has found many applications in disparate domains such as supply-chain management, inventory monitoring, payment systems, automobile immobilizers, medical management. In this part, we review some example applications where RFID tags are used.

Transportation: Transportation industry is one of the main fields where RFID technology is used. RFID applications in transportation involves traffic management, tolls and fees, fare collection, equipment identification, fleet management, solid waste hauling, and fuel dispensing [33]. For example, RFID tags have been used for transport payment for person/vehicle identification or for recording prepaid balances [35]. These system has been adopted in different countries. For example, in US E-ZPass system and in Norway AutoPass system serves as an electronic vehicle tolling. Also, Octopus cards in Hong Kong and EZ-Link cards in Singapore are examples of transit fare collection [35].

Access Control: RFID technology is used as an electronic key in this field, for controlling access to secure locations or equipment [35]. Indeed, this ensures that only authorized persons are allowed access related rooms or buildings. For example, many hotels and business facilities employ RFID systems to control access of their employee or customers. Besides access control, such a system also makes possible to manage and track RFID attached individuals [33].

Supply Chain Management: Basically, supply chain management interests movement of items between organizations across a supply chain from raw materials to finished products delivered to the consumer [36]. Although the advantages of RFID in supply chains have been well-known, use of RFID tags is supply chain management has been so far limited. The reason is apparent – high cost of RFID tags with considering high

volumes of objects. Since it is usual that tags may be attached to large inventories of relatively inexpensive products, for a cost efficient supply chain management, the RFID tags need to be inexpensive. According to some analysts, if an RFID tag costs less than 5 cents, then the technology can be truly competitive [37]. By the early 2000's cost of RFID tags had reduced and in several pilot projects RFID tags are deployed to obtain required data collection [36]. The US Department of Defense and Wal-Mart mandated all their major suppliers to use RFID technology in their supply chains and this gave a great push on use of RFID systems in supply chains [31, 35]. Use of RFID in supply chain management provides many benefits. For instance, it provides superior tracking of goods, i.e. leading to a reduction in lost goods [35]. Also it allows real time awareness of stock. Furthermore, by means of RFID technology automation of warehousing and distribution become possible.

Library: RFID can be very useful in libraries for example comfortably inventorying of books, easiy checking-in and checking-out books and monitoring the traffic. Around 30 million library items worldwide are estimated to have RFID tags attached [35].

Automobile Security: RFID tags are also used in automobile industry such as the in immobilizer system by which the car starts only if the key has the proper chip embedded into it [33].

Healthcare: RFID is now commonplace for the healthcare industry. For instance, control of counterfeit products, tracking hospital staff, devices and supplies can be easily handled with use of RFID tags. Another example, researchers presented that an RFID-enabled medicine cabinet could provide the medications are taken in a timely fashion [29].

3. DESIGN REQUIREMENTS OF RFID AUTHENTICATION PROTOCOLS

As mentioned in previous chapter, RFID technology has gained a lot of attention from industrial community and has been found in various fields. One reason behind widespread deployment of RFID systems is RFID technology provides low-cost solutions for large-scale applications. The increasing popularity resulted in further cost reductions, i.e. very simple low-cost RFID tags have been produced. Nevertheless, this advantageous feature limits capabilities of an RFID tag and imposes protocol designers to use only computationally simple operations at the tag side. It is equivalent to say that the low-cost RFID tags cannot compute complex functions due to limitation of power and storage resources.

On the other side, pervasive adoption of RFID in public and involvement of humans raise the security and privacy issues that have to be addressed. In this chapter, we discuss design requirements of an RFID protocol for low-cost tags; mainly security, privacy and performance requirements.

3.1. RFID Privacy Requirements

RFID technology poses privacy concerns because humans cannot sense the interaction between a fake reader and an RFID tag and the tags themselves typically maintain no history of past readings. That is to say tags are promiscuous: they can be read by unauthorized entities other than legal readers and without their owners' knowledge [38]. RFID privacy violations can be broadly split in two classes: tag tracking and information leakage [36].

3.1.1. Tag Tracking

RFID technology raises significant privacy issues regarding the traceability concerns. Basically, by tracking a tag the actions of individuals carrying it can be monitored and their future behaviors potentially inferred using RFID tags associated with them [36]. While a person could be traced by tracking his/her mobile phone through a carrier, such a method is no more useful once the phone is turned off. However, this is not the case for someone carrying an RFID gadget. First of all, most users are not aware that they are carrying RFID tags. In fact, even if they know it, tags could not be turned off in general and worse, it automatically responds to queries via radio signals. Therefore, in RFID systems, the attack scenarios and accompanying countermeasures are quite different than the typical wired or wireless systems. For example, suppose a tag always sends its static ID upon receiving a request from the reader. Then a person carrying an item with this RFID tag, a book or a bag, can be easily traced by an adversary with using a rogue reader [39]. In general, tracking activities of an adversary can be named as tracking attacks by which an adversary can traces tags from the protocol message flows [40]. This attack type violates the location privacy of tag's holder such that an attacker can trace tags movement [41].

Intuitively, a protocol satisfies untraceability if an adversary is not able to recognize a previously observed tag [42]. Untraceability issue has been treated formally in different security models, notably driven by Avoine in [43], by Vaudenay in [44] and by Juels–Weis in [45]. According to [45] as a formal definition, untraceability can be defined in terms of privacy experiments. The aim of the adversary in this experiment is to distinguish between two different tags within the limits of its computational power and functionality-call bounds. The privacy experiment consists of two phases: The learning phase and the challenge phase. In the former, the adversary $\mathcal{A}$ may initiate a communication with the reader $\mathcal{R}$ (ReaderInit) or tags $\mathcal{T}$ (TagInit). Then he may interact with them according to the corresponding protocol steps. In the challenge phase, the adversary selects two tag candidates $\mathcal{T}_0$ and $\mathcal{T}_1$ to be tested. Then he chooses one of these tags randomly, called $\mathcal{T}^*$ and $\mathcal{A}$ is given access to this tag. The adversary may again interact with the reader and the tags. Eventually, $\mathcal{A}$ termi-

nates the test and decides whether the selected tag is $\mathcal{T}_0$ or $\mathcal{T}_1$. If the adversary has a non-negligible advantage in successfully guessing the selected tag, then he succeeds in attack and the protocol is not untraceable.

3.1.2. Tag Information Leakage

In this case, information stored in an RFID tag is recovered without the consent of its owner [36]. In other words, the information leakage problem arises when data sent by the tag gives information about the attached object. So, if an adversary cannot learn any information associated with the tag when given the corresponding communication messages, we can say that any information leakage does not occur [46]. For instance, if the information associated with a tag attached to a passport is obtained by an adversary, then some personal data of the corresponding individual is revealed [47].

3.1.3. Providing Privacy Requirements

From the previous parts, it is apparent that in order to prevent from the above threats an RFID protocol satisfy the following requirements [47]:

- To resist tag tracking attacks, messages sent by the tag must be anonymous. Hence, an adversary cannot distinguish messages of the from those of other tags.

- To overcome information leakage problem, the RFID system guarantees that information associated with a tag can be only accessed by authorized entities.

3.2. RFID Security Requirements

In order to give a better understanding of security issues in RFID systems, we provide an overview of some possible attack types. Nevertheless, for a detailed description we refer the reader to the studies in [40, 48–51] that have developed a classification

of RFID attacks and presented a comprehensive analysis of possible security threats in RFID systems.

- Tag impersonation attack: In this type of attack, an adversary attempts to impersonate a legal tag to spoof an authorized reader. Thus, the adversary convinces the reader to believe the fake tags are legitimate.

- Denial of service (DoS) attack: In these attacks, the normal operation of RFID tags is interrupted by intentionally blocking access to them [51]. These incapacitates RFID tags temporarily or permanently.

- Replay attack: In replay attacks, an adversary copies valid responses of RFID protocol flows and retransmits them at a later time to one or more parties in order to perform impersonation [51].

- Eavesdropping: In this type of attacks, an adversary intercepts messages exchanged by legitimate parties and is able to decode them. Also, this type of attack can be performed in both directions tag to reader and reader to tag [51].

- Backward traceability: We define backward untraceability as given in [47]. A protocol is backward traceable, if an adversary who knows the internal state of the tag at time τ, identifies the tag's past interactions at time $\tau' < \tau$.

- Forward traceability: We define forward untraceability as given in [47]. A protocol is forward traceable if an adversary who knows the internal state of the tag at time τ, identifies the tag's future interactions at time $\tau' > \tau$.

In addition to attack types, attackers can be categorized depending on their attack capabilities. In [47], Song and Mitchell have classified attackers into two groups as follows:

- Weak attacker: This type of attacker has capabilities to observe and manipulate communications between a reader and a tag. In other words this attacker has the ability to modify, insert or delete messages that agree with the corresponding protocol's procedures. For example, a weak attacker can realize tag impersonation, DoS and replay attacks.

- Strong attacker: A strong attacker has all capabilities of a weak attacker, but additionally he has ability to compromise a tag and access all tag's internal information. Backward traceability and forward traceability are related issues with only a strong attacker.

Considering the above security threats, a secure RFID protocol has to resist all possible attacks mentioned above as tag impersonation, DoS and replay attacks, also it has to provide backward and forward untraceability even if the tag is compromised. These requirements can be summarized as follows:

i) As long as the tag is not compromised, the protocol does not allow authentication of fake tags, i.e. resistance to tag impersonation attacks.
ii) Deletion, insertion or modification of messages exchanged between a reader and a tag does not prevent future communications of the tag with the server, i.e. resistance to DoS attacks.
iii) The protocol prevents an adversary to perform impersonation by reusing messages transmitted between a server and a tag, i.e. resistance to replay attacks.
iv) The protocol provides backward and forward untraceability even if the tag is compromised [32].

3.3. RFID Performance Requirements

In addition to providing security and privacy, the following issues should be also taken into consideration by the RFID protocol designers.

3.3.1. Computational Effort of Tag

Low-cost RFID tags cannot perform computationally intensive operations, because of low computational power, derived from the RFID reader's signal, and small memory size. Thus, another design criterion for RFID security protocols is the computational effort required at the tag side, i.e. the required operations at the tag side must be realizable.

3.3.2. Storage of Tag

The cost effective RFID tags have very limited area for storage, so an implementable protocol fulfills the RFID tag specifications and cannot impose a storage exceeding capacity of a tag.

3.3.3. Message Traffic

The number of message flows defined in tag–reader interaction, also number and size of the messages exchanged between a tag and a reader should be minimized for a fast and effective protocol [32].

3.3.4. Scalability

Although issue of computational complexity taken by the server in tag identification is detailed in the next chapter, we shortly point out the related subject, scalability, here. Since low-cost RFID tags cannot realize computationally complex operations, many RFID schemes take the burden from the tags and place it on the server nodes. To identify or authenticate a responder tag, the server has to perform an exhaustive search over the list of all tag entries in the database. For instance; in a system having N tags, symmetric cryptography based protocols require $O(N)$ decryption operations. Such a tedious search procedure brings up the so called scalability problem that many systems like [2, 3, 5, 11, 52] have suffered from. Although different solutions have been proposed to mitigate computational load at the server side, such as design of ultra-

lightweight protocols (see Section 4.4), use of RFID delegation methods (see Section 8.1), introduction of batch mode authentication (whereby a reader interrogates a multitude of tags and later identifies/authenticates them in bulk [53]), all of these have some shortcomings and it is still a challenging task to design authentication protocols for low-cost RFID tags resisting all of the known attacks and threats and at the same time providing scalability.

4. OVERVIEW OF RFID AUTHENTICATION PROTOCOLS

In recent years, several RFID authentication protocols have been proposed to resolve the security and privacy issues. Although details can vary dramatically from one protocol to another, these RFID protocols can be classified depending on time complexity taken by the server for its overall computations in tag identification process or the cryptographic algorithms used in tags with respect to their computational complexity. For the former case, in [39], such a classification is presented and the protocols are categorized into three groups based on the time complexity performed by the server in tag identification: Constant-time, logarithmic-time and linear-time protocols. As can be inferred from their names, for these RFID protocols the server accomplishes tag identification with complexity $O(1)$, $O(\log N)$ and $O(N)$ respectively, where N denotes the number of tags in the database. On the other hand, for the latter case the RFID protocols can be classified into four groups with respect to computational complexity at the tag side. These groups are namely full-fledged, simple, lightweight and ultralightweight protocols [17]. The full-fledged protocols use conventional cryptographic functions such as symmetric encryption, one-way functions or public key algorithms. Simple type protocols require random number generator (RNG) and one-way hashing. The third class named as lightweight protocols demand RNG and simple functions like Cyclic Redundancy Code (CRC) checksum but not hash function. The last category referred as "ultralightweight" protocols only use simple bitwise operations on tags.

In this chapter, we present an overview of previously proposed RFID authentication protocols with following the classification based on the computational efficiency of tag identification and discuss their security weakness or performance drawbacks. Additionally, we consider examples of ultralightweight protocols.

4.1. Linear-Time RFID Protocols

In a protocol of this class, the authentication of a tag imposes a linear search at the server side. As the number of the tags in the system increases, the server suffers from the heavy overloads in tag identification process and this results in the major disadvantage of this class – the efficiency and the scalability issue. Apart from the efficiency problem, in this part we give a brief security analysis for some examples of this class.

4.1.1. The Ohkubo Suzuki Kinoshita Protocol

In [2], Ohkubo et al. proposed an RFID privacy protection scheme, named as OSK protocol, providing indistinguishability (i.e. a tag output is indistinguishable from a truly random value and unlinkable to the ID of the tag) and backward untraceability.

This protocol relies on hash chains. When a tag is queried by a reader, it sends a hash of its current identifier with H_1 and then updates it using a second hash function H_2. Each tag stores in its memory a random identifier s_i^1. The message flows of the protocol can be depicted as follows: The reader sends an identification request to the tag and receives back $r_i^k = H_1(s_i^k)$ where s_i^k is the current identifier of the tag. Then tag replaces s_i^k by $s_i^{k+1} = H_2(s_i^k)$. On the server side, from r_i^k, the system identifies the corresponding tag. In order to do this, it constructs the hash chains from each N initial value until it finds the match with checking $r_i^k = H_1(H_2^t(s_j^1)$ for $0 \leq t \leq \delta$, i.e. until it reaches a given maximum limit δ on the chain length. The threshold δ is the number of read operations on a single tag between two updates of the database and $H_2^t()$ denotes t iterations of $H_2()$. A summary of the OSK protocol is given in Figure 4.1.

In [1] it is stated that OSK scheme is not immune to replay attacks, and an adversary can impersonate a tag without knowing the tag secrets.

Data Base / Reader $[ID_i, s_i^1]$		**Tag** $[s_i^k]$
	ID request	
Linear search through all ID_j in database:	$\longrightarrow$	Computes:
For each ID_j make iterations with check	$\{r_i^k\}$	$r_i^k = H_1(s_i^k)$
$r_i^k = H_1(H_2^t(s_j^1)) : 0 \leq t \leq \delta$	$\longleftarrow$	$s_i^k = H_2(s_i^k)$

Figure 4.1. The OSK protocol.

Data Base / Reader $[ID_i]$		**Tag** $[ID_i]$
	Query	Generates r
Linear search through all ID_j in database	$\longrightarrow$	Computes:
such that $M_1 = H(ID_j \|\| r)$	r, M_1	$M_1 = H(ID_i \|\| r)$
	$\longleftarrow$	
	ID_i	
	$\longrightarrow$	

Figure 4.2. The RAC protocol.

4.1.2. The Randomised Access Control Protocol

Weis et al. [52] presented an RFID security scheme, Randomized Access Control (RAC), based on one-way hash functions with requiring a linear search in tag identification process. Upon reception of a reader query, the tag firstly generates a random number r and responds with a hash of the tag ID and r as $M_1 = H(ID_i||r)$. When a tag is locked, it sends different response in every session by producing a different random number and hashing it with the ID. The server performs an exhaustive search of all stored IDs in the database until it finds a match ID_j such that $M_1 = H(ID_j||r)$. After the tag is identified, it transmits ID_i to unlock the tag. The protocol is depicted in Figure 4.2.

Notice that, a tag impersonation attack is possible for this model because an intercepted response can be replayed. Moreover, because the tag ID is fixed, for a compromised tag scenario the scheme does not provide backward untraceability.

4.1.3. The Rhee Kwak Kim Won Protocol

In [3], Rhee et al. proposed a hash function based challenge-response authentication scheme which we call the RKKW protocol. The authentication process can be summarized as follows:

The reader transmits $Query$ and a random number r_1 to the tag. The tag generates a random number r_2, computes $H(ID||r_1||r_2)$ and sends the result with r_2 to the reader. The reader delivers the tag's response to back-end server. Next, for each ID stored in the back-end database, the server concatenates ID, r_1 and r_2 then hashes the result and checks whether or not it is equal to hash value obtained from the tag in order to authenticate the tag. The search process continues till a match is found. If the authentication is successful, the back-end database sends $H(ID||r_2)$ to the reader and the reader forwards it to the tag. A summary of the scheme is shown in Figure 4.3.

As stated in [1], the RKKW protocol cannot provide backward untraceability, since the attacker can trace the past communications from this tag in case of compromise of a tag.

4.1.4. The Duc Park Lee Kim Protocol

In [4] Duc et al. suggested a synchronization based protocol, referred as the DPLK protocol, for the EPCGlobal Class 1 Gen-2 RFID tag. The system uses a pseudo-random number generator and a cyclic redundancy code. According to the protocol the server stores the following values for each tag: EPC_i, tag's access pin PIN_i and the tag key K_i. On the other hand, each tag keeps $\{EPC_i, PIN_i, K_i\}$ tuple.

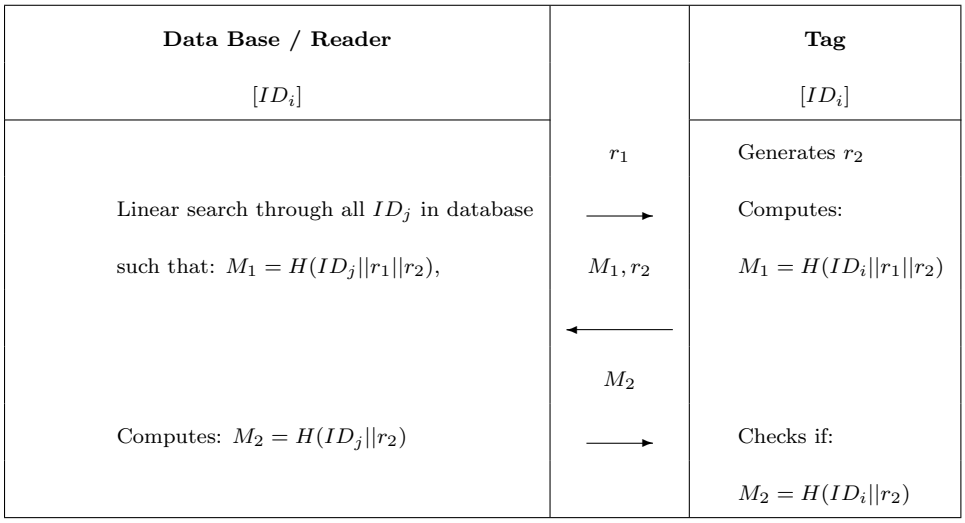

Figure 4.3. The RKKW protocol.

We can briefly describe the steps of the protocol as given below: The reader firstly queries a request to tag. The tag generates a random number r, computes $M_1 = CRC(EPC_i||r) \oplus K_i$ and $C = CRC(M_1 \oplus r)$. Then the tag sends the values (M_1, C, r) back to the reader, which will forward these values to the server, where EPC_i is Electronic Product Code and CRC stands for Cyclic Redundancy Code. For each tuple (EPC_i, K_i) in back-end database, the server verifies that $M_1 \oplus K_i$ equals $CRC(EPC_i||r)$ and $C = CRC(M_1 \oplus r)$. If it can find a match, then the tag is successfully identified and authenticated. Next, the server computes $M_2 = CRC(EPC_i||PIN||r) \oplus K_i$ and sends M_2 to the tag through the reader in order to authenticate itself to the tag. Once the tag receives M_2, by using the local values it computes $CRC(EPC_i||PIN||r) \oplus K_i$ and checks it with M_2. If they are equal, the tag authenticates the server and updates its secret K_i with $K_i = PRNG(K_i)$, where $PRNG$ stands for the pseudo-random number generator function. The same secret update procedure is also performed by the server. The protocol steps are also shown in Figure 4.4.

This protocol suffers from replay attacks between successful authentications. Moreover, a DoS attack could permanently desynchronize a server and a tag as pointed in [1]. It also does not provide backward untraceability if the fixed EPC code and the access key PIN are compromised [1].

Data Base / Reader $[EPC_i, PIN_i, K_i]$		**Tag** $[EPC_i, PIN_i, K_i]$
	Query	Generates r
Linear search through all EPC_j, K_j in database	$\longrightarrow$	Computes:
such that: $M_1 \oplus K_j = CRC(EPC_j \|\| r)$,	M_1, C, r	$M_1 = CRC(EPC_i \|\| r) \oplus K_i$
$C = CRC(M_1 \oplus r)$	$\longleftarrow$	$C = CRC(M_1 \oplus r)$
	M_2	
Computes: $M_2 = CRC(EPC_i \|\| PIN \|\| r) \oplus K_i$	$\longrightarrow$	Checks if:
Updates: $K_i = PRNG(K_i)$		$M_2 = CRC(EPC_i \|\| PIN \|\| r) \oplus K_i$
		Updates: $K_i = PRNG(K_i)$

Figure 4.4. The DPLK protocol.

4.1.5. The Chien and Chen Protocol

In [1], an RFID challenge-response protocol, conforming to the EPC Class 1 GEN-2 standards was proposed by Chien and Chen to avoid replay attacks. The protocol is referred as CC protocol due to name of the authors. One major difference of this protocol compared to predecessor examples is it stores both new and old tag keys in server database to prevent DoS attacks.

We use the same notations given in Section 4.1.4 to simplify the descriptions. Initially, the server stores $\{EPC_i, P_i, K_i, P_i^{old}, K_i^{old}\}$, where P_i and K_i current access and authentication keys. On the other hand, P_i^{old} and K_i^{old} denote the old access and authentication keys respectively. Also, each tag stores $\{EPC_i, P_i, K_i\}$ tuple. The authentication steps between the tag and the server is described as follows:

Firstly, the reader sends a random number r_1 to the tag. The tag produces another random number r_2 and computes the message $M_1 = CRC(EPC_i||r_1||r_2) \oplus K_i$. Then, the tag responds with r_1, M_1. Upon receiving r_1, M_1 through the reader, the server performs a linear search such that for each (EPC_i, K_i, K_i^{old}) tuple, the server checks that whether $M_1 \oplus K_i$ or $M_1 \oplus K_i^{old}$ is equal to $CRC(EPC_i||r_1||r_2)$. If a match is

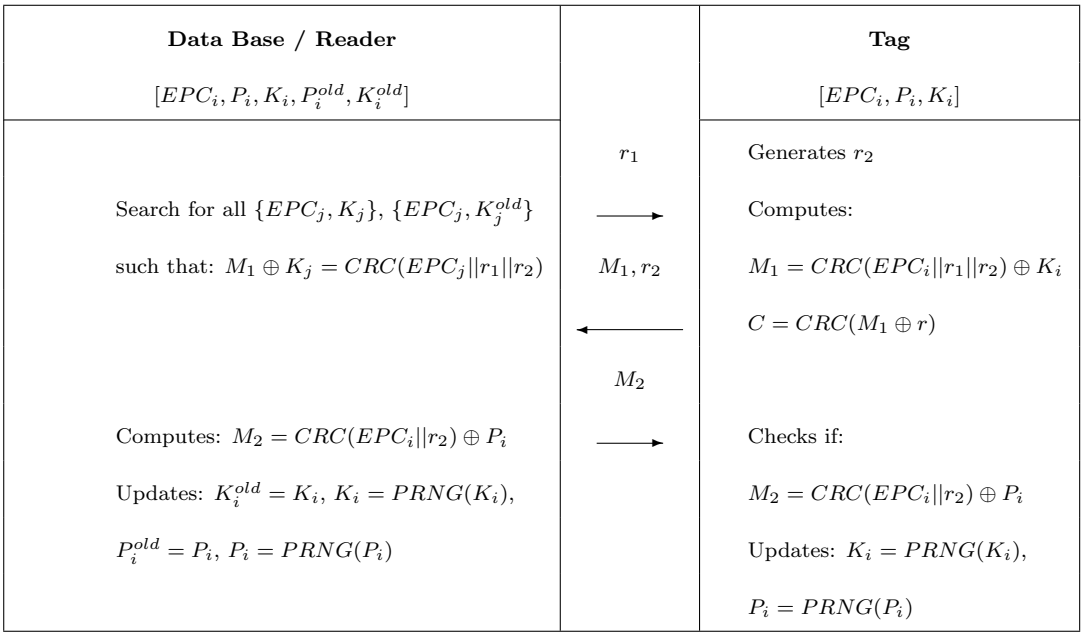

Figure 4.5. The CC protocol.

obtained, then the server authenticates the tag and calculates $M_2 = CRC(EPC_i||r_2) \oplus P_i$ or $M_2 = CRC(EPC_i||r_2) \oplus P_i^{old}$ depending on which value (K_i or K_i^{old}) satisfied equality in the previous step. Next, the server updates the secrets as $K_i^{old} = K_i$, $P_i^{old} = P_i$, $K_i = PRNG(K_i)$, $P_i = PRNG(P_i)$ and the reader delivers M_2 to the tag. If the tag verifies $M_2 = CRC(EPC_i||r_2) \oplus P_i$, then it updates its secrets as $K_i = PRNG(K_i)$, $P_i = PRNG(P_i)$. The whole authentication process is summarized in Figure 4.5.

Note that once the tag secrets are compromised by an adversary, he can identify the tag's past interactions, the fixed EPC of the tag and the tag's future transactions [5]. Hence, the protocol permits backward traceability.

4.1.6. The Song and Mitchell Protocol

Song and Mitchell (SM) proposed an RFID authentication protocol in [5]. In this scheme, a server stores secrets u_i and t_i for each tag T_i as well as the most recent secrets $\hat{u}_i$ and $\hat{t}_i$. Initially, secret u_i is a string of l bits assigned to T_i, and t_i is a hash of u_i, i.e. $t_i = h(u_i)$. A tag keeps the value of t_i as its identifier. This scheme

Data Base / Reader $[u_i, t_i, u_i^{old}, t_i^{old}]$		**Tag** $[t_i]$
	r_1	Generates r_2
Search for all $\{t_j\}$, $\{t_j^{old}\}$	$\longrightarrow$	Computes:
such that: $M_2 = f_{t_j}(r_1 \oplus (M_1 \oplus t_i))$	M_1, M_2	$M_1 = (t_i \oplus r_2)$, $M_2 = f_{t_i}(r_1 \oplus r_2)$
	$\longleftarrow$	$C = CRC(M_1 \oplus r)$
	M_3	
Computes: $M_3 = u_i \oplus (r_2 \gg l/2)$	$\longrightarrow$	Checks if:
Updates: $t_i^{old} = t_i$, $u_i^{old} = u_i$,		$h(M_3 \oplus (r_2 \gg l/2)) = t_i$
$u_i = (u_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2$,		Updates:
$t_i = h(u_i)$		$t_i = h((u_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2)$

Figure 4.6. The SM protocol.

uses a hash function h to update the secret t_i, a keyed hash function f to protect messages, and a combination of simple functions such as right and left shifts and a bit-wise exclusive-or operation to combine data strings. When an authentication session completes successfully, the server updates the secret values of (u_i, t_i) and $(\hat{u}_i, \hat{t}_i)$ for T_i, and the tag also updates its secret t_i using h. The SM protocol is summarized in Figure 4.6.

In [11], it was presented that SM protocol is vulnerable to both tag impersonation attack and reader impersonation attack, which enable an adversary to impersonate any legitimate reader or tag.

4.2. Logarithmic-Time RFID Protocols

4.2.1. The Molnar and Wagner Protocol

In order to solve the scalability problem of linear-time protocols, Molnar and Wagner proposed an RFID protocol (MW) with using a tree-based key space such that the tags hold a set of keys arranged in a tree [8]. A single particular path in the tree

is assigned for each tag, while the server knows all the secrets. To identify a tag the reader performs a challenge-response protocol through the tree from its root to the leaves looking for a match to the tag's response. This scheme requires logarithmic time complexity, so the work for the server to identify a tag has complexity of $O(\log N)$. Figure 4.7 depicts an example tree setup for the MW protocol. In this example, the number of tags is 8 and the branching factor is 2. For instance, the set of keys $\{k_{1,1}, k_{2,2}, k_{3,4}\}$ is assigned for the tag T_4 and this key set becomes its unique identifier.

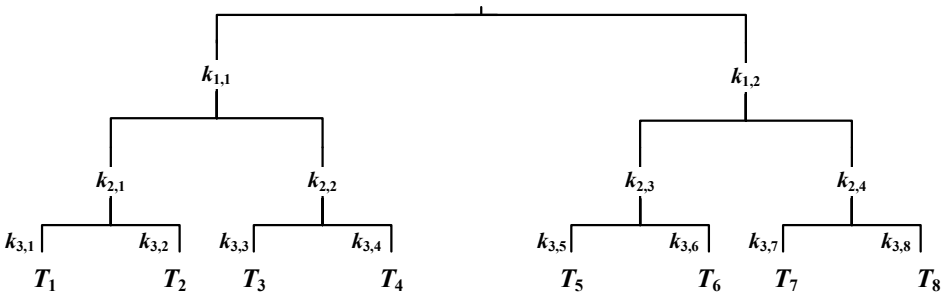

Figure 4.7. A tree-based MW system with eight tags.

Although the MW protocol reduces the computational complexity at the server side, in [48] it is shown that the protocol could degrade the privacy if one or more tags are compromised.

4.2.2. The Cheon Hong Tsudik Protocol

Inspired by the well-known "meet-in-the- middle" strategy used in the past to attack certain symmetric ciphers, Cheon et al. proposed a simple and efficient RFID tag identification and authentication technique (CHT) that reduces server computation to $O(\sqrt{N}\log N)$, where N denotes the number of tags in the system [54].

According to the protocol description, in setup phase firstly two sets of keys as $\mathcal{K}_1$, $\mathcal{K}_2$ are constructed by the server such that each has size of $\sqrt{N}$. Then, for each tag $\mathcal{T}_i$ two keys $k_1^i \in \mathcal{K}_1$, $k_2^i \in \mathcal{K}_2$ are assigned. In authentication process, the reader sends a random nonce r_1 to the tag. Next, the tag generates another random value r_2, computes $C = PRF_{k_1^i}(r_1, r_2) \oplus PRF_{k_2^i}(r_1, r_2)$ and responds with r_2, C, where PRF is a pseudo random function. In first pase of the tag identification the reader

builds a table with entries $\{k_1^i, PRF_{k_1^i}(r_1, r_2)\}$ and then sorts this table. In the second phase, the reader makes an exhaustive search such that for each $k_2^i \in \mathcal{K}_2$ it computes $C \oplus PRF_{k_2^i}(r_1, r_2)$ and searches the result in the sorted table. If a match is found, the corresponding k_1^i, k_2^i pair give the ID of tag.

However, as stated in [39], just like Molnar and Wagner's protocol, if one or more tags are compromised, the privacy of system is affected. For example, if secrets of t tags are compromised, then an adversary can identify $t^2 - t$ uncompromised tags in the system .

4.3. Constant-Time RFID Protocols

4.3.1. The Henrici and Muller Protocol

Henrici and Muller [7] proposed a scheme, named as HM, in which the server only needs to perform $O(1)$ work in tag identification process to mitigate computational load at the server side. A tag sends two hashed values as its response to a query, and updates its stored values, including its ID, after a successful authentication.

Initially, the server stores $\{ID_i, h(ID_i), k_i, k_i^{last}\}$, and the tag stores $\{ID_i, k_i, k_i^{last}\}$, where $h()$ is hash function, k_i is a transaction number and k_i^{last} denotes the number of the last successful session.

Firstly, the reader sends a query to the tag. Then, the tag increments k_i by one, computes $h(ID_i)$ and $\Delta k = k_i - k_i^{last}$. Next, the tag calculates $M_1 = h(k_i \oplus ID_i)$ and transmits the tuple $\{h(ID_i), \Delta k, M_1\}$. The $h(ID_i)$ is used to identify the tag at the server side. Also, the server uses Δk to recover k, i.e. to compute $h(k_i \oplus ID_i)\}$. Next, validity of the values are checked by the server and if they satisfy then the reader sends a random value r_1 with the message $M_2 = h(r_1 \oplus k_i \oplus ID_i)$ to the tag. Once the tag receives r_1, M_2, it checks whether $h(r_1 \oplus k_i \oplus ID_i) = M_2$. If yes, then it updates ID_i and k_i^{last} as $ID_i = ID_i \oplus r_1$ and $k_i^{last} = k_i$. The summary of the protocol is depicted in Figure 4.8.

Data Base / Reader $[ID_i, h(ID_i), k_i, k_i^{last}]$		**Tag** $[ID_i, k_i, k_i^{last}]$
	Query $\longrightarrow$	Computes: $k_i \leftarrow k_i + 1, \Delta k \leftarrow k_i - k_i^{last}$,
Find $h(ID_i)$ in database		
Calculate k_i from Δk		$M_1 = h(k_i \oplus ID_i)$
Check if: $M_1 = h(k_i \oplus ID_i)$,	$h(ID_i), \Delta k, M_1$ $\longleftarrow$	
Generates r_1	M_2	
Computes: $M_2 = h(r_1 \oplus k_i \oplus ID_i)$	$\longrightarrow$	Checks if: $M_2 = h(r_1 \oplus k_i \oplus ID_i)$
Updates: $ID_i \leftarrow ID_i \oplus r_1$, $k_i^{last} \leftarrow k_i$		Updates: $ID_i \leftarrow ID_i \oplus r_1$, $k_i^{last} \leftarrow k_i$

Figure 4.8. The HM protocol.

As stated in [1], the HM scheme is vulnerable a degree of tag tracking, because a tag always replies with the same hashed ID before it is successfully authenticated.

4.3.2. The Dimitriou Protocol

In [6], Dimitriou proposed an RFID authentication protocol which needs $O(1)$ effort for a server to authenticate a tag. According to the protocol description a tag stores ID_i, while the server stores $\{ID_i, h(ID_i)\}$ pair for each tag in the database, where $h()$ is a hash function.

The protocol steps of the protocol can be described as follows: Firstly, the reader sends a random nonce r_1 to the tag. Once receiving r_1, the tag generates another random nonce r_2. Next, it evaluates $h(ID_i)$ and $M_1 = f_{ID_i}(r_2||r_1)$, where $f()$ is a keyed hash function. Then tag sends $\{r_2, h(ID_i), M_1\}$ to the reader which then delivers these values to the back-end database. If the server finds $h(ID_i)$ in the database list, it checks whether $M_1 = f_{ID_i}(r_2||r_1)$. In case of verification, the server replaces ID_i with

Data Base / Reader $[ID_i, h(ID_i)]$		**Tag** $[ID_i]$
	r_1	Generates r_2
Search $h(ID_j)$ in the list	$\longrightarrow$	Computes: $h(ID_i)$
Checks if: $M_1 = f_{ID_j}(r_2 \| r_1)$	$r_2, h(ID_i), M_1$	$M_1 = f_{ID_i}(r_2 \| r_1)$
	$\longleftarrow$	
	M_2	
Updates: $ID_i \leftarrow g(ID_i)$	$\longrightarrow$	Computes $ID_i' = g(ID_i)$ and checks if:
Computes: $M_2 = f_{ID_i}(r_2 \| r_1)$		$M_2 = f_{ID_i'}(r_2 \| r_1)$
		If OK: $ID_i \leftarrow ID_i'$

Figure 4.9. The Dimitriou protocol.

$g(ID_i)$, where $g()$ is a hash function. Next, the server computes $M_2 = f_{ID_i}(r_2||r_1)$ and transmits it to the tag through the reader. The tag calculates $ID_i' = g(ID_i)$ and then checks $M_2 = f_{ID_i'}(r_2||r_1)$. If the equation is satisfied, the tag authenticates the reader and updates its ID as $ID_i \leftarrow ID_i'$. The steps of the protocol is shown in Figure 4.9.

One can see that the protocol is vulnerable to a tag tracking attack, since the tag ID is not changed in case of an unsuccessful authentication session [6, 32].

4.4. Ultralightweight RFID Protocols

In the previous sections of this chapter, we have considered some well known RFID protocols that are classified depending on time complexity taken by the server in tag identification. Besides the performance concern at the server side, tag computational effort required in authentication process is another crucial issue due to computational constraints mostly driven by the cost concerns of RFID tags. Therefore several ultralightweight authentication protocols, like in [16–20], have been proposed to solve the design challenge in RFID tags by using only basic bitwise and arithmetic operations such as XOR, OR and addition modulo powers of 2 rather than the bullet proven, yet computationally intensive cryptographic primitives.

In this section we review some examples of ultra-lightweight authentication protocols and shortly give their security weakness.

4.4.1. The LMAP Protocol

In [16] Peris-Lopez et al. proposed an RFID protocol which only needs bitwise XOR ($\oplus$), bitwise OR ($\vee$), bitwise AND ($\wedge$), and addition modulo powers of 2 operations. Initially, for each tag a key K^i is assigned and it is divided into four equal parts as $K^i = K1^i||K2^i||K3^i||K4^i$. Additionally, a dynamic index-pseudonym IDS^i is used and it is updated with K at the end of each successful session. Also, a static ID^i is stored at both the server and the tag side.

Firstly, the reader produces two random numbers $n1$ and $n2$. By using $K1^i$, $K2^i$, $K3^i$, $n1$ and $n2$ the reader computes the messages A, B and C and then sends them to the tag. Upon receiving the messages, the tag authenticates the reader and obtains $n1$ from A, B. Next, the tag recovers $n2$ from the message C and computes message D. Finally, tag sends D to the reader.

The authentication steps of the LMAP is depicted in Table 4.1. Also the secret updating of the scheme is given below:

$$\begin{aligned} IDS^i &= (IDS^i + (n2 \oplus K4^i)) \oplus ID^i \\ K1^i &= K1^i \oplus n2 \oplus (K3^i + ID^i) \\ K2^i &= K2^i \oplus n2 \oplus (K4^i + ID^i) \\ K3^i &= (K3^i \oplus n1) + (K1^i \oplus ID^i) \\ K4^i &= (K4^i \oplus n1) + (K2^i \oplus ID^i) \end{aligned}$$

In [55], security wakness of LMAP is shown and a desynchronization and full-disclosure attack against LMAP are presented.

Table 4.1. The protocol run of LMAP.

$\mathcal{R} \rightarrow \mathcal{T}:$	$hello$
$\mathcal{T} \rightarrow \mathcal{R}:$	IDS^i
$\mathcal{R} \rightarrow \mathcal{T}:$	$A\|\|B\|\|C$
$\mathcal{T}:$	Obtains $n1, n2$ and computes D
$\mathcal{T} \rightarrow \mathcal{R}:$	D
where:	
	$A = IDS^i \oplus K1^i \oplus n1$
	$B = (IDS^i \vee K2^i) + n1$
	$C = IDS^i + K3^i + n2$
	$D = (IDS^i + ID^i) \oplus n1 \oplus n2$

4.4.2. The SASI Protocol

SASI is an ultra-lightweight RFID protocol proposed by Chien in [17] to fix the security flaws of earlier ultra-lightweight protocols. The scheme only needs simple bitwise operations and the rotation operator $\ll$ at the tag side. According to protocol description, each tag has a static ID^i, and shares a pseudonym IDS^i and two keys $K1^i, K2^i$ with the back-end server. To prevent a possible desynchronization attack, a tag stores two entries of the form $(IDS^i, K1^i, K2^i)$: one is for the old values used in the most recent completed session, and the other is for the next values to be used in the next protocol session.

A step by step description of the SASI authentication process is given below:

- $\mathcal{R}$ sends a *hello* message to $\mathcal{T}$.
- Upon interrogation by the reader, $\mathcal{T}$ sets $IDS^i_{next} \leftarrow IDS^i$, $K1^i_{next} \leftarrow K1^i$, $K2^i_{next} \leftarrow K2^i$.
- $\mathcal{T}$ sends IDS^i to the reader.

- $\mathcal{R}$ searches for IDS^i in the database list. If a match is not found. It retransmits the *hello* message.
- If second *hello* message is received, then $\mathcal{T}$ sets $IDS^i \leftarrow IDS^i_{old}$, $K1^i \leftarrow K1^i_{old}$, $K2^i \leftarrow K2^i_{old.}$. Next, it transmits IDS^i to the reader.
- If the reader finds IDS^i in one of the two cases, then it continues the protocol steps with corresponding $K1^i, K2^i$ pair either *new* or *old*.
- $\mathcal{R}$ then uses the $K1^i, K2^i$ pair and two generated random values $n1$ and $n2$ to compute the values A, B, C, $\bar{K}1^i$ and $\bar{K}2^i$ as follows:

$$\begin{aligned} A &= IDS^i \oplus K1^i \oplus n1 \\ B &= (IDS^i \vee K2^i) + n2 \\ \bar{K}1^i &= (K1^i \oplus n2) \ll K1^i \\ \bar{K}2^i &= (K2^i \oplus n1) \ll K2^i \\ C &= (K1^i \oplus \bar{K}2^i) + (\bar{K}1^i \oplus K2^i). \end{aligned}$$

- Next, $\mathcal{R}$ sends $A||B||C$ to the tag.
- $\mathcal{T}$ firstly obtains $n1$ from A and extracts $n2$ from B. Then it computes $\bar{K}1^i$ and $\bar{K}2^i$ and checks the value of C. If the verification succeeds, then it computes the $D = (\bar{K}2^i + ID^i) \oplus ((K1^i \oplus K2^i) \vee \bar{K}1^i)$ and transmits it to $\mathcal{R}$.
- The reader locally computes D checks it with received D. If so, it updates the secrets as $IDS^i = (IDS^i + ID^i) \oplus (n2 \oplus \bar{K}1^i)$ and $K1^i = \bar{K}1^i$, $K2^i = \bar{K}2^i$.
- On the other side, $\mathcal{T}$ updates its secrets as $IDS^i_{old} = IDS^i$, $IDS^i_{next} = (IDS^i + ID^i) \oplus (n2 \oplus \bar{K}1^i)$, $K1^i_{old} = K1^i$, $K2^i_{old} = K2^i$, $K1^i_{next} = \bar{K}1^i$, $K2^i_{next} = \bar{K}2^i$.

In [56], it is shown that the SASI protocol is vulnerable to a tracking attack, so it does not achieve its design objective of untraceability.

4.4.3. The SLMAP Protocol

Some security flaws of LMAP protocol, described in Section 4.4.1, have been presented by Li and Wang in [57]. Later, the same authors introduced another ultra-lightweight RFID authentication protocol and presented it as a more stable version of

LMAP denoted by SLMAP in [19]. In order to simplify the description of SLMAP, we use the same notations of the LMAP protocol. The SLMAP protocol is briefly described in the following:

- $\mathcal{R} \rightarrow \mathcal{T}$: $hello$
- $\mathcal{T} \rightarrow \mathcal{R}$: IDS^i
- $\mathcal{R}$ checks whether IDS exists or not at the back-end server. If not; the session is terminated, otherwise it finds the corresponding $\{K1^i, K2^i, ID^i\}$ tuple and continues with the next steps.
- $\mathcal{R}$ generates a random r, then by using it the messages A and B are computed where

$$\begin{aligned} A &= IDS^i \oplus K1^i + r, \\ B &= (IDS^i + K2^i) \oplus r. \end{aligned}$$

- $\mathcal{R} \rightarrow \mathcal{T}$: $A||B$

- $\mathcal{T}$ extracts r from A and B and verifies correctness of $K1^i, K2^i$.
- If so, it prepares the message C, where $C = (IDS^i + ID^i \oplus r) \oplus (K1^i + r) \oplus (K2^i + r)$
- $\mathcal{T} \rightarrow \mathcal{R}$: C
- $\mathcal{R}$ verifies message C with its local computation. If they are equal, then $\mathcal{R}$ updates the secrets:

$$\begin{aligned} IDS^i_{next} &= (IDS^i + K1^i \oplus r + (ID^i + K2^i)), \\ K1^i &= K1^i \oplus r + (IDS^i_{next} + K2^i + ID^i), \\ K2^i &= K2^i \oplus r + (IDS^i_{next} + K1^i + ID^i). \end{aligned}$$

- $\mathcal{R}$ computes message D, where $D = IDS^i_{next} \oplus (ID^i + r) + (K1^i \oplus K2^i \oplus r)$
- $\mathcal{R} \rightarrow \mathcal{T}$: D.

In [58], a metaheuristic-based attack against the traceability of the SLMAP protocol is presented. It is shown that the SLMAP protocol does not achieve untraceability, which is one of the main aims of RFID protocols to avoid tracking attacks.

5. UNBALANCED STATES VIOLATES PRIVACY OF RFID PROTOCOLS

Recently, new security protocols like in [9, 13, 14, 47, 59], have been introduced to reduce the computational load on the back-end database. The common characteristic of these protocols is defining different tag states, so the server authenticates the tag in constant/sub-linear time in a more frequent normal state and needs a linear search in a rare abnormal states. In this chapter, we show that such protocols fail to fulfill their privacy claims, if an adversary has access to side channel information that leaks the computational complexity of the back-end database to authenticate a tag.

5.1. Privacy Definitions

Privacy, both in terms of tag anonymity and tag untraceability is a significant concern that needs to be addressed if RFIDs are to be as widely deployed as conceived by proponents [60]. As mentioned in Section 3.1.1, untraceability issue has been treated formally in different security models like in [43–45]. In this section, instead of giving whole detailed definitions we briefly describe here the basic ideas of Vaudenay's privacy model [44] and its an extended version the Avoine's privacy model [61] which will be sufficient to describe our security analysis. The Juels–Weis privacy model is based on indistinguishability of the tags and indeed this makes it more practical. Nevertheless, Vaudenay's model is more flexible by allowing adversaries with variant degrees of capabilities. Thus, our privacy analysis delicately combines definitions of these two models and defines untraceability in terms of a privacy experiment by which an adversary could distinguish two different tags by using all accessible oracles. According to Vaudenay's model, an adversary $\mathcal{A}$ takes a public key K_P as input and can run the following oracles:

- $O^{\text{CreateTag}}(ID, b)$: This oracle creates a free tag, either legitimate ($b = 1$) or not ($b = 0$), with unique identifier ID. By convention, b is implicitly 1 when omitted.

- $O^{\text{DrawTag}}(distr) \rightarrow (t_1, b_1, \cdots, t_k, b_k)$: This oracle moves from the set of free tags to the set of drawn tags a tuple of k tags at random following the probability distribution $distr$. For each chosen tag, the oracle gives it a new pseudonym denoted t_i and changes its status from free to drawn. Finally, the oracle returns all the generated pseudonyms $(t_1, b_1, \cdots, t_k, b_k)$ in any order.
- $O^{\text{Free}}(t)$: This oracle moves the tag with pseudonym t from the status drawn to the status free. It makes this tag unreachable for $\mathcal{A}$.
- $O^{\text{Launch}} \rightarrow \pi$: This oracle makes the reader launch a new protocol instance π which is returned as the output.
- $O^{\text{SendReader}}(m, \pi) \rightarrow r$: This oracle sends a message m to the reader $\mathcal{R}$ for a protocol instance π. It outputs the response r from the reader.
- $O^{\text{SendTag}}(m, t) \rightarrow r$: This oracle sends a message m to the tag $\mathcal{T}$ with pseudonym t. It outputs the response r from $\mathcal{T}$.
- $O^{\text{Execute}}(t) \rightarrow (\pi, transcript)$: This oracle executes a complete protocol between $\mathcal{R}$ and $\mathcal{T}$ with pseudonym t. It returns the transcript of the protocol, i.e. the list of successive protocol messages
- $O^{\text{Return}}(\pi) \rightarrow x$: When π is completed this oracle outputs $x = 1$ if the output of the reader is $\neq \perp$, and $x = 0$ otherwise.
- $O^{\text{Corrupt}}(t) \rightarrow tk_t$: This oracle returns the tag-dependent key tk_t of tag t. If t is no longer used after this oracle call, we say that t is destroyed.

In addition to these oracles, we describe two new oracles[1]

$O^{\text{Complexity}}(\pi, \alpha) \rightarrow \delta$: This oracle returns computational complexity performed by back-end database for its overall computations during the protocol instance π as δ in units of α, e.g. α may be relevant to time, power consumption, electromagnetic leaks, sound emission etc.

[1]In Avoine's privacy model $O^{\text{Timer}}(\pi)$ oracle is defined. It outputs the time δ taken by the reader for its overall computations in protocol instance π. We extend functionality of this oracle for our needs as $O^{\text{Complexity}}(\pi, \alpha)$ which may give side channel information in disparate domains depending on α such as power consumption, sound, time etc.

$O^{\text{SetState}}(S,t)$: This oracle sets state of the tag with pseudonym t to S.

Definition 5.1. *Suppose that for an RFID protocol a tag $\mathcal{T}$ is allowed to be in one of the states of the finite set $\mathbf{S}_{\mathcal{T}} = \{S_0, S_1, \cdots, S_k\}$, where $k \geq 1$, through interaction with the RFID readers. Then this protocol is called Unbalanced Authentication Protocol (UAP) and the states are named as unbalanced states, if there exists any two states $S_i, S_j \in \mathbf{S}_{\mathcal{T}}$ for which the server performs identification of the tag in different order of computational complexities.*

Definition 5.2. *Let $S_i, S_j \in \mathbf{S}_{\mathcal{T}}$ be two unbalanced states of an UAP. An adversary $\mathcal{A}$ having the ability to set a tag's state from S_i to S_j or vice versa is denoted by $\mathcal{A}^{Stat}$ and called a stateful–adversary.*

As can be inferred from the above definition, a stateful–adversary $\mathcal{A}^{Stat}$ has access to the oracle $O^{\text{SetState}}(S,t)$.

5.2. Previous Work

Use of side channel information – e.g. computational time – in security analysis of RFID authentication protocols has been recognized in a number of reported studies and particularly in the following ones. In [45], Juels and Weis introduced the idea that adversary may access to the protocol output which shows whether a reader succeeded to identify a legitimate tag or not. For instance, opening a door with a proximity card or acceptance of a payment card can give this information. Of course such an information gives a hint to an adversary to distinguish two different tags, i.e. it breaks the privacy of the protocol. Moreover, they mentioned that computation time of the reader can shed critical light on protocol design and showed O–TRAP protocol, described in [62], cannot provide strong privacy if this side channel information is used. The idea that the adversary knows whether a reader succeeded to identify a legitimate tag or not is also formalized in Vaudenay's privacy model [44]. In [62], timing attacks have been briefly considered by Burmester et al. and the following has been claimed: "In particular the time taken for each pass must be constant. This can be done by inserting an artificial

delay on the trusted server...". Also, in [61] a privacy model extending the Vaudenay's one has formalized computational time of the reader. A new privacy level TIMEFUL which is determined by leaked information from the computational time of the reader is added to privacy levels of model in [44].

In this section, within a formal model we show that UAPs suffer from a side-channel attack which breaks privacy of the schemes. As a consequence, we stress that use of unbalanced states must be avoided in designing RFID authentication protocols.

5.3. Description of the Proposed Attack

Before we elaborate on our attacking strategies, we will first state our assumption that an adversary may access to all previously mentioned oracles.

Theorem 5.3. *Any RFID authentication protocol involving unbalanced states are traceable to a stateful–adversary.*

Proof. To prove the statement we describe the following privacy experiment $\mathbf{Exp}^{priv}_{UAP}$:

i) $\mathcal{A}^{Stat}$ creates two legitimate tags using twice $O^{\text{CreateTag}}(ID)$ and calls the oracle $O^{\text{DrawTag}}(\frac{1}{2}, 2)$. So, he obtains two pseudonyms t_0 and t_1.

ii) $\mathcal{A}^{Stat}$ puts the tag t_0 into state S_i and t_1 into state S_j by calling $O^{\text{SetState}}(S_i, t_0)$ and $O^{\text{SetState}}(S_j, t_1)$ respectively.

iii) $\mathcal{A}^{Stat}$ gets $(\pi_0, transcript_0)$ and $(\pi_1, transcript_1)$ by calling the oracles $O^{\text{Execute}}(t_0)$ and $O^{\text{Execute}}(t_1)$ respectively.

iv) $\mathcal{A}^{Stat}$ requests the computational complexities of each authentication by using $O^{\text{Complexity}}(\pi_0, \alpha)$ and $O^{\text{Complexity}}(\pi_1, \alpha)$ to obtain δ_0 and δ_1.

v) Again $\mathcal{A}^{Stat}$ puts the tag t_0 into state S_i and t_1 into state S_j with requesting $O^{\text{SetState}}(S_i, t_0)$ and $O^{\text{SetState}}(S_j, t_1)$.

vi) $\mathcal{A}^{Stat}$ frees both of the tags by calling $O^{\text{Free}}(t_0)$ and $O^{\text{Free}}(t_1)$ and reaffects only one of them with $O^{\text{DrawTag}}(\frac{1}{2}, 1)$. The adversary gets a new pseudonym t^*_b.

vii) $\mathcal{A}^{Stat}$ executes an instance protocol by calling $O^{\text{EXECUTE}}(t_b^*) \rightarrow (\pi^*, transcript^*)$ and gets δ^* with requesting $O^{\text{COMPLEXITY}}(\pi^*, \alpha)$.

viii) If $\delta^* = \delta_0$ $\mathcal{A}$ guesses $b = 0$ and decides $t_b^* = t_0$, otherwise guesses $t_b^* = t_1$.

Clearly, the success probability of the adversary in guessing b is 1, because δ_0 and δ_1 are outputs of different order of computational complexities and they are comparably different. Thus, a stateful–adversary can distinguish between two different tags, i.e. breaks privacy of the system. □

This theorem shows an impossibility for obtaining any form of untraceability for an RFID system as long as it posses unbalanced states that can settled by an adversary. Notice that as can be inferred from the above privacy experiment, a stateful–adversary can trace a tag without requesting the oracle $O^{\text{CORRUPT}}(t)$.

5.4. The Proposed Attack against Some Unbalanced Authentication Protocols

In this section, we apply our privacy experiment described in Section 5.3 to reveal traceability in some RFID schemes that have unbalanced states in their protocol descriptions. Then we point out that all of these protocols suffer from the presented side-channel attack and they fail to fulfill their untraceability claims.

5.4.1. The Scalable Song–Mitchell Protocol and Its Security Analysis

In [13], Song and Mitchell proposed an RFID protocol, referred to here as the SSM protocol, claims to assure the untraceability, authentication, and robustness against replay and spoofing attacks. SSM assumes that the channel between the reader and the back-end server is secure, hence $\mathcal{DB}$ is considered as a single entity consists of the back-end server and the reader. Initially, a secret l-bit string, s_j, and its hash

value (computed by the server), $k_j = H(s_j)$, is attached to the tag entries $\mathcal{T}_j \in \mathcal{DB}$. Moreover, for every tag $\mathcal{T}_j$, $\mathcal{DB}$ stores a hash-chain $\{x_0, x_1, \cdots, x_m\}$ where m is a positive integer, $x_i = e_k(x_{i-1})$ for $1 \leq i \leq m$ and x_0 is a random l-bit string. In case of a need to resynchronize the tag, $\mathcal{DB}$ further stores $\{\hat{s}_j, \hat{k}_j\}$ as the most recent secrets assigned to $\mathcal{T}_j$. On the other hand, each $\mathcal{T}_j$ stores k, x and x_m, where x is initially set to x_0.

A summary of SSM protocol is seen in Figure 5.1, where $e(), f(), g()$ are keyed one-way hash functions. Notice that on the condition whether x is equal to x_m or not, SSM puts the tags in two states.

In a normal state which is the case $x \neq x_m$ whenever $\mathcal{T}$ receives r_R, the tag calculates $M_T = f_k(r_R||x)$ and updates its identifier x with $e_k(x)$. $\mathcal{T}$ transmits r_R, x and M_T to $\mathcal{DB}$. If the updated x is equal to x_m, $\mathcal{T}$ waits for $\mathcal{DB}$ response, keeping r_R and M_T in short term memory.

On the other side, when $\mathcal{DB}$ receives $\{r_R, x, M_T\}$ payload, firstly, $\mathcal{DB}$ searches its look-up table for a value x_i matching to the received x. If nothing matches, the session terminates. Otherwise, $\mathcal{DB}$ identifies the tag and checks whether $f_k(r_R||x_{i-1})$ equals the received M_T, where k is the identified tag $\mathcal{T}$'s key. If this verification succeeds and $x \neq x_m$, then the authentication session terminates successfully. However, if $x = x_m$, then the server starts a regular secret update process in which $\mathcal{DB}$ picks a random l-bit string s' and an integer m', and computes a key $k' = H(s')$ and a sequence of m' identifiers $x'_i = e_{k'}(x'_{i-1})$ for $1 \leq i \leq m'$, where x'_0 is set to x. Then, $\mathcal{DB}$ computes $M_S = g_k(r_R||x||M_T) \oplus (s||k'||x'_{m'})$, and sends r_R and M_S to $\mathcal{T}$. Finally, $\mathcal{DB}$ updates the set of stored values for $\mathcal{T}$ from $(\hat{s}, \hat{k}, s, k, x_0, x_1, \cdots, x_m)$ to $(s, k, s', k', x, x'_1, \cdots, x'_{m'})$. Notice that, in the above process server authenticates the tag in constant time by simply running through the database.

In the normal state, if the tag $\mathcal{T}$ receives $\{r_R, M_S\}$ payload, $(s||k'||x'_{m'}) = M_S \oplus g_k(r_R||x||M_T)$ is evaluated. Moreover, $\mathcal{T}$ authenticates $\mathcal{DB}$ and updates k and x_m with

Data Base / Reader $[\hat{s}, \hat{k}, s, k, x_0, \cdots, x_i \cdots, x_m]$		**Tag** $[k, x, x_m]$
Generate r_R	r_R $\longrightarrow$	**If** $x \neq x_m$, $M_T = f_k(r_R \| x)$ $x \leftarrow e_k(x)$
	r_R, x, M_T $\longleftarrow$	
Case 1: Search for $x_i = x$ in the DB Check $M_T = f_k(r_R \| x_{i-1})$ **Case 2:** **If** $x = x_m$, $M_S = g_k(r_R \| x \| M_T) \oplus (s \| k' \| x'_{m'})$ Update secrets for Tag $\hat{s} \leftarrow s, \hat{k} \leftarrow k, s \leftarrow s', k \leftarrow k', x_0 \leftarrow x$ $x_i (1 \leq i \leq m) \leftarrow x'_i (1 \leq i \leq m')$	r_R, M_S $\longrightarrow$	$(s \| k' \| x'_{m'}) = M_S \oplus g_k(r_R \| x \| M_T)$ **If** $H(s) = k$, $k \leftarrow k'$ and $x_m \leftarrow x'_{m'}$
Case 3: Search for $x = x_m$ or x_0 for which $M_1 = f_k(r_R \| (M_2 \oplus x))$ $r_T = M_2 \oplus x$ **If** $x = x_m$, $M_S = g_k(r_R \| r_T \| M_1) \oplus (s \| k' \| x'_{m'})$ **If** $x = x_0$, $M_S = g_{\hat{k}}(r_R \| r_T \| M_1) \oplus (\hat{s} \| k \| x_m)$ Update secrets for Tag $s \leftarrow s', k \leftarrow k', x_0 \leftarrow x$ $x_i (1 \leq i \leq m) \leftarrow x'_i (1 \leq i \leq m')$	r_R, M_1, M_2 $\longleftarrow$ *SecReq* r_R, M_S $\longrightarrow$	**If** $x = x_m$, Generate r_T $M_1 = f_k(r_R \| r_T)$ $M_2 = (r_T \oplus x)$ $(s \| k' \| x'_{m'}) = M_S \oplus g_k(r_R \| r_T \| M_1)$ **If** $H(s) = k$, $k \leftarrow k'$ and $x_m \leftarrow x'_{m'}$

Figure 5.1. The SSM protocol.

k' and $x'_{m'}$ respectively, if $H(s)$ is equal to k. The secret update session is then ended successfully.

On the other hand, the process is slightly different if the tag is in a rare abnormal state which is the case $x = x_m$ when $\mathcal{T}$ receives the nonce r_R. The process in this state which is also illustrated in the lower box of Figure 5.1 starts with the generation of a random number r_T and values $M_1 = f_k(r_R||r_T)$ and $M_2 = r_T \oplus x$. The tag, then sends r_R, M_1 and M_2 back to $\mathcal{DB}$ with a request for an update of the shared secrets as $SecReq$ and waits for the server response, keeping r_R, r_T and M_1 in its short term memory.

On the server side, once $\mathcal{DB}$ receives the $\{r_R,\ M_1,\ M_2, SecReq\}$ payload, the server starts an irregular secret update process. Firstly, $\mathcal{DB}$ searches its look-up table for a value $x = x_m$ or $x = x_0$ for which $M_1 = f_k(r_R||(M_2 \oplus x))$. Observe that this search corresponds to the usual linear search and particularly carried in the abnormal state. If such a value is found, $\mathcal{DB}$ authenticates the tag. Otherwise, the session terminates.

In this step, $x = x_m$ means that even if $\mathcal{T}$ sent $x = x_m$ to $\mathcal{DB}$ in the previous session, $\mathcal{DB}$ would not receive it correctly, hence, the shared secrets are not fresh. In this case, $\mathcal{DB}$ performs the following steps:

- $\mathcal{DB}$ chooses a random l-bit string s' and an integer m', and computes a key $k' = H(s')$ and a sequence of m' identifiers $x'_i = e_{k'}(x'_{i-1})$ for $1 \leq i \leq m'$, where x'_0 is set to x.
- $\mathcal{DB}$ computes $r_T = M_2 \oplus x$ and $M_S = g_k(r_R||r_T||M_1) \oplus (s||k'||x'_{m'})$ and sends r_R and M_S to $\mathcal{T}$.
- $\mathcal{DB}$ assigns the set of values $\{s, k, s', k', x, x'_1, \cdots, x'_{m'}\}$ to the tag $\mathcal{T}$, updating the previously stored values $\{\hat{s}, \hat{k}, s, k, x_0, x_1, \cdots, x_m\}$.

However, if $x = x_0$ it means that M_S did not reach $\mathcal{T}$ correctly in the previous session, and thus $\mathcal{T}$ could not update its secrets, although $\mathcal{DB}$ did. In this case,

the following steps are performed by the server. $\mathcal{DB}$ computes $r_T = M_2 \oplus x$ and $M_S = g_{\hat{k}}(r_R||r_T||M_1) \oplus (\hat{s}||k||x_m)$, and sends r_R and M_S to $\mathcal{T}$

On the tag side, since M_1 and r_T were kept previously, when $\mathcal{T}$ receives r_R and M_S, it calculates $(s||k'||x'_{m'}) = M_S \oplus g_k(r_R||r_T||M_1)$. Following this step, if $H(s)$ is equal to k, $\mathcal{T}$ authenticates $\mathcal{DB}$ and terminates successfully after and updating k and x_m to k' and $x'_{m'}$ respectively.

Proposition 5.4. *SSM protocol is vulnerable to the proposed attack and cannot provide untraceability.*

Proof. The statement is confirmed, if we show that the SSM protocol is an UAP and a stateful–adversary $\mathcal{A}^{Stat}$ exists for this system. As given in the protocol description the server authenticates a registered tag in $O(1)$ if $x \neq x_m$; we call this state as S_0. On the other hand, the server performs a linear search costing $O(N)$ in case of $x = x_m$ and this state is represented as S_1. Note that two unbalanced states as S_0 and S_1 exist for SSM i.e. it is an UAP. Furthermore, an adversary can put any tag into state S_0 or S_1 by running the following procedures respectively. □

Procedure 5.5. *Setstate* S_0

i) *For a selected tag with pseudonym* t_i, $\mathcal{A}$ *transmits some random nonce* r *to* $\mathcal{T}_i$ *by calling* $O^{\text{SEND TAG}}(r_R, t_i)$.

ii) $\mathcal{A}$ *repeats the previous step until* $\mathcal{T}_i$ *response contains SecReq, indicating a secret update request.*

iii) $\mathcal{A}$ *calls the oracle* $O^{\text{EXECUTE}}(t_i)$.

By using the above procedure, the tag updates its secrets and it is assured that in the next query of the tag it would not wait for a secret update, since $m \geq 1$ i.e. $x \neq x_m$. Hence, it is guaranteed that the tag is in state S_0.

Procedure 5.6. *Setstate* S_1

i) For a selected tag with pseudonym t_i, $\mathcal{A}$ *transmits some random nonce* r_R *to* $\mathcal{T}_i$ *by calling* $O^{\text{SEND TAG}}(r_R, t_i)$.

ii) $\mathcal{A}$ *repeats the previous step until* $\mathcal{T}_i$ *response contains SecReq, demanding a secret update.*

After execution of *Setstate* S_1 procedure the tag will be in state S_1, because $x = x_m$. Thus, in the next protocol run it requests a secret update which imposes a linear search at the server side. Notice that the adversary is able to put any selected tag into one of the unbalanced states. Therefore, the adversary is a stateful–adversary and can request the oracle $O^{\text{SETSTATE}}(S, t)$. From Theorem 5.3, it is apparent that SSM protocol is vulnerable to the proposed attack and cannot provide untraceability.

5.4.2. The LRMAP Protocol and and Its Security Analysis

LRMAP is a mutual authentication protocol proposed by Ha et al. [14]. Similar to SSM, LRMAP reduces the computational load on the back-end database by putting the tags in two different states. In a (normal) synchronized state the protocol requires only 3 hash operations on the server side even number of tags in the system is large. However; in the case of an (unusual) desynchronization state, the recovery time on average is $N + 3$ hash operations, where N denotes the number of tags.

A brief description of LRMAP protocol is seen in Figure 5.2. Note that the back-end database $\mathcal{DB}$ manages the ID, hashed values HID, and PID records assigned to each tag $\mathcal{T}$. According to the state of the tag, the $\mathcal{DB}$ finds the ID for the current session or PID used for the previous session by comparing the received message with the HID and PID.

The state of the tag is determined by the value of $SYNC$ variable. In a normal state, the tag's variable $SYNC = 0$ and the tag's response to a challenge of $\mathcal{R}$ is simply sending $P = H(ID)$, $L(Q)$ and r_T after setting $SYNC = 1$, where $Q = H(ID||r_T||r_R)$.

Data Base / Reader $[ID, HID, PID]$		**Tag** $[ID, SYNC]$
Generate r_R	r_R →	Generate r_T **If** $SYNC = 0$, $P = H(ID)$ **else** $P = H(ID\|\|r_T)$ $Q = H(ID\|\|r_T\|\|r_R)$ $SYNC = 1$
	$P, L(Q), r_T$ ←	
If $P = H(ID)$, $PID = ID$ **else if** $P = H(ID\|\|r_T)$, $PID = ID$ **else if** $P = H(PID\|\|r_T)$, $PID = PID$ $Q' = H(PID\|\|r_T\|\|r_R)$ **If** $L(Q') = L(Q)$, $ID = H(PID\|\|r_R)$, $HID = H(ID)$	$R(Q')$ →	
		If $R(Q') = R(Q)$ $ID = H(ID\|\|r_R)$ $SYNC = 0$

Figure 5.2. The LRMAP protocol.

On the server side, $\mathcal{DB}$ searches P with the HID values stored in the database. If the values match, $\mathcal{DB}$ regards the ID as the identity of $\mathcal{T}$ and states $PID = ID$, then, $\mathcal{DB}$ calculates $Q' = H(PID||r_T||r_R)$ and checks whether $L(Q')$ is equal to $L(Q)$ or not . If it holds, $\mathcal{DB}$ sends $R(Q')$ to $\mathcal{R}$ and sets $ID = H(PID||r_R)$ and $HID = H(ID)$. $\mathcal{R}$ forwards received $R(Q')$ to $\mathcal{T}$. If $R(Q') = R(Q)$, then $\mathcal{T}$ updates its ID as $ID = H(ID||r_R)$ and sets $SYNC = 0$. In fact, this explains how the LRMAP protocol behaves in a normal state, where no uncertainty exists.

The abnormal state of LRMAP is set when $SYNC = 1$ where the tag's response to a challenge is sending $P = H(ID||r_T)$, $L(Q)$ and r_T after setting $SYNC = 1$, where $Q = H(ID||r_T||r_R)$. Naturally, $\mathcal{DB}$ would not be able to find the P in the HID values stored in the database, hence, it has to go through a linear search for a match $P = H(ID||r_T)$. If such an effort fails, $\mathcal{DB}$ retries a third search that computes $H(PID||r_T)$ and compares it with P. $\mathcal{DB}$ gets a match in the third search process when $\mathcal{R}$'s last messages were blocked in the previous session, that is, S$YNC = 1$ and $\mathcal{DB}$ updated the ID, but ID of the tag was not updated. If $\mathcal{DB}$ finds a match in these search cases, PID is set to PID for the third search and set to ID otherwise, then, $\mathcal{DB}$ calculates $Q' = H(PID||r_T||r_R)$ and checks whether $L(Q')$ is equal to $L(Q)$ or not. If it holds, $\mathcal{DB}$ sends $R(Q')$ to $\mathcal{R}$ and sets $ID = H(PID||r_R)$ and $HID = H(ID)$. $\mathcal{R}$ forwards received $R(Q')$ to $\mathcal{T}$. The tag's process to this last message is exactly the same as in the normal state.

Proposition 5.7. *The LRMAP protocol suffers the proposed attack and cannot achieve untraceability.*

Proof. To prove the statement we have to show that the LRMAP protocol is an UAP and a stateful–adversary $\mathcal{A}^{Stat}$ exists for this system. According to the protocol description the server authenticates a registered tag in $O(1)$ if $SYNC = 0$; we call this state as S_0. On the other hand, the server performs a linear search requiring $O(N)$ in case of $SYNC = 1$ and this state is denoted as S_1. Notice that the state of the tag is determined by the value of $SYNC$ variable and two unbalanced states as S_0 and S_1

exist. Thus, the protocol is an UAP. Also, an adversary has ability to set $SYNC$ value. In other words, he can put any tag to state S_0 or S_1 as follows: By using the oracle $O^{\text{Execute}}(t_i)$ for a selected tag with pseudonym t_i, $\mathcal{A}$ puts the tag state into S_0, because at the end of a successful authentication the tag sets $SYNC = 0$. On the other hand if the adversary transmits some random nonce r_R to for a selected tag with pseudonym t_i and breaks the protocol, the tag will be in state S_1. The reason is obvious the protocol is not completed and $SYNC = 1$ value is kept by the tag.

By considering these facts one can conclude that the adversary is able to put any selected tag into one of the unbalanced states. Hence, the adversary is a stateful–adversary and can call the oracle $O^{\text{SetState}}(S, t)$. From Theorem 5.3, it is obvious that the Ha's protocol cannot provide untraceability. □

5.4.3. The Chang–Wu Protocol and Its Security Analysis

The protocol, referred to here as the CW protocol, proposed by Chang and Wu [59] relies on a hybrid scheme from the randomized hash lock scheme of Weis et al. [52] and Dimitriou's mutual authentication scheme [6]. Initially, a random secret, $A_{i,t}$ shared both in a particular tag and the server is generated, where $A_{i,t}$ denotes the secret of $\mathcal{T}_i$ at instance t. Let $\alpha_1, \ldots, \alpha_m$ be the enumerate of all possible random strings α of length $\log m$. To be efficient to find the ID of a tag, the indexes $H(\alpha_k || A_{i,t})$ is stored $\forall k \in \{1, \ldots, m\}$.

According to the protocol description, $\mathcal{R}$ is considered as a single entity consists of the back-end server and the reader. The authentication process of the CW protocol goes as follows:

- $\mathcal{R}$ sends a long random string r_R to a tag $\mathcal{T}_i$.
- If the counter C_i is less than some threshold value δ, $\mathcal{T}_i$ generates a short random string α and a long random string r_T. Then it computes two hash values $M_1 =$

$H(\alpha||A_{i,t})$ and $M_2 = H(A_{i,t}||r_T||r_R)$. Next, the tag transmits $\langle M_1, M_2, r_T, \alpha\rangle$ to the reader.

- The $\mathcal{R}$ uses M_1 to quickly search for the index in the database and find out ID of the tag in constant time. If a match is found in the list, then the reader checks $M_2 = H(A_{i,t}||r_T||r_R)$. If the equation is verified, $\mathcal{R}$ replies to $\mathcal{T}_i$ with $M_3 = H(A_{i,t}||r_T)$ and updates $A_{i,t}$ to $A_{i,t+1}$.
- Upon reception of M_3, the tag checks whether it is a true value by computing $H(A_{i,t}||r_T)$. If yes, then it allows the reader to use its all functionality, sets $C_i = 0$ and updates $A_{i,t}$ to $A_{i,t+1}$. If not, however, the tag rejects and records this illegitimate query by incrementing the counter C_i.
- On the other hand, in case of $C_i > \delta$, the tag starts to run the fully randomized hash lock protocol [52] with readers. This mode of moderation requires a linear search with $O(N)$ work.

Proposition 5.8. *The CW protocol is vulnerable the proposed attack and cannot ensure untraceability.*

Proof. If we show that the CW protocol is an UAP and a stateful–adversary $\mathcal{A}^{Stat}$ exists for this scheme, then we prove the statement. The protocol is run depending on two different modes: If $C_i \leq \delta$ at the tag side, the server authenticates tag in sub-linear time via searching the value in an indexed list. However, if $C_i > \delta$, then the tag executes a different protocol process which results in a linear search at the server side in tag authentication process. Note that the state of the tag is depends on the counter value C_i, so two unbalanced states as S_0 and S_1 exist for the cases whether C_i is less than or greater than δ respectively. Thus, the protocol is an UAP. Also, an adversary has ability to put a tag into one of two states: By requesting the oracle $O^{\text{Execute}}(t_i)$ for a selected tag with pseudonym t_i, $\mathcal{A}$ puts the tag state into S_0, because at the end of a successful authentication the tag sets $C_i = 0$. On the side if the adversary queries a tag more than δ times, the tag will be in state S_1 due to C_i exceeding δ. Therefore, the adversary is a stateful–adversary and can request the

oracle $O^{\textsc{SetState}}(S, t)$. Considering Theorem 5.3, we can say that the CW protocol cannot provide untraceability. □

6. SECURITY ANALYSIS OF RECENT RFID PROTOCOLS

In this chapter, we exploit security flaws of some recently proposed RFID authentication protocols that have received no attacks yet. Each section involves a description of the target protocol and details of proposed attack (or attacks).

6.1. Security Flaws in the YL Protocol

As discussed previously, a large number of RFID protocols have been proposed to reduce the computational load on the back-end database. One such protocol which we will call YL due to name of the author was introduced by Yanfei Liu in [15]. The author provides a detailed security analysis of the protocol and claims that YL achieves a list of security properties, including resistance to tracking, tag impersonating, replay, denial of service and compromising attacks.

In this section, we examine the security of the YL scheme and show that YL mutual authentication protocol is vulnerable to the tag tracking, tag impersonation and desynchronization attacks. After analyzing the weaknesses of the protocol, we propose our revisions to remedy the vulnerabilities.

6.1.1. Description of the YL protocol

In initiation for an RFID tag $\mathcal{T}_i$ a random number x_i is generated by $\mathcal{DB}$ which computes the secret key of $\mathcal{T}_i$ as $y_i = f_k(x_i)$, where k is the secret key of $\mathcal{DB}$. Then, x_i, y_i are assigned to $\mathcal{T}_i$ which stores them with hashing as $h(k)$. On the other hand, $\mathcal{DB}$ stores k and $\{h(k), y_i, y_i^{old}\}$ tuple with related information for each tag $\mathcal{T}_i$. Initially, $y_i = y_i^{old}$.

A step by step description of the YL authentication is given below, also a summary of the protocol is depicted in Figure 6.1:

i) $\mathcal{R}$ generates a random nonce r_1 and sends it to $\mathcal{T}_i$ as a query.

ii) $\mathcal{T}_i$ generates another random nonce r_2 and computes $M_1 = x_i \oplus h(h(k) \oplus r_2)$, $M_2 = h(y_i \oplus r_1 \oplus r_2)$.

iii) $\mathcal{T}_i$ sends $\{r_2, M_1, M_2\}$ to $\mathcal{R}$.

iv) $\mathcal{R}$ transmits $\{r_1, r_2, M_1, M_2\}$ to $\mathcal{DB}$.

v) $\mathcal{DB}$ computes $x_i = M_1 \oplus h(h(k) \oplus r_2)$ and $y_i = f_k(x_i)$. Then checks whether $M_2 = h(y_i \oplus r_1 \oplus r_2)$. If it holds, $\mathcal{DB}$ authenticates $\mathcal{T}_i$. Otherwise it sends an error to $\mathcal{R}$ and terminates the session.

vi) $\mathcal{DB}$ computes $x_i^* = h(x_i \oplus y_i \oplus r_1 \oplus r_2)$ and new key $y_i^* = f_k(x_i^*)$.

vii) $\mathcal{DB}$ calculates $M_3 = y_i^* \oplus h(x_i^* \oplus y_i)$, $M_4 = h(x_i^* \oplus y_i^*)$, sends $\{M_3, M_4\}$ with related data of $\mathcal{T}_i$ to $\mathcal{R}$ and sets $y_i^{old} \leftarrow y_i$, $y_i \leftarrow y_i^*$.

viii) $\mathcal{R}$ forwards M_3, M_4 to $\mathcal{T}_i$.

ix) $\mathcal{T}_i$ calculates $x_i^* = h(x_i \oplus y_i \oplus r_1 \oplus r_2)$ and obtains $y_i^* = M_3 \oplus h(x_i^* \oplus y_i)$. After it checks whether $M_4 = h(x_i^* \oplus y_i^*)$. If the check succeeds $\mathcal{T}_i$ authenticates the server and sets: $x_i \leftarrow x_i^*$, $y_i \leftarrow y_i^*$. Otherwise keeps x_i, y_i unchanged.

6.1.2. Security Claims of the YL Protocol

YL protocol is asserted to have a list of security properties. In this part, we summarize those that are related to our attacks as follows:

Tag Impersonation: Under this attack, the adversary usually first queries $\mathcal{T}_i$ and obtains a response, and then places the response on fake RFID tags. In YL protocol, $\mathcal{T}_i$ responds with M_1 and M_2, which are decided by a random nonce r_1 from the $\mathcal{R}$. Since r_1 is different in every query, M_1 and M_2 could not be computed correctly by the fake RFID tags, i.e. they could not be authenticated by $\mathcal{DB}$.

Tag Tracking: The response of a tag $\mathcal{T}_i$ is anonymous, since r_1, r_2 are random values, and x_i, y_i are also changed after every successful authentication in YL protocol. The eavesdropper could not distinguish one tags responses from anothers. Thus it is difficult to track the location of a tag. Furthermore, since y_i, is changed by random numbers

Data Base / Reader $[k, h(k), y_i, y_i^{old}]$		**Tag** $[h(k), x_i, y_i]$
Generates r_1	r_1 →	Generates r_2 and computes: $M_1 = x_i \oplus h(h(k) \oplus r_2)$, $M_2 = h(y_i \oplus r_1 \oplus r_2)$
	← $\{r_2, M_1, M_2\}$	
Computes: $x_i = M_1 \oplus h(h(k) \oplus r_2)$, $y_i = f_k(x_i)$ **If** $M_2 = h(y_i \oplus r_2 \oplus r_1)$, computes: $x_i^* = h(x_i \oplus y_i \oplus r_1 \oplus r_2)$ $y_i^* = f_k(x_i^*)$, $M_3 = y_i^* \oplus h(x_i^* \oplus y_i)$ $M_4 = h(x_i^* \oplus y_i^*)$ $y_i^{old} \leftarrow y_i$, $y_i \leftarrow y_i^*$	M_3, M_4 →	Computes: $x_i^* = h(x_i \oplus y_i \oplus r_1 \oplus r_2)$, $y_i^* = M_3 \oplus h(x_i^* \oplus y_i)$ **If** $M_4 = h(x_i^* \oplus y_i^*)$ $x_i \leftarrow x_i^*$, $y_i \leftarrow y_i^*$

Figure 6.1. The YL protocol.

r_1, r_2 and one-way keyed hash function $f_k()$, the adversary cannot find the relationship between the forward key and the backward key. Thereby, backward traceability and forward traceability are also not extent in the scheme.

Compromising: For this attack, the adversary can impersonate valid RFID tags or counterfeit fake RFID tags through a compromising tag. Compromising attack would exist when many tags share a same secret key. However, since the secret key y_i that each tag stores is independent and different from each other in the YL protocol, compromising attack cannot be taken in the proposed scheme.

Denial of Service: If an attacker intercepts the response from the server to the tag, the shared secret of the server and the tag might be out of synchronization, since the server will refresh the secret key while the tag keeps the current key. However, in YL protocol, even if the adversary blocks the messages M_3 and M_4, since the $\mathcal{DB}$ maintains both the old and new keys for each tag, $\mathcal{DB}$ can resynchronize with the tag in the next authentication.

6.1.3. Security Analysis of the YL Protocol

The wireless communications between RFID readers and tags are subject to a variety of attacks. Before we elaborate on our attacking strategies, we will first state a set of reasonable assumptions about our attack model to address the security flaws of YL. In our attack model, it is assumed that the adversary can eavesdrop, block, modify and inject messages in any communication between a reader and a tag to obtain tag identifier, track tag location, impersonate tag and reader, and trigger denial of service. In this section, we present tracking, tag impersonation and desynchronization attacks on YL protocol. Also, the adversary can initiate communication with the reader and the tags. Besides, as mentioned in the original study the used hash function $h()$ is publicly known, i.e. available to adversary. In addition to this, we suppose that the adversary is able to tamper with only one tag e.g., his own tag. Because low-cost tags are not tamper-resistant, such an assumption could be feasible.

Indeed, using a common key for all the tags would be a security flaw such that an adversary who tampers with one tag, would also be able to track all the other tags in the system. It is also mentioned in the YL study and claimed that compromising attack cannot be taken in the YL scheme, since the secret key y_j that each tag stores is independent and different from each other. Nevertheless, $h(k)$ is the common value and stored at each tag and this is the main security flaw of the protocol: If the attacker has the possibility to tamper with only one RFID tag, then he knows $h(k)$ value. Hence in our attack models, we assume that the adversary has knowledge of $h(k)$ which is unique in YL RFID system.

Tracking Attack: According to the YL protocol, $\mathcal{T}_i$ uses the same x_i in computation of M_1 until a successful authentication is done. Moreover the adversary $\mathcal{A}$ can reveal easily x_i from M_1 as $x_i = M_1 \oplus h(h(k) \oplus r_2)$, because both $h(k)$ and r_2 are available to him. For the privacy experiment of YL, the adversary follows the attack as described below.

In the learning phase of the attack, two tags $\mathcal{T}_0$ and $\mathcal{T}_1$ are selected and the tag $\mathcal{T}_0$ is queried by the adversary $\mathcal{A}$ and he captures x_0 of $\mathcal{T}_0$.

- $\mathcal{A}$ *randomly chooses a pair of distinct tags* $\mathcal{T}_0$ *and* $\mathcal{T}_1$.
- $\mathcal{A}$ *initiates communication with* $\mathcal{T}_0$ *using* TagInit.
- $\mathcal{A}$ *transmits some random nonce* r_1 *to* $\mathcal{T}_0$.
- $\mathcal{A}$ *receives* $\mathcal{T}_0$ *response* $\{r_2, M_1, M_2\}$.
- $\mathcal{A}$ *obtains* $x_0 = M_1 \oplus h(h(k) \oplus r_2)$ *by computing* $h(h(k) \oplus r_2)$.

In the challenge phase, the adversary only the queries the selected tag once and observes the response. Then he checks whether the tag x_* is x_0 or not: If the selected

tag is $\mathcal{T}_0$, then $x_* = x_0$. Otherwise $\mathcal{A}$ guesses the selected tag as $\mathcal{T}_1$.

- *$\mathcal{A}$ takes $\mathcal{T}_0$ and $\mathcal{T}_1$ as its challenge candidates.*
- *$\mathcal{A}$ transmits a random nonce r'_1 to the selected tag $\mathcal{T}^*$.*
- *$\mathcal{A}$ receives $\mathcal{T}^*$ response $\{r'_2, M_1, M_2\}$.*
- *$\mathcal{A}$ computes $h(h(k) \oplus r'_2)$ and obtains $x_* = M_1 \oplus h(h(k) \oplus r'_2)$.*
- *If x_* is x_0, $\mathcal{A}$ decides $\mathcal{T}^* = \mathcal{T}_0$. Otherwise guesses $\mathcal{T}^* = \mathcal{T}_1$.*

Notice that, by using the above attack an adversary $\mathcal{A}$ has non-negligible advantage in guessing the identity of a selected tag. In other words, we show that YL protocol do not achieve the untraceability property.

Tag Impersonation Attack: In this type of attack, an adversary attempts to impersonate a target tag without knowing the tags internal secrets. We show that an adversary can impersonate a tag to a legitimate server in the YL mutual authentication protocol. The attack has two phases as given below:

In Phase 1, the adversary queries a tag $\mathcal{T}_i$ as a reader:

- *$\mathcal{A}$ transmits some random nonce r'_1 to $\mathcal{T}_i$.*
- *$\mathcal{T}_i$ generates another random nonce r'_2 and computes $M'_1 = x_i \oplus h(h(k) \oplus r'_2)$, $M'_2 = h(y_i \oplus r'_1 \oplus r'_2)$.*
- *$\mathcal{T}_i$ sends $\{r'_2, M'_1, M'_2\}$ to $\mathcal{A}$.*
- *$\mathcal{A}$ computes $h(h(k) \oplus r'_2)$ and obtains $x_i = M'_1 \oplus h(h(k) \oplus r'_2)$.*
- *$\mathcal{A}$ keeps $\{x_i, r'_1, r'_2, M'_2\}$.*

In Phase 2, the adversary impersonates the tag $\mathcal{T}_i$ to the valid server:

- $\mathcal{A}$ *initiates a session with* $\mathcal{R}$.
- $\mathcal{R}$ *transmits some random nonce* r_1 *to* $\mathcal{A}$.
- $\mathcal{A}$ *sends response* $\{r_2, M_1, M_2\}$ *to* $\mathcal{R}$, *where* $r_2 = r_1' \oplus r_2' \oplus r_1$, $M_1 = x_i \oplus h(h(k) \oplus r_2)$, $M_2 = M_2'$.
- $\mathcal{R}$ *delivers* $\{r_2, M_1, M_2\}$ *to* $\mathcal{DB}$.
- $\mathcal{DB}$ *computes* $x_i = M_1 \oplus h(h(k) \oplus r_2)$, $y_i = f_k(x_i)$.
- $M_2 = h(y_i \oplus r_2 \oplus r_1)$ *is satisfied and* $\mathcal{DB}$ *authenticates* $\mathcal{A}$ *as* $\mathcal{T}_i$, *since*

$$\begin{aligned} h(y_i \oplus r_2 \oplus r_1) &= h(y_i \oplus r_1' \oplus r_2' \oplus r_1 \oplus r_1) \\ &= h(y_i \oplus r_1' \oplus r_2') \\ &= M_2 = M_2'. \end{aligned}$$

By using this attack, the adversary convinces an RFID reader and the authentication server to believe the fake tags are legitimate.

Desynchronization Attack: In a desynchronization attack, which is a particular form of denial-of-service attacks, an adversary intentionally makes a tag out of synchronization with the server to prevent subsequent authentications of the tag. In this part, it is shown that the YL protocol is vulnerable to a desynchronization attack, in which an adversary can update a tag's secret to a random value. Consequently, the tag is desynchronized with the server and authentication of the tag is prevented. Our attack can be launched to any tag as follows:

In Phase 1, the adversary queries a tag $\mathcal{T}_i$ as a reader:

- $\mathcal{A}$ *transmits some random nonce* r_1' *to* $\mathcal{T}_i$.
- $\mathcal{T}_i$ *generates another random nonce* r_2' *and computes* $M_1' = x_i \oplus h(h(k) \oplus r_2')$, $M_2' = h(y_i \oplus r_1' \oplus r_2')$.

- $\mathcal{T}_i$ *sends* $\{r_2', M_1', M_2'\}$ *to* $\mathcal{A}$.
- $\mathcal{A}$ *obtains* $x_i = M_1' \oplus h(h(k) \oplus r_2')$ *by computing* $h(h(k) \oplus r_2')$.

In Phase 2, the adversary authenticates the tag to the server:

- $\mathcal{A}$ *initiates a session between* $\mathcal{R}$ *and* $\mathcal{T}_i$.
- $\mathcal{R}$ *transmits some random nonce* r_1 *to* $\mathcal{T}_i$.
- $\mathcal{T}_i$ *generates random nonce* r_2 *and computes* $M_1 = x_i \oplus h(h(k) \oplus r_2)$, $M_2 = h(y_i \oplus r_1 \oplus r_2)$.
- $\mathcal{T}_i$ *sends* $\{r_2, M_1, M_2\}$ *to* $\mathcal{R}$ *that delivers them to* $\mathcal{DB}$.
- $\mathcal{DB}$ *uses* $\{r_2, M_1, M_2\}$ *and authenticates* $\mathcal{T}_i$.
- $\mathcal{DB}$ *computes* $x_i^* = h(x_i \oplus y_i \oplus r_1 \oplus r_2)$ *and new key* $y_i^* = f_k(x_i^*)$.
- $\mathcal{DB}$ *computes* M_3, M_4, *transmits them to* $\mathcal{T}_i$ *via* $\mathcal{R}$ *and sets* $y_i^{old} \leftarrow y_i$, $y_i \leftarrow y_i^*$.
- $\mathcal{T}_i$ *calculates* $x_i^* = h(x_i \oplus y_i \oplus r_1 \oplus r_2)$ *and obtains* $y_i^* = M_3 \oplus h(x_i^* \oplus y_i)$. *After check of* M_4, $\mathcal{T}_i$ *authenticates the server and sets:* $x_i \leftarrow x_i^*$, $y_i \leftarrow y_i^*$.

In Phase 3, the adversary impersonates the tag to the server and desynchronizes them:

- $\mathcal{A}$ *initiates a session with* $\mathcal{R}$.
- $\mathcal{R}$ *transmits some random nonce* $\overline{r}_1$ *to* $\mathcal{A}$.
- $\mathcal{A}$ *sends response* $\{\overline{r}_2, \overline{M}_1, \overline{M}_2\}$ *to* $\mathcal{R}$, *where* $\overline{r}_2 = \overline{r}_1 \oplus r_2' \oplus r_1'$, $\overline{M}_1 = x_i \oplus h(h(k) \oplus \overline{r}_2)$, $\overline{M}_2 = M_2'$ *(Note that* $\{x_i, r_2', r_1', M_2'\}$ *are available to* $\mathcal{A}$ *from Phase 1).*
- $\mathcal{R}$ *delivers* $\{\overline{r}_2, \overline{M}_1, \overline{M}_2\}$ *to* $\mathcal{DB}$
- $\mathcal{DB}$ *computes* $x_i = \overline{M}_1 \oplus h(h(k) \oplus \overline{r}_2)$, $y_i = f_k(x_i)$. $\mathcal{DB}$ *recognizes that* $y_i = y_i^{old}$. *Then checks whether* $\overline{M}_2 = h(y_i^{old} \oplus \overline{r}_2 \oplus \overline{r}_1)$. *Notice that* $h(y_i^{old} \oplus \overline{r}_2 \oplus \overline{r}_1) = h(y_i^{old} \oplus r_1' \oplus r_2' \oplus \overline{r}_1 \oplus \overline{r}_1) = h(y_i^{old} \oplus r_1' \oplus r_2') = \overline{M}_2$. *Thus,* $\overline{M}_2 = h(y_i^{old} \oplus \overline{r}_2 \oplus \overline{r}_1)$ *is satisfied and* $\mathcal{DB}$ *authenticates* $\mathcal{A}$ *as* $\mathcal{T}_i$.
- $\mathcal{DB}$ *computes* $x_i^{**} = h(x_i \oplus y_i \oplus \overline{r}_1 \oplus \overline{r}_2)$ *and new key* $y_i^{**} = f_k(x_i^{**})$.
- $\mathcal{DB}$ *computes* $\overline{M}_3$, $\overline{M}_4$, *transmits them to* $\mathcal{A}$ *through* $\mathcal{R}$ *and* $y_i^{old} \leftarrow y_i$, $y_i \leftarrow y_i^{**}$.

- $\mathcal{A}$ *ends the protocol.*

At the end, $\mathcal{DB}$ stores y_i^{old}, y_i^{**} as the key pairs for $\mathcal{T}_i$. On the other hand, $\mathcal{T}_i$ keeps y_i^* as its secret key. Hence, if we prove that $y_i^* \neq y_i^{old}$ and $y_i^* \neq y_i^{**}$, it is obviously said that in the future authentication sessions $\mathcal{DB}$ will not able to authenticate $\mathcal{T}_i$. We have the following information:

$$\begin{aligned} x_i^{old} &= x_i, \\ x_i^* &= h(x_i \oplus y_i \oplus r_1 \oplus r_2), \\ x_i^{**} &= h(x_i \oplus y_i \oplus \overline{r}_1 \oplus \overline{r}_2) \\ &= h(x_i \oplus y_i \oplus \overline{r}_1 \oplus \overline{r}_1 \oplus r_1' \oplus r_2') \\ &= h(x_i \oplus y_i \oplus r_1' \oplus r_2'). \end{aligned}$$

One can see that $r_1 \oplus r_2 \neq r_1' \oplus r_2'$, since r_1, r_2, r_2' are randomly selected (in fact r_1' should be also selected randomly depending on the adversary choice). Thus, $x_i^{old} \neq x_i^* \neq x_i^{**}$. Because $y_n = f_k(x_n)$ for $n \leq N$, this leads to $y_i^* \neq y_i^{old}$ and $y_i^* \neq y_i^{**}$.

6.2. Improving the YL Protocol

In this section, in the light of described weaknesses of YL protocol, we are motivated to propose some countermeasures to eliminate these vulnerabilities. So, instead of proposing a new RFID authentication protocol, we only revise the YL protocol and present security enhancements by making some minor changes.

6.2.1. A Revised Protocol to Improve Security of the YL

In this part, we revise YL with minor changes to make the resulting protocol immune against the security risks described previously. Let K be a secret key shared only by $\mathcal{DB}$ and the registered tags. Then the revised protocol is described as follows:

Initially, for each RFID tag $\mathcal{T}_i$ a random number x_i a secret key y_i are generated by $\mathcal{DB}$. Then, x_i, y_i pair is assigned to $\mathcal{T}_i$ which stores them with K. On the other hand, $\mathcal{DB}$ stores K and $\{x_i, y_i, x_i^{old}, y_i^{old}\}$ tuple with related information for each tag $\mathcal{T}_i$. Initially, $x_i^{old} = x_i$ and $y_i^{old} = y_i$.

A step by step description of the revised model is given below, also a summary of the revised protocol is seen in Figure 6.2:

i) $\mathcal{R}$ generates a random nonce r_1 and sends it to $\mathcal{T}_i$ as a query.

ii) $\mathcal{T}_i$ produces another random nonce r_2 and computes $M_1 = x_i \oplus h(K \oplus r_2)$, $M_2 = h(y_i||r_1||r_2)$.

iii) $\mathcal{T}_i$ sends $\{r_2, M_1, M_2\}$ to $\mathcal{R}$.

iv) $\mathcal{R}$ transmits $\{r_1, r_2, M_1, M_2\}$ to $\mathcal{DB}$.

v) $\mathcal{DB}$ computes $x_i = M_1 \oplus h(K \oplus r_2)$ and searches x_i with the x and x^{old} values stored in the database. If the values exists in the database, then $\mathcal{DB}$ uses corresponding y_i in the table and checks whether $M_2 = h(y_i||r_1||r_2)$. If it holds, $\mathcal{DB}$ authenticates $\mathcal{T}_i$. Otherwise it sends an error to $\mathcal{R}$ and terminates the session.

vi) $\mathcal{DB}$ computes new key $y_i^* = h(1||x_i \oplus y_i||r_1 \oplus r_2)$.

vii) $\mathcal{DB}$ calculates $M_3 = x_i^* = h(x_i \oplus y_i||r_1||r_2)$, sends M_3 with related data of $\mathcal{T}_i$ to $\mathcal{R}$ and sets $x_i^{old} \leftarrow x_i$, $x_i \leftarrow x_i^*$, $y_i^{old} \leftarrow y_i$, $y_i \leftarrow y_i^*$.

viii) $\mathcal{R}$ forwards M_3 to $\mathcal{T}_i$.

ix) $\mathcal{T}_i$ checks $M_3 = h(x_i \oplus y_i||r_1||r_2)$. If it holds, $\mathcal{T}_i$ authenticates the server and sets $x_i^* = M_3$ and new key $y_i^* = h(1||x_i \oplus y_i||r_1 \oplus r_2)$. Next, it sets: $x_i \leftarrow x_i^*$, $y_i \leftarrow y_i^*$. Otherwise keeps x_i, y_i unchanged.

6.2.2. Security Analysis of the Revised Model

In this section, we analyze the security of the revised protocol with respect to the previously described attacks under the same assumptions.

Resistance to Tag Impersonation Attack: If one analyzes the revised scheme, it can be seen that an adversary cannot impersonate a tag $\mathcal{T}$ to reader $\mathcal{R}$ by using the previ-

Data Base / Reader $[K, x_i, y_i, x_i^{old}, y_i^{old}]$		**Tag** $[K, x_i, y_i]$
Generates r_1	r_1 ⟶	Generates r_2 and computes: $M_1 = x_i \oplus h(K \oplus r_2)$, $M_2 = h(y_i \| \| r_1 \| \| r_2)$
	$\{r_2, M_1, M_2\}$ ⟵	
Computes: $x_i = M_1 \oplus h(K \oplus r_2)$, find corresponding y_i **If** $M_2 = h(y_i \| \| r_1 \| \| r_2)$, computes: $y_i^* = h(1 \| \| x_i \oplus y_i \| \| r_1 \oplus r_2)$ $M_3 = x_i^* = h(x_i \oplus y_i \| \| r_1 \| \| r_2)$ $x_i^{old} \leftarrow x_i$, $x_i \leftarrow x_i^*$ $y_i^{old} \leftarrow y_i$, $y_i \leftarrow y_i^*$	M_3 ⟶	**If** $M_3 = h(x_i \oplus y_i \| \| r_1 \| \| r_2)$, Computes: $x_i^* = M_3$ $y_i^* = h(1 \| \| x_i \oplus y_i \| \| r_1 \oplus r_2)$ $x_i \leftarrow x_i^*$, $y_i \leftarrow y_i^*$

Figure 6.2. The revised YL protocol.

ously described tag impersonation attack. The reason is that only the registered tag can compute $M_2 = h(y_i||r_1||r_2)$ which directly depends on freshly generated r_1 and r_2 values, not their XOR summation. The YL protocol is vulnerable to the described tag impersonation attack due to the use of inexpensive security operation $\oplus$ inappropriately, while we prefer to use $||$ operator instead to prevent $\oplus$ message modification tricks. Hence, an adversary cannot convince the reader to believe the fake tags are legitimate, even if he knows K.

Resistance to Desynchronization Attack: The proposed model prevents an adversary to realize the last phase of the described desynchronization attack, because only the legitimate tag which has y_i can convince the back-end server to initiate a secret update process.

Resistance to Tracking Attack: In this model, a common secret as K is still used and stored by all tags. Thus, an adversary can easily realize the tracking attack described in Section 6.1.3. Obviously, to provide resistance against the tracking attack, K should be assigned such that it is different for each tag. Nevertheless in this case $\mathcal{DB}$ needs to perform a linear search in tag identification and this conflicts with the reducing the computational complexity at the server side as the main motivation of YL.

6.2.3. Performance Comparison of the YL and the Revised Protocol

Our revisions on the original protocol reduces the computational complexity at both tag and server side. To compare the computational complexities of the revised and YL protocols, we take into account computationally complex functions as the hash function $h()$ and the keyed-hash function $f()$ under the assumption that they require same computational effort. A tag in our revised protocol only needs to compute the hash function $h()$ four times (two times in producing M_1, M_2, one time in authenticating the server and one time in the secret update process). Also the server requires to compute four hash functions for our revised model (in authentication of a tag $h()$ is used two times and in the secret update process $h()$ is used two times). On the other hand, for YL protocol the tag and the server uses the hash functions five times and

Table 6.1. Comparisons of YL and the revised protocol.

	Tag Computation	Server Computation
YL	5H	7H
Revised	4H	4H

H: A one-way hash function

seven times respectively (at the tag side $h()$ is used five times while at the server $f()$ is used two times and $h()$ is used five times). Hence, the revised model requires fewer complex function invocations at the tag and server side than YL needs. Furthermore, our revision on the YL protocol does not change the storage requirement at the tag side. Table 6.1 gives a summary of comparison between the revised protocol and YL.

6.3. Security Flaws in the SLMAP* Protocol

Recently, Li, Deng and Wang improved SLMAP, described in Section 4.4.3, by re-designing it [55]. We denote this improved protocol as SLMAP* and describe its details and security claims in the following paragraphs. Also, we exploit an unnoticed flaw in its design rationale and show that some of the freshly produced variables can be assigned to different values for the reader and the tag that presumably causes a desynchronization vulnerability.

In order to simplify the presentation, we use following notation throughout the text.

IDS	Index pseudonym
$SecretKeys$	The secret keys $K1, K2, K3$
X_{LSB}	Least significant bit of X
$+$	Addition mod 2^m
$\oplus$	Bitwise XOR operation
X_{MSB}	Most significant bit of X

6.3.1. Description of the SLMAP*

Initially, each RFID tag is assigned with two identifiers: one is an index pseudonym as IDS which is updated at the end of a successful protocol session; the other is a real identifier as ID which is a permanent identifier of the tag. Moreover, each tag is associated with three secret keys as $K1$, $K2$ and $K3$ and they are also updated for every successful protocol run. The $K1$, $K2$, $K3$ and ID are kept secret and shared only by the back-end server and the corresponding tag. For each tag, the back-end database stores the tuple $[IDS, ID, K1, K2, K3]$.

A step by step description of the SLMAP* authentication process is given below. Also a short summary of the protocol is depicted in Table 6.2.

- $\mathcal{R} \rightarrow \mathcal{T}$: *hello*
- $\mathcal{T} \rightarrow \mathcal{R}$: IDS
- $\mathcal{R}$ checks whether IDS exists or not at the back-end server. If not; the session is terminated, otherwise it continues with the next steps.
- $\mathcal{R}$ generates a random r, then by using it the messages A and B are computed where

$$\begin{aligned} A &= IDS \oplus K1 + r, \\ B &= (IDS + K2) \oplus r. \end{aligned}$$

- $\mathcal{R} \rightarrow \mathcal{T}$: $A||B$

- $\mathcal{T}$ computes $r1$ from A and $r2$ from B by using $SecretKeys$ $K1$, $K2$ and IDS as follows.

$$\begin{aligned} r1 &= A - (IDS^n \oplus K1^n), \\ r2 &= B \oplus (IDS^n + K2^n). \end{aligned}$$

- If $r1 = r2$, $\mathcal{T}$ obtains r correctly and sets the protocol status bit to 1. Then it prepares the answer message C, where

$$\begin{aligned} C &= (IDS + ID \oplus r) \oplus (K1 + r) \\ &\quad \oplus (K2 + r) \oplus (K3 + r). \end{aligned}$$

- If $r1 \neq r2$, $\mathcal{T}$ generates a random message C' and sends it to the $\mathcal{R}$.
- $\mathcal{T} \rightarrow \mathcal{R}$: C (or C')
- $\mathcal{R}$ verifies message C with its local computation. If they are equal, then $\mathcal{R}$ computes message D, where

$$\begin{aligned} D &= IDS_{next} \\ &= (IDS + K1 + K2) \oplus r + (ID + K3) \oplus r. \end{aligned}$$

- $\mathcal{R}$ updates IDS, $K1$, $K2$ and $K3$.
- $\mathcal{R} \rightarrow \mathcal{T}$: D
- $\mathcal{T}$ locally computes the message D and compares it with the received one. If they are equal, $\mathcal{T}$ updates IDS, and $SecretKeys$ $K1$,$K2$, $K3$ and sets the protocol status bit to 0 for the purpose of indicating the completion of a successful protocol execution.

As mentioned above an additional status bit is used: it is set to 0 if the protocol is completed and set to 1 otherwise. As long as status bit is 1, the tag expects a completion message D from the reader. However, after some delay or a number of trials, e.g. c, the tag resets its state and returns its status to the beginning of current protocol run without changing any secrets.

Table 6.2. The nth protocol run of SLMAP*.

$\mathcal{R} \rightarrow \mathcal{T}:$	$hello$
$\mathcal{T} \rightarrow \mathcal{R}:$	IDS^n
$\mathcal{R} \rightarrow \mathcal{T}:$	$A \| \| B$
$\mathcal{T}:$	If $r1 = r2$ computes C, else produces C'
$\mathcal{T} \rightarrow \mathcal{R}:$	C or C'
$\mathcal{R} \rightarrow \mathcal{T}:$	D

where:

$$A = IDS^n \oplus K1^n + r$$
$$B = (IDS^n + K2^n) \oplus r$$
$$r1 = A - (IDS^n \oplus K1^n)$$
$$r2 = B \oplus (IDS^n + K2^n)$$
$$C = (IDS^n + ID \oplus r) \bigoplus_{i=1}^{3} (Ki^n + r)$$
$$D = IDS^{n+1} = (IDS^n + K1^n + K2^n) \oplus r + (ID + K3^n) \oplus r$$

6.3.2. Security Claims of the SLMAP*

We summarize the security claims of SLMAP* by using the active attack scenario given at the original paper [55] in which an adversary can eavesdrop, block, modify and inject messages in any communication between a reader and a tag.

i) The adversary $\mathcal{A}$ intercepts $A||B$ and sends a faked message $\hat{A}||\hat{B}$ to the tag $\mathcal{T}$. The tag obtains $r1$ and $r2$ from $\hat{A}$ and $\hat{B}$ respectively. $\mathcal{A}$ then expects a reply from $\mathcal{T}$ that tells whether the modification is a success or not. According to SLMAP* design, $\mathcal{T}$ will always send back a reply C or C' depending on whether $r1 = r2$ is verified. Hence, $\mathcal{A}$ has no way to distinguish these two replies unless secrets stored in the tag are available to $\mathcal{A}$. In addition, if no valid message $A||B$ pair is received, $\mathcal{T}$ terminates the protocol after transmitting C'. Therefore to continue the protocol, $\mathcal{A}$ has no other way but to send the original pair $A||B$.
ii) $\mathcal{A}$ modifies C received from $\mathcal{T}$ as $\hat{C}$ and sends it $\mathcal{R}$. Then, $\mathcal{A}$ analyzes the response of $\mathcal{R}$. In fact, $\mathcal{R}$ responds with D, only if C is verified. However, it is assumed that $\mathcal{R}$ is equipped with enough resources to detect or recover malicious attacks, e.g., the multiple trials from $\mathcal{A}$. Thus, $\mathcal{R}$ only accepts the message C based on the secret values $\{K1, K2, K3, r, ID\}$ and it prevents launching of trials by $\mathcal{A}$.
iii) $\mathcal{A}$ intercepts D and modifies it as $\hat{D}$. Next, $\mathcal{A}$ sends it to $\mathcal{T}$. Since D is also generated by a series of current secret values, it is computationally impractical to mislead $\mathcal{T}$ in accepting $\hat{D}$.

According to the described scenario, SLMAP* provides resistance against active attacks. However, in the next section, we show that the first item misleads the SLMAP* security analysis and it makes the scheme vulnerable to a desynchronization attack by manipulating bits of the transmitted messages A and B.

6.3.3. A Practical Desynchronization Attack on SLMAP*

Most RFID authentication protocols include an update operation for the tag secrets and its ID at the end of a successful protocol run. Since this operation is

performed at the back-end server as well as at the tag, synchronization of secret information between the database and the tag is crucial for future authentications. In a desynchronization attack which is one kind of DoS attacks, an adversary intentionally makes a tag out of synchronization with the server to prevent subsequent authentications of the tag.

In this part, we firstly describe a bit manipulation attack against SLMAP* so that the reader and the tag compute different values for r. We will, however, show that both of the reader and the tag will not be aware of this flaw and complete the protocol with a success. Eventually, having different r values at the tag and the reader side presumably causes a desynchronization vulnerability. Since the secret update procedure of SLMAP* is not explicitly given[2] in the original study, speculating whether this flaw gives rise to a desynchronization attack is not straightforward. On the other hand, as most of the similar protocols like in [16–18, 20] use freshly produced values in their secrets update stages. We believe, it is reasonable to consider the desynchronization risk for SLMAP* since r corresponds to such a value for SLMAP*. For example, the SLMAP protocol uses r in the secret update phase as follows [19]:

$$\begin{aligned} K_1^{m+1} &= K_1^m \oplus r + (IDS^{m+1} + K_2^m + ID), \\ K_2^{m+1} &= K_2^m \oplus r + (IDS^{m+1} + K_1^m + ID). \end{aligned}$$

In our attack model, it is assumed that the adversary can initiate communication with the reader and the tags. Moreover, we suppose that the attacker can eavesdrop, block, modify and inject messages in any communication between a reader and a tag. In this respect, our attack boils down to create different r values at the tag and the reader side, and expect to have different secrets after the update procedure that causes a desynchronization. We show that if an adversary flips $[r]_{MSB}$, then he/she puts the

[2]The authors of SLMAP* do not explicitly describe the update procedure for the secrets, since they mention update of the secret values securely at both sides with ultra-lightweight-only settings is a challenging problem. Therefore they have left it as a future work.

tag and the reader out of synchronization. Our attack can be launched as follows:

In Phase 1, the adversary authenticates the tag to the reader:
In this phase, the adversary $\mathcal{A}$ initiates a session between the reader and the tag. Then, $\mathcal{A}$ relays all the messages except message D, from reader to tag. Reader updates $SecretKeys$ and IDS after sending D while tag does not, since D is blocked.

- $\mathcal{A}$ initiates a session between $\mathcal{R}$ and $\mathcal{T}$: $\mathcal{A}$ sends *hello* to $\mathcal{T}$.
- $\mathcal{T}$ transmits IDS to $\mathcal{A}$ who delivers it to $\mathcal{R}$.
- $\mathcal{R}$ checks existence of IDS. Then it generates a random r and computes the values A and B.
- $\mathcal{R}$ replies with the message $A||B$ to the adversary and he delivers it to $\mathcal{T}$.
- $\mathcal{T}$ obtains $r1$ and $r2$ from A and B respectively.
- $\mathcal{T}$ computes C and sends it to $\mathcal{A}$ who relays it to $\mathcal{R}$.
- $\mathcal{R}$ verifies C and computes D.
- $\mathcal{R}$ updates $IDS, K1, K2, K3$ and sends D to $\mathcal{A}$.
- $\mathcal{A}$ stores the tuple $\{A, B, D\}$ and ends the protocol.

In Phase 2, the tag and the reader computes different r values:
From the previous phase of the attack $\mathcal{A}$ does not know the value r, but he has information of $\{A, B, D\}$. Let I be the bit sequence as $I = [100 \cdots 000]$ and $\hat{r}$ denote the modified r as $\hat{r} = r \oplus I = r \oplus [100 \cdots 000]$. After a while or multiple tries (e.g. c times), the tag resets the status bit to 0 without changing any $SecretKeys$ or IDS [55]. Afterwards, the adversary executes the second phase of the attack as follows:

- To result in $r1 = r2 = \hat{r}$ such that $[r1]_{MSB} = [r2]_{MSB} = [r]_{MSB} \oplus 1$ at the tag side, $\mathcal{A}$ modifies A and B messages as $\hat{A}$ and $\hat{B}$ respectively, where $\hat{A} = A + I$ and $\hat{B} = B \oplus I$.
- $\mathcal{A}$ initiates a session with $\mathcal{T}$: $\mathcal{A}$ sends *hello* to $\mathcal{T}$.
- $\mathcal{T}$ transmits IDS to $\mathcal{A}$.
- $\mathcal{A}$ sends $\hat{A}||\hat{B}$ to $\mathcal{T}$.

- After receiving $\hat{A}||\hat{B}$, the $\mathcal{T}$ computes:

$$\begin{aligned} r1 &= \hat{A} - (IDS \oplus K1), \\ &= A + I - (IDS \oplus K1), \\ &= r + I + (IDS \oplus K1) - (IDS \oplus K1), \\ &= r + I. \end{aligned}$$

- $\mathcal{T}$ obtains $r1 = r + I$. In a similar way the tag evaluates:

$$\begin{aligned} r2 &= \hat{B} \oplus (IDS + K2), \\ &= r \oplus I \oplus (IDS + K2) \oplus (IDS + K2), \\ &= r \oplus I. \end{aligned}$$

- $\mathcal{T}$ obtains $r2 = r \oplus I$ and checks whether $r1 = r2$. Notice that only $[I]_{MSB} = 1$ and other bits of I are 0, so $r \oplus I = r + I$ and the equation is satisfied. It is equivalent to say that $r1 = r2 = \hat{r}$.
- $\mathcal{T}$ computes C and transmits it to $\mathcal{A}$.
- $\mathcal{A}$ ignores C message received from $\mathcal{T}$ and he sends D to $\mathcal{T}$.
- $\mathcal{T}$ calculates $\hat{D} = (IDS + K1 + K2) \oplus \hat{r} + (ID + K3) \oplus \hat{r}$.
- Because $D = (IDS + K1 + K2) \oplus r + (ID + K3) \oplus r$, $\hat{D} = D$ is hold and $\mathcal{T}$ updates its secrets.

Theorem 6.1. *By considering the proposed attack if the equality $\hat{r} = r1 = r2 = r \oplus I$ is satisfied at the tag side, then for any value of $K1$, $K2$, $K3$, ID and IDS, the tag verifies $\hat{D} = D$, i.e. the described desynchronization attack is succeeded.*

Proof. Let X and Y stand for $(IDS + K1 + K2)$ and $(ID + K3)$ respectively. Then we can express

$$D = (X \oplus r) + (Y \oplus r),$$

$$\hat{D} = (X \oplus \hat{r}) + (Y \oplus \hat{r}).$$

Note that $\hat{D}$ and D can differ only in their most significant bits, because the modified bit of r is the most significant bit as $\hat{r} = r \oplus I$. Thus, if we show that in all cases $[\hat{D}]_{MSB} = [D]_{MSB}$, we also confirm $\hat{D} = D$ for all values of $K1$, $K2$, $K3$, ID and IDS. Therefore we use the truth table given below in order to prove the statement.

Table 6.3. Truth table of the most significant bit values of X, Y, r, $\hat{r}$, D and $\hat{D}$.

X_{MSB}	Y_{MSB}	$r_{MSB}/\hat{r}_{MSB}$	$D_{MSB}/\hat{D}_{MSB}$
0	0	0/1	0/0
0	0	1/0	0/0
0	1	0/1	1/1
0	1	1/0	1/1
1	0	0/1	1/1
1	0	1/0	1/1
1	1	0/1	0/0
1	1	1/0	0/0

As it can be inferred from Table 6.3, $D = \hat{D}$ for all values of the respective variables. Hence, at the end of execution of the proposed attack, the tag verifies received D with the computed one and updates $SecretKeys$ and IDS. □

Example 6.2. *To illustrate the attack, we will give an example with reduced bits. Let's assume* $K1 = 11110011$, $K2 = 00111011$, $K3 = 11100100$, $r = 10011011$, $ID = 11000011$, $IDS = 01111100$. *From these values one can compute* $A = 00101010$, $B = 00101100$ *and* $D = 01101101$. *The modified results* $\hat{A} = 10101010$ *and* $\hat{B} = 10101100$ *are generated by an adversary by using* A, B *and he sends* $\hat{A}||\hat{B}$ *to the tag. Upon receiving the messages* $\hat{A}$ *and* $\hat{B}$ *the tag calculates* $\hat{r1}, \hat{r2}$ *as* $\hat{r1} = 00011011$ $\hat{r2} = 00011011$. *Note that* $\hat{r1} = \hat{r2}$ *is verified, but* $\hat{r1} = \hat{r2} = r \oplus I$ *is obtained instead of* r. *Next, the tag calculates* C *and transmits it to the* $\mathcal{A}$. *The adversary discards*

the response of tag C, because it has no importance in remaining part of the attack. $\mathcal{A}$ sends D to the tag which checks it with computing $\hat{D} = 01101101$. Note that the equality $D = \hat{D}$ is hold and the tag updates its $SecretKeys$ and IDS. As it can be seen that the tag does not recognize a false r value, and such a flaw violates the security claim of SLMAP.*

6.4. Security Flaws in the SBAP Protocol

In this section, we analyze a recently proposed ultra light weight security method named as the Spacing Based Authentication Protocol (SBAP). We manage to attain a total breakdown by compromising the secret key information. Other weaknesses, we report for SBAP includes traceability, replay and desynchronization flaws.

6.4.1. Description of the SBAP

SBAP is a ultra lightweight protocol that uses an XOR operation and a spacing algorithm to generate a new secret ID for each session [63]. SBAP proposes two authentication methods for the same protocol where the second method is an enhanced version of the first one that reduces server time complexity.

Assume that each tag keeps its own secret S and a tag ID, and the reader stores a list of secret IDs. For tag authentication, SBAP uses a partial ID which is denoted by P and generated as follows:

$$P = P_{odd} \oplus P_{even}$$

where $P_{odd} := f(S, u, odd)$ and $P_{even} := f(S, u, even)$ for an ultra-light weight extracting function $f(S, u, b)$ having three inputs: bit string S, random spacing factor u and a Boolean variable b.

Generation of the f function is quite simple: let L be the length of the bit stream S, u be a positive integer dividing L. Thus, we may write $S = s_0 s_1 \dots s_{L-1}$, and

partition S to get smaller bit streams $q_i = (s_{iu}s_{iu+1}\ldots s_{iu+u-1})$ for $i = 0, 1, \ldots, L/u-1$. Once S is partitioned into q_i values, $P_{odd} := f(S, u, 1)$ and $P_{even} := f(S, u, 0)$ are simply calculated by concatenating the odd and even indexed q_i digits respectively.

For instance, in Figure 6.3, we illustrate how the spacing process works for a secret S having $L = 16$ bits long, and the spacing factor $u = 2$.

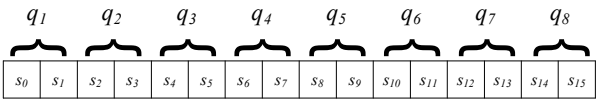

Figure 6.3. An example for the spacing algorithm.

Note that after the partitioning, one computes P_{odd} and P_{even} values as follows:

$$P_{odd} = q_1 \parallel q_3 \parallel q_5 \parallel q_7,$$
$$P_{even} = q_0 \parallel q_2 \parallel q_4 \parallel q_6$$

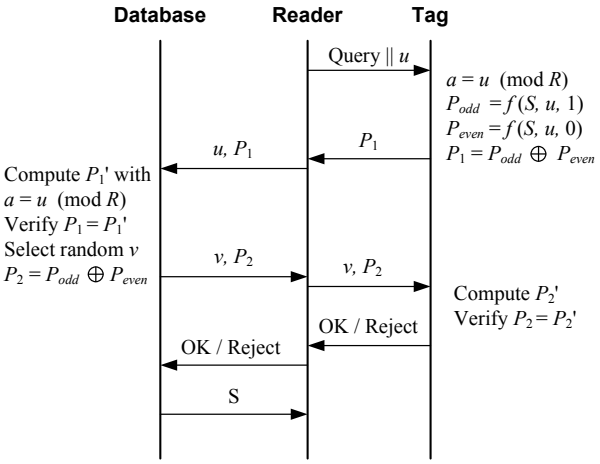

Figure 6.4. Protocol flow for method 1 in SBAP.

Our next step is to describe SBAP with its regular and enhanced methods. The enhanced SBAP is proposed to reduce the searching/authentication time in the server if there is a fair amount of tags in the system. The main difference between two protocols is that in the enhanced method, both tag and server generate and save a P_n value for the next section's use. In other words, the tag always responses with P_n and the server

initially performs a quick search for P_n in its saved P_n database. If a match exist, the server authenticates the tag otherwise it goes through the regular process. Such an approach surely reduces the server load as enhanced SBAP performs less spacing operations.

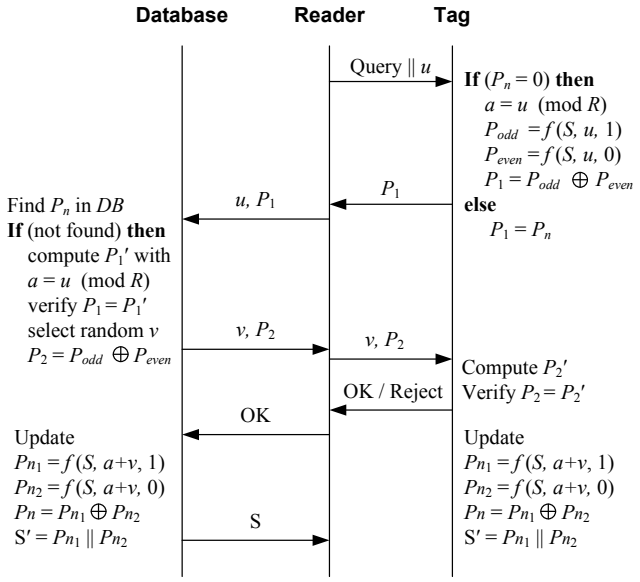

Figure 6.5. Protocol flow for enhanced method in SBAP.

With these remarks in mind, the details of both regular and enhanced SBAP are described in the following paragraphs and further illustrated in Figure 6.4 and Figure 6.5 respectively.

i) Partial ID Generation:

- The reader generates a random nonce, and sends a request along with this nonce to the tag.
- The first method reduces the range of this random nonce and generates a new partial ID using the calculations above, and responses with it. But the second method only computes a new partial ID if it does not have any P_n computed in the previous successful session. After a successful session, the second method always responses with the last computed P_n value.

ii) Authentication:

- Upon receiving partial ID P_1, the reader forwards it to the database.
- The server looks for a match for P_1 in the P_n database, if no such P_1 exists, it calculates P_1 with the random nonce u and secret key S
- After verification database generates a random nonce v, and calculates a message P_2 with this nonce using the same spacing algorithm. The reader then forwards (P_2, v) pair to the tag.

iii) Verification & Key Update:

- Upon receiving verification message P_2, the tag verifies its correctness.
- After checking P_2 value, in both methods, the tag simply sends "OK/Reject" response to the reader, and the reader forwards it to the database.
- However, the enhanced SBAP performs an additional key update session. After verification, it updates the secret key by using the following two values:

$$P_{n_1} = f(S, a + v, odd),$$
$$P_{n_2} = f(S, a + v, even)\ .$$

  The P_n value for the coming session is calculated as $P_n = P_{n_1} \oplus P_{n_2}$ and the secret key S is updated as $S = P_{n_1} \parallel P_{n_2}$.
- On the other hand, once the database receives the last "OK" message it performs the same update to calculate the fresh P_n and S values.

6.4.2. Description of the Attacks on the SBAP Protocol

Although SBAP claims to provide both security and privacy in its design objectives, in this section we outline very efficient attacks that SBAP fails to fulfill its claims. In fact, we manage to attain a total breakdown by compromising the secret key information. Other weaknesses, we report for SBAP includes traceability, replay and desynchronization flaws.

Key Secrecy: We claim that SBAP does not hold the key secrecy. In order to prove this this claim, we exhibit an attack for the enhanced method that captures the secret key of the tag without an exhaustive search.

Assume that P_n is not assigned (means it is equivalent to 0). In this phase any request to the tag would be replied back with a fresh P_1. Notice that the adversary could get the length information of the secret key S by simply sending the nonce $N_r = 1$ to the tag since the tag would response such a message with P_1 having a length equals to the half of the secret key length.

Remark 6.3. *The authors in [63] did not explicitly discuss the relation of the secret length L with the spacing factor u. However, according to our analysis the protocol would be extremely weak if*

- *spacing factor u does not divide L*
- *u divides L but L/u is odd*

Notice that, in both cases the secret S needs a padding in order to generate P_1 value. Since XOR of the padding and the secret key bits are open the adversary would extract the secret key bits from P_1.

Next proposition shows that even if the assumptions in Remark 6.3 are satisfied, the key space for S can be shrunk.

Proposition 6.4. *Let S be a secret having a length L; the spacing factor $u = 2$ divide L and $L/2$ be even, then the bit search space for S can be shrunk to $L/4$.*

Proof. Since L/u is even for $u = 1$ and $u = 2$, the adversary may send the nonces 1 and 2 without completing a session successfully. If P_1 and P_1' are the respective responses of the tag and $S = s_0 s_1 \dots s_{L-1}$ represents the secret bit stream, P_1 and P_1' can be

given as follows:

$$P_1 = (s_0 \oplus s_1)(s_2 \oplus s_3)\dots(s_{L-2} \oplus s_{L-1}) \ ,$$
$$P_1' = (s_0 \oplus s_2)(s_1 \oplus s_2)\dots(s_{L-3} \oplus s_{L-1}) \ .$$

From these values, the following linear equations could be written for $i = 0, 1, \dots, L/4-1$:

$$P_1[2i] = P_{odd}[2i] \oplus P_{even}[2i] = s_{4i} \oplus s_{4i+1} \ ,$$
$$P_1'[2i] = P_{odd}'[2i] \oplus P_{even}'[2i] = s_{4i} \oplus s_{4i+2} \ ,$$
$$P_1[2i+1] = P_{odd}[2i+1] \oplus P_{even}[2i+1] = s_{4i+1} \oplus s_{4i+3} \ ,$$

but these leads to

$$s_{4i+1} = P_1[2i] \oplus s_{4i} \ ,$$
$$s_{4i+2} = P_1'[2i] \oplus s_{4i} \ ,$$
$$s_{4i+3} = P_1[2i+1] \oplus s_{4i+1} = P_1[2i+1] \oplus P_1[2i] \oplus s_{4i} \ ,$$

which means that the bit search space for S can be shrunk to $L/4$ since s_{4i+1}, s_{4i+2} and s_{4i+3} can be written as a sum of s_{4i} for $i = 0, 1, \dots, L/4-1$. □

In fact the choice of L shrinks the bit search space of S even further, in particular, if L is a power of 2, S can be compromised with a single bit search. We give this result as a corollary for the following theorem.

Theorem 6.5. *Let $L = k2^m$ for some positive integer m and odd k, then the bit search space for S can be shrunk to $L/2^m$.*

Proof. Since L/u is even for 2^i for $i = 0, 1, \dots, m-1$, assume that the adversary sends the nonces $1, 2^1, 2^2, \dots, 2^{m-1}$ without completing a session successfully, and gets the

following respective responses from the tag:

$$\begin{aligned}
P_1^0 &= (s_0 \oplus s_1)(s_2 \oplus s_3)\dots(s_{L-2} \oplus s_{L-1}) \ , \\
P_1^1 &= (s_0 \oplus s_2)(s_1 \oplus s_3)\dots(s_{L-3} \oplus s_{L-1}) \ , \\
P_1^2 &= (s_0 \oplus s_4)(s_1 \oplus s_5)\dots(s_{L-5} \oplus s_{L-1}) \ , \\
&\vdots \qquad\qquad\qquad\qquad \vdots \\
P_1^m &= (s_0 \oplus s_{2^m})(s_2 \oplus s_{2^m+1})\dots(s_{L-2^{m-1}-1} \oplus s_{L-1}) \ .
\end{aligned}$$

Notice that for $i = 0, 1, 2, \dots, 2^m - 1$, any s_i could be written as a combination of the bits of $P_1^0, P_1^1, \dots, P_1^m$ and s_0. Since $L = k2^m$, a similar analysis could be done for each adjacent disjoint subset of S having 2^m bits. In other words, any s_i in S could be written as a combination of the bits of $P_1^0, P_1^1, \dots, P_1^m$ and $s_0, s_{2^m}, s_{2\cdot 2^m}, \dots, s_{(k-1)2^m}$. Since there are k such base elements and P_1^i for $i = 0, 1, 2, \dots, m$ are known, it suffices to search the bit space $k = L/2^m$ instead of searching L. □

Corollary 6.6. *Let $L = 2^m$ for some positive integers m then the bit search space for S in SBAP shrinks to a single bit search.*

Proof. The length of S is $L = 2^m$ implies $k = 1$, hence, the bit search space is single bit search by Theorem 1. In other words, S can be written as a linear combination of s_0. □

We give the following toy example in order to present the power of the described attack.

Example 6.7. *Assume that secret ID S is a 8 bit key where $S = s_0 s_1 \dots s_7$. Let the adversary send three nonces 1,2 and 4 without completing a session successfully, then the tag responses with P_1^0, P_1^1 and P_1^2 messages calculated as follows:*

$$\begin{aligned}
P_1^0 &= (s_0 \oplus s_1)(s_2 \oplus s_3)(s_4 \oplus s_5)(s_6 \oplus s_7) \\
P_1^1 &= (s_0 \oplus s_2)(s_1 \oplus s_3)(s_4 \oplus s_6)(s_5 \oplus s_7) \\
P_1^2 &= (s_0 \oplus s_4)(s_1 \oplus s_5)(s_2 \oplus s_6)(s_3 \oplus s_7)
\end{aligned}$$

Lets say that $P_1^0 = 1010, P_1^1 = 0110$ *and* $P_1^2 = 0011$ *are recorded responses by the adversary. Using these values, the following linear equations can be written using the bits of the secret key* S*.*

$$
\begin{aligned}
&s_0 = s_0 \ , \\
&s_1 = P_1^0[0] \oplus s_0 = s_0 \ , \\
&s_2 = P_1^1[0] \oplus s_0 = s_0 \ , \\
&s_3 = P_1^0[1] \oplus s_2 = P_1^0[1] \oplus P_1^1[0] \oplus s_0 = \bar{s}_0 \ , \\
&s_4 = P_1^2[0] \oplus s_0 = \bar{s}_0 \ , \\
&s_5 = P_1^2[1] \oplus s_1 = P_1^2[1] \oplus P_1^0[0] \oplus s_0 = \bar{s}_0 \ , \\
&s_6 = P_1^1[2] \oplus s_4 = P_1^1[2] \oplus P_1^2[0] \oplus s_0 = \bar{s}_0 \ , \\
&s_7 = P_1^0[3] \oplus s_6 = P_1^0[3] \oplus P_1^1[2] \oplus P_1^2[0] \oplus s_0 = s_0 \ .
\end{aligned}
$$

Note that using a spacing parameter which is close to the range R is subject to even simpler key recovery attack. In fact, this is the case where a padding is necessary to either P_{odd} or P_{even} to generate a legitimate P_1. Although the authors did not mention the padding scheme explicitly, any padding which does not involve random bits would face SBAP to simpler attacks.

Location Privacy–Untraceability: Observe that the first method of SBAP protocol does not use any update mechanisms. If the adversary always queries the tag with the same nonce, the output will always be the same therefore making the tag traceable. This method is also applicable to the second method of SBAP when P_n is equal to zero.

When P_n is not equal to zero, the second method sends the same P_n value until next successful session. If the adversary queries the tag without completing a session, the tag will always respond with the same value which makes the tag an easy target for tracking.

Desynchronization: This attack is only applicable to the second method of SBAP protocol since the first method does not update the keys. At the end of the second method if the tag updates its key, it sends "OK" response to the reader. Upon receiving this reply, the database also updates its key. If the adversary intercepts this message from reaching to the reader, the tag will update its key whereas the key in the database will not be updated. This will render the tag useless for further interactions.

Replay Attack: In the enhanced method of SBAP protocol when P_n is not equal to zero, the tag does not use the nonce generated by the server. Adversary can obtain access, by simply forwarding one of the tag's P_n message to the reader, and send "OK" when he receives P_2 message from the reader.

This attack will also make the legitimate tag to be desynchronized with the server. It is obvious that the protocol lacks the mutual authentication property.

6.5. Security Flaws in the SSM Protocol

In Section 5.4.1, the description of SSM protocol is given and also it shown that the scheme is vulnerable to a side-channel attack. In this section, we also point out that the SSM protocol suffers from a desynchronization attack which prevents a server from authenticating a legitimate tag in the future sessions.

6.5.1. Desynchronization Attack on SSM

In this part, we show that the SSM protocol is vulnerable to a desynchronization, in which an adversary could set the tag's secret to a random value. Consequently, the tag is desynchronized with the server and authentication of the tag is prevented. The attack could be launched to any tag that is in state $x = x_m$ and requesting secret update.

Note that an attacker could easily put any tag into this state by running Procedure 6.8 given below. As $\mathcal{DB}$ responds with the payload $\{r_R, M_S\}$ to $\mathcal{T}$'s secret update

request, an active adversary could intercept and block the message to the tag. At this point, the adversary sends a forged message $\{r_R, \bar{M}_S\}$ to the tag with

$$\bar{M}_S = M_S \oplus (0||r_1||r_2)$$

where r_1, r_2 are l-bit non-zero strings. Since $\mathcal{T}$ computes $\bar{M}_S \oplus g_k(r_R||r_T||M_1)$ and obtains $(s||\bar{k}||\bar{x}_m)$, where $\bar{k} = k' \oplus r_1$ and $\bar{x}_m = x'_{m'} \oplus r_2$, $H(s) = k$ is verified and the modified secrets are accepted by the tag. Therefore, $\mathcal{DB}$ and $\mathcal{T}$ set their secrets to different values. To be specific, the server stores $(s, k, s', k', x, x'_1, \cdots, x'_{m'})$, while the tag stores $\bar{k}$ and $\bar{x}_m$.

Procedure 6.8. *Set $x = x_m$*

i) $\mathcal{A}$ *arbitrarily chooses a pair of tags* $\mathcal{T}_0$ *and* $\mathcal{T}_1$.
ii) $\mathcal{A}$ *initiates communication with* $\mathcal{T}_0$ *using* TagInit.
iii) $\mathcal{A}$ *transmits some random nonce* r_A *to* $\mathcal{T}_0$.
iv) $\mathcal{A}$ *repeats the previous two steps till* $\mathcal{T}_0$ *responds with SecReq.*

In the next query of the tag, it sends x and M_T to the server, where $x = e_{\bar{k}}(x)$ and $M_T = f_{\bar{k}}(r_R||x)$. When $\mathcal{DB}$ receives x, it would not find a match in the look-up table. Moreover, $\bar{x}_m$ is not in the keyed hash chain list of $e_{\bar{k}}(x)$, so the tag would not request a secure update in the future queries either. Therefore, the reader and tag would be in a desynchronized state where future authentication of the tag is impossible. The steps of this desynchronization attack are given below:

Procedure 6.9. *Desynchronization Attack*

i) $\mathcal{A}$ *takes* $\mathcal{T}_i$ *as its target tag.*
ii) $\mathcal{A}$ *initiates communication with* $\mathcal{T}_i$ *using* TagInit.
iii) $\mathcal{A}$ *transmits some random nonce* r_A *to* $\mathcal{T}_i$.
iv) $\mathcal{A}$ *repeats the previous steps till* $\mathcal{T}_i$ *responds SecReq.*
v) $\mathcal{A}$ *initiates communication with* $\mathcal{DB}$ *using* ReaderInit *and gets* r_R.
vi) $\mathcal{A}$ *initiates communication with* $\mathcal{T}_i$ *using* TagInit.

vii) $\mathcal{A}$ *transmits* r_R *to* $\mathcal{T}_i$.

viii) $\mathcal{A}$ *delivers* $\mathcal{T}_i$ *response* $\{r_R, M_1, M_2, SecReq\}$ *to* $\mathcal{DB}$.

ix) $\mathcal{A}$ *blocks* $\mathcal{DB}$ *response* $\{r_R, M_S\}$ *to* $\mathcal{T}_i$.

x) $\mathcal{A}$ *forges the message with* $\{r_R, \bar{M} = M_S \oplus (0||r_1||r_2)\}$.

xi) $\mathcal{A}$ *sends the modified value,* $\{r_R, \bar{M}_S\}$, *to* $\mathcal{T}_i$.

xii) $\mathcal{T}_i$ *computes* $\bar{M}_S \oplus g_k(r_R||r_T||M_1)$ *and gets* $(s||\bar{k}||\bar{x}_m)$, *where* $\bar{k} = k' \oplus r_1$ *and* $\bar{x}_m = x'_{m'} \oplus r_2$.

xiii) $\mathcal{T}_i$ *verifies* $H(s) = k$ *and updates the secrets as* $\bar{k}$ *and* $\bar{x}_m$, *while they are stored as* k' *and* $x'_{m'}$ *at* $\mathcal{DB}$. *Thus* $\mathcal{DB}$ *will not able to identify or authenticate* $\mathcal{T}_i$ *in the future sessions.*

7. TIMING ANALYSIS OF BACK-END SERVER RESPONSE

In side channel analysis, an attacker utilizes some legitimate function queries in order to collect the corresponding responses of a cryptographic system while it is functioning in a normal mode. If those responses reveal some unwanted information about the secrecy or privacy, this leakage is called side channel information and these responses are called side channels. In this respect, careless deployments of "secure" RFID authentication protocols are not exceptions and subject to side channel attacks.

In this chapter, our goal is to point out a salient missing link in RFID security protocols; namely the back-end server (or the database) role and potential pitfalls or side channels in RFID system realization. Focusing on lightweight RFID security protocols; we examine the server responses for several RFID tags and realize that if the database querying is performed through a static process, the RFID system is subject to timing attacks that could easily jeopardize the system's untraceability criteria. We demonstrate our attack on some well known protocols and outline a countermeasure by precisely describing the database query mechanism. Furthermore, we analyze the success probability of the attack in terms of the system parameters such as the number of tags, number of cryptographic operations has to be carried out and server's computational power.

7.1. Background

7.1.1. Related Work

As mentioned in Section 5.2, the potential security risks in RFID systems hinging the differences in computation time are discussed in a few published work. Juels-Weis [45] introduced the idea that witnessing a reader's success in identifying a tag

could be used in distinguishing two different tags, i.e. breaking the privacy of the protocol.

This ability of the adversary is also touched by Vaudenay [44] and it is formalized in Vaudenay's privacy model. Moreover, Juels-Weis point that computation time of the reader can shed critical light on protocol design.

In [62], Burmester et al. briefly considered timing phenomenon by claiming: "In particular the time taken for each pass must be constant. This can be done by inserting an artificial delay on the trusted server...". Alternatively, Tsudik [10] has investigated the RFID security protocols against timing attacks targetting computations carried in the tag. It is stated that the time variance in tag computations corresponds to different states of the tags that might make them distinguishable.

In [61] Avoine et al. extend the Vaudenay's privacy model by formalizing the computational time of the reader. The authors define a new privacy level – TIMEFUL – which is determined by leaked information from the computational time of the reader and add this notion to the privacy levels of model in [44]. Moreover, they present theoretical solutions to the time problem by assigning boolean decisions about TIMEFUL–PRIVACY of a protocol. However, the parameters that may affect the success of the adversary, such as precision of reader time measurement, have been addressed as an engineering problem.

In this chapter, we present the actual implementation results and probabilistic analysis for successful timing attack. To be more precise, we give the success probability of the attack in terms of the system parameters such as the number of tags, number of cryptographic operations has to be carried out and server's computational power.

7.1.2. Definitions for Database Search

In general, an RFID mutual authentication protocol requires at least three rounds: The reader initiates the communication (Round 1), the tag produces a challenge and

sends it to the reader (Round 2) and the reader replies to the challenge (Round 3). In most lightweight RFID authentication protocols, including [1–5, 11, 52, 64–70][3] the server could need to query its entire database in order to authenticate responder tag in Round 2. In fact, this should not be confused with the simple database search since this querying corresponds to a cryptographic exhaustive search where every single query needs a cryptographic operation having a non trivial time complexity. Therefore, the time complexity of the authentication phase becomes linear in time (i.e. $O(N)$ where N denotes the number of tags in the system). We define these systems as follows:

Definition 7.1. *An RFID system is called Linear-time Authentication System denoted with LAS if its server performs an exhaustive search to identify or authenticate a tag.*

In order to measure the running time differences for different tag searches, it is sufficient to have an exhaustive search process which is identical for each search instance. In fact, for some cases it is possible to achieve some side channels even if the processes are not identical. However, we keep these cases out of our scope and formally define the exhaustive search process as follows:

Definition 7.2. *Let P be the item to be searched, $\mathbb{S}$ be the set of the search space, and $(C_t) = \{c_1, c_2, \ldots, c_t, \ldots\}$ be a sequence on $\mathbb{S}$ (i.e. $(C_t) : \mathbb{N} \mapsto \mathbb{S}$). If (C_t) is one-to-one on $\mathbb{S}$ and $C_t = 0$ for $t > N$, then we call (C_t) as the query sequence for P.*

Note that, the query sequence gives the order of the exhaustive search process. For instance; the query sequence having the general term $C_t = t$ for $t \in \mathbb{N}$ with the initial condition $c_1 = 1$ clearly gives the standard exhaustive search process as shown in Figure 7.1. If this taken as the process for every search item P, it would be possible compare the measurements of search time differences.

Definition 7.3. *An LAS RFID scheme is said to be Static Linear-time Authentication System represented as SLAS if the query sequence for all searched items are identical.*

[3]Weis et al.'s the Randomized Access Control scheme [52] performs an exhaustive search in identification of the tag.

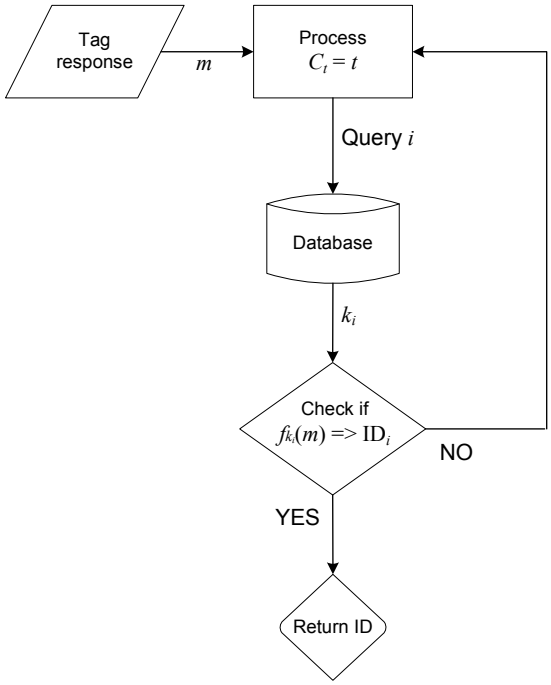

Figure 7.1. Standard exhaustive search having the general term $C_t = t$.

It is equivalent to say that for an $SLAS$ RFID scheme, in tag identification step the order for choosing the candidates amongst the whole database is same for all sessions. As the number of tags in the system increases, variance in elapsed time of the reader responses corresponding to the different RFID tags can be measurable for an $SLAS$ RFID scheme. If an adversary is able to access this time difference[4] , then this information will be used as tool to trace the tags in our attack model.

7.1.3. The Basic Authentication Protocol

BAP is a generic challenge response authentication protocol used as a basis for most of the RFID authentication protocols. Let Δ denote elapsed time between 2nd and 3rd message flow. A step by step description of the BAP that satisfies the LAS properties is given below:

- $\mathcal{R}$ challenges $\mathcal{T}$ with a random nonce r_R.

[4]An adversary may know the amount of time spent for the tag authentication procedures on the server by simply measuring the elapsed time between the tag's authentication request and its response from the server. Note that this may not be a challenge response; it may be the protocol payload showing whether the server is succeeded or not, in identifying a legitimate tag.

- $\mathcal{T}$ chooses a random nonce r_T, computes $M_1 = f_K(r_R, r_T)$, where K is the secret information and different for each tag and $f()$ is a symmetric cryptographic operation. Then transmits the result with r_T to $\mathcal{R}$.
- $\mathcal{R}$ delivers the messages from $\mathcal{T}$ to $\mathcal{DB}$ with r_R.
- $\mathcal{DB}$ maintains a list of pairs $(ID; K_i)$ and identifies $\mathcal{T}$ by performing an exhaustive search of all stored tag records by computing $M_1' = f_{K_i}(r_R, r_T)$ for each stored ID_i in turn, until it finds a match with M_1. If a match is found, $\mathcal{DB}$ regards the ID as the identity of $\mathcal{T}$.
- $\mathcal{R}$ replies to challenge of $\mathcal{T}$.

Note that, throughout this text, BAP will be used in description of our attack model and will be a basis for many RFID authentication protocols that are vulnerable to the attack.

7.2. Description of the Timing Attack

Timing attacks provide an attacker with secrets maintained in a security system by measuring the time it takes the system to respond to various queries. For instance, Kocher [71] designed a timing attack to recover secret keys used for RSA decryption. In addition, Brumley and Boneh presented a timing attack on unprotected OpenSSL implementations and showed that such attack was practical, that is, an attacker could measure the response-time variances of a secure Web server and could derive that servers RSA private key [72]. With a similar approach: Since in different steps of the RFID protocols, tags and the server execute different processes, if time taken to execute these steps differ based on the input of tags' state or responses, an attacker can attempt to mount a timing attack to distinguish the tags by analyzing the time variances corresponding to their input. So, with precise measurements of the time difference, an attacker can easily trace the tags and break the untraceability property of the protocol.

Untraceability notion is briefly described in 3.1.1. The Juels–Weis model in [45] characterizes a very strong adversary with a relatively simple definition and according to this model untraceability is defined in terms of privacy experiments by which an adversary could distinguish two different tags within the limits of its computational power and functionality-call bounds. Throughout this text, we adopt the terms and notions of [45] to our needs. In this privacy model, two of the available functions for an adversary are ReaderInit and TagInit. When receiving a ReaderInit message, $\mathcal{R}$ initializes a new session and returns the first challenge of an interactive challenge-response protocol. On the other side, by receiving TagInit message a tag $\mathcal{T}$ is involved in the corresponding protocol session and it may respond to a protocol message or challenge. For an RFID system $\mathcal{S}$, the privacy experiment, $\mathbf{Exp}_{\mathcal{S}}^{priv}$, consists of the following two phases:

i) *Learning Phase:* In this phase, according to Juels–Weis privacy model [45], an adversary $\mathcal{A}$ might initiate a communication with the reader $\mathcal{R}$ (ReaderInit) or a tag $\mathcal{T}$ (TagInit). Also, $\mathcal{A}$ has the ability to modify, insert or delete messages that agree with the corresponding protocol's procedures. In other words, $\mathcal{A}$ controls the communication channel between $\mathcal{R}$ and each $\mathcal{T}$ and may make any ReaderInit or TagInit calls in any interleaved order without exceeding its parameter bounds.

ii) *Challenge Phase:* In this phase the adversary $\mathcal{A}$ selects two tag candidates $\mathcal{T}_0$ and $\mathcal{T}_1$ and tests these with the identifers ID_0 and ID_1 respectively. Depending on a randomly chosen bit $b \in \{0, 1\}$, $\mathcal{A}$ is given a challenger identifier ID_b from the set $\{ID_0, ID_1\}$. That is $\mathcal{A}$ is given access to one of these tags randomly, called $\mathcal{T}_b$. The adversary might again interact with the reader and the tags. Eventually, at some point $\mathcal{A}$ decides to terminate the experiment and returns the bit $\hat{b}$ as its guess for the value of b. The success of $\mathcal{A}$ in guessing b is equivalent to its success of breaking the untraceability, and is quantified as $\mathcal{A}$'s advantage in distinguishing the tag's identity compared to random selection. This is expressed formally as:

$$\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A},\mathcal{S}}(k) = |\Pr[\hat{b} = b] - \frac{1}{2}|, \tag{7.1}$$

where k is the security parameter (i.e. the bit length of the unknown secret ID). An RFID system, $\mathcal{S}$, achieves untraceability if $\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A},\mathcal{S}}(k) < \varepsilon(k)$ for some negligible function $\varepsilon()$.

It is equivalent to say, an attack is successful in tracing the tags if the adversary has a non-negligible advantage in guessing the selected tag. As an illustrative example, assume that the probability of a correct guess is 1/2 (i.e. $\Pr[\hat{b} = b] = \frac{1}{2}$). In this case, $\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A},\mathcal{S}}(k)$ is zero. Thus, the adversary, $\mathcal{A}$, does not have any advantage in guessing b.

Definition 7.4. *Let γ denote the attacker's precision in distinguishing elapsed time of the reader responses and expressed in terms of seconds, i.e. timing resolution.*

In our attack model, we suppose the examined protocol is based on $SLAS$ BAP which we call SLBAP and for the privacy experiments, the adversary can follow the steps below:

In the learning phase, two tags $\mathcal{T}_0$ and $\mathcal{T}_1$ are randomly selected and then the adversary observes successful authenticated protocols between the reader and the tags and notes respective elapsed time of the reader responses as Δ_0 and Δ_1.

- $\mathcal{A}$ *randomly chooses a pair of distinct tags* $\mathcal{T}_0$ *and* $\mathcal{T}_1$.
- $\mathcal{A}$ *initiates communication with* $\mathcal{R}$ *using* ReaderInit *and gets* r_{R_0}.
- $\mathcal{A}$ *initiates communication with* $\mathcal{T}_0$ *using* TagInit.
- $\mathcal{A}$ *transmits* r_{R_0} *to* $\mathcal{T}_0$.

- *$\mathcal{A}$ delivers $\mathcal{T}_0$ response to $\mathcal{R}$.*
- *$\mathcal{A}$ measures elapsed time, Δ_0, between 2nd and 3rd message flow.*
- *$\mathcal{A}$ initiates communication with $\mathcal{R}$ using* ReaderInit *and gets r_{R_1}.*
- *$\mathcal{A}$ initiates communication with $\mathcal{T}_1$ using* TagInit.
- *$\mathcal{A}$ transmits r_{R_1} to $\mathcal{T}_1$.*
- *$\mathcal{A}$ delivers $\mathcal{T}_1$ response to $\mathcal{R}$.*
- *$\mathcal{A}$ measures elapsed time, Δ_1, between 2nd and 3rd message flow.*

Notice that in our attack, $\mathcal{A}$ always provides an answer. Thus in the challenge phase, if $|\Delta_0 - \Delta_1| < \gamma$, he makes a random guess for the selected tag $\mathcal{T}_b^*$. On the other hand if $|\Delta_0 - \Delta_1| \geq \gamma$, the adversary only observes a successful authentication between the legitimate reader and the selected tag and records time duration between the second and the third message flow, call it Δ_*. If $\Delta_* \approx \Delta_0$ the challenge tag is $\mathcal{T}_0$, otherwise the selected tag is $\mathcal{T}_1$.

- *If $|\Delta_0 - \Delta_1| < \gamma$, then $\mathcal{A}$ randomly flips a coin for the value of b.*
- *If $|\Delta_0 - \Delta_1| \geq \gamma$, then:*
  - *i) $\mathcal{A}$ takes $\mathcal{T}_0$ and $\mathcal{T}_1$ as its challenge candidates.*
  - *ii) $\mathcal{A}$ initiates communication with $\mathcal{R}$ using* ReaderInit *and gets r_R.*
  - *iii) $\mathcal{A}$ transmits r_R to the selected tag $\mathcal{T}_b^*$.*
  - *iv) $\mathcal{A}$ delivers $\mathcal{T}_b^*$ response to $\mathcal{R}$.*
  - *v) $\mathcal{A}$ measures elapsed time, Δ_*, between 2nd and 3rd message flow:*
  - *vi) If $\Delta_* \approx \Delta_0$ $\mathcal{A}$ guesses $b = 0$ and decides $\mathcal{T}_b^* = \mathcal{T}_0$, otherwise guesses $\mathcal{T}_b^* = \mathcal{T}_1$.*

Lemma 7.5. *Suppose in exhaustive search of database for each item m cryptographic operations are evaluated and each operation can be carried out in β seconds. Let n denote the maximum index difference of two candidate elements c_x and c_y of the query sequence (C_t), related to the tags $\mathcal{T}_i$ and $\mathcal{T}_j$ respectively, such that the adversary cannot distinguish the tags by using the above attack. It is equivalent to say that:*

$n \triangleq \max_{x,y\in[1,N]} |x-y|$ *such that* $c_x = item_i$, $c_y = item_j$, $\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A},SLBAP}(k) < \varepsilon(k)$. *Then we can express* $n = \lfloor \frac{\gamma}{m\cdot\beta} \rfloor$.

Proof. If $|\Delta_i - \Delta_j| < \gamma$ then the $\mathcal{A}$ cannot realize time difference, so not have a non-negligible advantage in distinguishing the tags. We know that $|\Delta_i - \Delta_j| = |x-y| \cdot m \cdot \beta$, so $|x-y| \cdot m \cdot \beta < \gamma$ must be satisfied to give a negligible advantage to the adversary. Hence the maximum index difference n can be expressed as:

$$n = \lfloor \frac{\gamma}{m \cdot \beta} \rfloor. \tag{7.2}$$

□

Definition 7.6. *For privacy experiment* $\mathbf{Exp_z}$, *suppose* $\mathcal{T}_i$ *and* $\mathcal{T}_j$ *are selected and* $c_x = item_i$, $c_y = item_j$ *denote the respective candidates in the exhaustive search process. Then the discrete random variable* $Q^{\mathbf{Exp_z}}$, *describing the probability of being* $|x-y| > n$, *i.e.* $\mathcal{A}$ *can sense the time difference, is defined as below:*

$$Q^{\mathbf{Exp_z}} = \begin{cases} 1 if\ |x-y| > n \\ 0 \quad otherwise \end{cases}$$

Proposition 7.7. *If* N *denotes number of tags in the database, then for any selected tags* $\mathcal{T}_i$ *and* $\mathcal{T}_j$, *the* $\mathcal{A}$*'s advantage by considering the described attack is expressed as:*

$$\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A},\mathcal{SLBAP}}(k) = \frac{1}{2}\left[1 - \frac{1}{N}\sum_{i=1}^{N} \frac{\min(i-1,n)+\min(N-i,n)}{N-1}\right]$$

Proof. Success probability of the attack depends on $\Pr[Q^{\mathbf{Exp}} = 1]$. If $Q^{\mathbf{Exp}} = 0$, i.e. $|x-y| \leq n$, the adversary has zero advantage since he could just as well have flipped a coin to make the guess, which would have given him the same probability of correct guessing. On the other hand if $Q^{\mathbf{Exp}} = 1$, he can recognize the time difference of the

reader responses and makes a correct guess for the privacy experiment with maximum advantage. Therefore the correct guess probability can be expressed as:

$$\begin{aligned}\Pr[\hat{b}=b]&=\sum_{\forall a\in\{0,1\}}\Pr[\hat{b}=b|Q^{\mathbf{Exp}}=a]\times\Pr[Q^{\mathbf{Exp}}=a]\\&=\Pr[\hat{b}=b|Q^{\mathbf{Exp}}=0]\times\Pr[Q^{\mathbf{Exp}}=0]\\&\quad+\Pr[\hat{b}=b|Q^{\mathbf{Exp}}=1]\times\Pr[Q^{\mathbf{Exp}}=1].\end{aligned}\tag{7.3}$$

The marginal probabilities of $\Pr[Q^{\mathbf{Exp}}]$ is derived as follows:

$$\Pr[Q^{\mathbf{Exp}}]=\begin{cases}\Pr[Q^{\mathbf{Exp}}=0]=\Pr[|x-y|\leq n]\\\qquad\qquad=\frac{1}{N}\sum_{i=1}^{N}\frac{\min(i-1,n)+\min(N-i,n)}{N-1},\\\Pr[Q^{\mathbf{Exp}}=1]=1-\Pr[Q^{\mathbf{Exp}}=0]\\\qquad\qquad=1-\frac{1}{N}\sum_{i=1}^{N}\frac{\min(i-1,n)+\min(N-i,n)}{N-1}\end{cases}\tag{7.4}$$

Notice that $\Pr[\hat{b}=b|Q^{\mathbf{Exp}}=1]=1$ and $\Pr[\hat{b}=b|Q^{\mathbf{Exp}}=0]=\Pr[\textit{random coin flip}]=\frac{1}{2}$. Thus, if we replace these values together with those given in (7.4) in (7.3), we obtain:

$$\Pr[\hat{b}=b]=1-\frac{1}{2N}\sum_{i=1}^{N}\frac{\min(i-1,n)+\min(N-i,n)}{N-1}\tag{7.5}$$

From (7.1) we obtain:

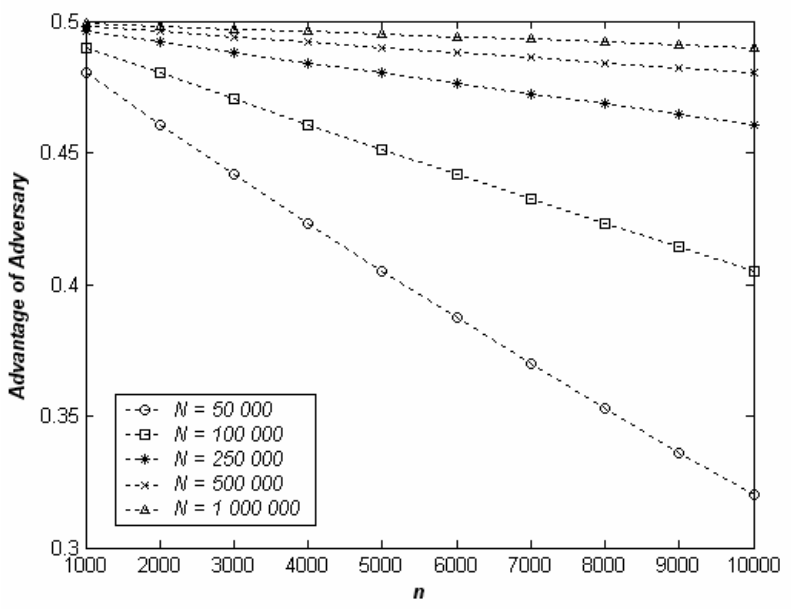

Figure 7.2. Advantage of the adversary for different n and N.

$$\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A},\mathcal{SLBAP}}(k) = \frac{1}{2}\left[1 - \frac{1}{N}\sum_{i=1}^{N}\frac{\min(i-1,n)+\min(N-i,n)}{N-1}\right] \tag{7.6}$$

□

In Figure 7.2, advantage of the adversary for different n and N values is shown. Note that as $N \gg n$, $\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A}}(k)$ becomes closer to $1/2$, which is the maximum advantage.

Remark 7.8. *Since* $\frac{2n}{N} > \frac{2n}{N-1} > \frac{1}{N}\sum_{i=1}^{N}\frac{\min(i-1,n)+\min(N-i,n)}{N-1}$, $\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A},SLBAP}(k) > \frac{1}{2} - \frac{n}{N}$. *Therefore, for* $N \gg n$ *advantage of an adversary can be approximated as*

$$\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A},SLBAP}(k) \approx \frac{1}{2} - \frac{n}{N}. \tag{7.7}$$

In order to illustrate the realization of the presented attack, we consider a real life scenario in a library, where tags are used to identify books and the protocol is an $SLBAP$.

Example 7.9. *We consider an SLBAP RFID scheme installed in a public library to substitute bar codes on the books. Suppose the library has got about 1 million of books,* $N = 1\ 000000$*, and we assume that there are about* 100000 *candidates between* c_x *and* c_y *as the candidates related to some books* $book_i$ *and* $book_j$ *respectively in the database, i.e.* $|x - y| = 100000$*. Besides, we suppose that timing resolution of an adversary who attempts to apply the proposed attack is one millisecond and also to identify a single tag the server needs a single cryptographic operation which is performed in one microsecond. Thus,* $\beta = 10^{-6}$*,* $\gamma = 10^{-3}$ *and* $m = 1$*.*

In the light of given information, it is obvious that $book_i$ *identification process will be faster than those of* $book_j$*'s. Moreover by using (7.2) we obtain* $n = 1000$*. Because* $n < |x - y| = 100000$*, the adversary can easily distinguish these two tags by comparing the elapsed time between* $2nd$ *and* $3rd$ *rounds of the protocols for each tag.*

7.3. Analysis of the Some RFID Schemes

In this section, we examine some RFID privacy schemes as the SM, RKKW, DPLK and OSK protocols. In our analyzes we do not consider whether or not these protocols have been cryptanalyzed previously by some different type attacks like Denial of Service, Tag Impersonation, Replay attacks etc. rather we focus on realization of our attack for these schemes. The common point of these models is that how the candidates are chosen from database in the search process is not defined exactly and this makes them vulnerable against our attack. In the following parts, we assume that the system relies on a single computer which takes 2^{-20} seconds to carry out a cryptographic operation, i.e. $\beta = 2^{-20}$, the number of tags in the system, N, is 2^{20} and $\gamma = 2^{-10}$ seconds (i.e. ≈ 1 ms), unless otherwise is stated. Furthermore, below we only give the three message flows of the protocol since these parts interest us. Update of the secrets and other details can be found in the corresponding study.

7.3.1. The Timing Attack on the SM Protocol

The SM protocol is described previously in 4.1.6. The steps of the proposed attack presented in Section 7.2 can be directly applied on SM protocol. In exhaustive search process for each candidate two cryptographic operations, $f_{t_j}(r_1 \oplus M_1 \oplus t_j)$ and $f_{\hat{t}_j}(r_1 \oplus M_1 \oplus \hat{t}_j)$, are executed, so $m = 2$. By using (7.2), we obtain $n = 512$. Also from (7.6), $\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A},SM}(k) = 0.4995$ is computed.

Besides, in [11] an improvement to SM protocol is proposed and to the best of our knowledge it has received no attacks yet. However, same weakness also exists in this protocol and it can be easily broken with our attack.

7.3.2. The Timing Attack on the RKKW Protocol

In Section 4.1.3, we give a brief description of the RKKW protocol. The scheme is vulnerable to our attack as the back-end database is required to perform an ID search to find the specific information related to the tag requesting authentication.

In the brute force search of server for each candidate one cryptographic operation is done, so $m = 1$. If we replace the values in (7.2), we get $n = 1024$ and this leads to $\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A}}(k) = 0.499$ for our attack.

7.3.3. The Timing Attack on the DPLK Protocol

The proposed attack can be applied on DPLK protocol, which is reviewed in Section 4.1.4. Since CRC computation consumes less time than hash or other symmetric encryption we assume server can evaluate a CRC operation in 2^{-28} seconds so $\beta = 2^{-28}$. Moreover, for each entry from database one CRC is calculated; $m = 1$. For these values $n = 2^{18}$ is evaluated and from (7.6) it means $\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A}}(k) = 0.28$ for our attack.

7.3.4. The Timing Attack on the OSK Protocol

In this part, we examine the OSK protocol, described in Section 4.1.1. Notice that OSK protocol does not exactly fit to the steps of BAP, because after identification of the tag the reader does not send any message to the tag. Hence, how we can apply the proposed attack on OSK could arise in our minds. Although the reader does not respond in the third message flow, as presented in [45] the adversary can record the elapsed time till tag identification is realized by observing a validation event. For example, opening a door with a proximity card or acceptance of a payment card can be used as validation events. Nevertheless, one can argue that the work of attacker is more difficult than the cases of previous protocols. Therefore, we assume that the attacker's time distinguishing capability may be lower and set $\gamma = 1$ seconds. In addition, according to OSK protocol for each trial 2δ hash operations are computed. For $\delta = 128$ (a suited size for δ could be 128 as mentioned in [73]), we get $m = 256$ and by using (7.2) $n = 4096$ is evaluated. As a result, from the equation for advantage of the adversary $\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A},OSK}(k) = 0.496$ is obtained.

7.3.5. Experimental Results

We experimentally examine the capabilities of the proposed timing analysis by implementing the RKKW and DPLK protocols. These are chosen for simplicity though similar experimental result can be achieved for other mentioned protocols in this section. The source code was compiled using the MS Visual C# compiler with default optimizations. All of the experiments were run under the configuration shown in Table 7.1. We used random keys generated by .net System.Random class key generation routine. We measured the time using stopwatch class and take the averages in order to measure elapsed time accurately.

In the implementation of the RKKW protocol, we use the standard SHA-1 in MS .net System.Security.Cryptography class where we write a custom class for CRC implementation used in the DPLK protocol. Our CRC routine uses a lookup table which reflects the lightweight feature of the protocol as it outperforms the SHA-1

Table 7.1. The configuration used in the experiments.

operating system	Windows XP SP3
CPU	Intel Core2 Quad Q8300
compiler	MS Visual C#
cryptographic library	.net System.Security.Cryptography

implementation. In Tables 7.2 and 7.3, timings for exhaustive search steps of the protocols are tabulated, where "index difference" stands for $|x - y|$ as mentioned in Lemma 1. As formulated in the previous section timing attacks could be very powerful in case of a poor search process is chosen.

Table 7.2. Timing attack on the RKKW protocol.

index difference	Δ time
500	12 ms
1000	27 ms
10K	298 ms
100K	3033 ms
500K	14852 ms
1M	29780 ms

7.4. Countermeasures against the Proposed Timing Attack

Before giving our countermeasure against to the proposed attack, we want to elaborate on some other obvious but not efficient techniques that remedy the security flaw.

With consideration of the previous parts, intuitively one can provide security against the proposed timing attack by realizing the condition that the server response time variation for different tags is negligible, in other words the server responds with

Table 7.3. Timing attack on DPLK protocol.

index difference	Δ time
500	0.7 ms
1000	1.1 ms
10K	20 ms
100K	97 ms
500K	417 ms
1M	807 ms

an equal time and this is same for all tags. This condition can be achieved by using look-up tables as mentioned in [61] or artificially padding the delay in reader responses for all tags as reported in [62]. For the lookup-table model the server stores all possible answers of tags that are pre-computed previously. Thus, in authentication phase the server avoids to make an exhaustive search, instead responds in constant-time. Note that although this solution fixes the problem, constructing such a scheme with satisfying security and implementation requirements is impractical. In fact, if such a system exists, then clearly the use of exhaustive search in tag identification would be abandoned. On the other hand, inserting an artificial delay at the server side, determined by the worst case time, definitely eradicates the security flaw. However, this is clearly undesirable, because it reduces the efficiency of the overall system.

Our timing analysis and the experimental results of the previous section exploit the use of an *SLAS* RFID scheme which uses a static exhaustive search process on server authentication. However, if a dynamic search process is employed the described timing analysis would fail in measuring the running time differences for different tag searches. In this respect, the simplest countermeasure to avoid such attacks would be simply changing the the starting point of the exhaustive search process. We formulate this countermeasure as follows:

Countermeasure 7.10. *Let an RFID system implements an SLAS scheme, having the same query sequence* (C_t) *for all of its search items such as* $\{P_0, P_1, \ldots, P_l\}$ *for some positive integer* l*. Choosing non-identical random query sequences* (C_t) *for all search items gives the desired protection for the described attacks.*

Although, Countermeasure 7.10 gives a wide range of selection having different implementation complexities, naturally simplest countermeasures come from setting minimal differences between query sequences. Observe that query sequences (C_t) can also be seen as permutations on N items where N is the number tags. Thus, composing the query sequences with the following constant cyclic permutation π_j gives non-identical shifted query sequences

$$\pi_j(i) = i + r_j \bmod N, \text{ for random } r_j, j = 0, 1, \ldots, l \text{ and } 0 < i \leq N$$

In other words, for all search items $\{P_0, P_1, \ldots, P_l\}$ we have the following general terms for the corresponding query sequences

$$C_{\pi_j(t)}, \text{ for } j = 0, 1, \ldots, l.$$

In fact, this modification corresponds to randomly selection of the starting point of the exhaustive search process. Since we use different query sequences for different tag searches, for any selected two tags the index difference will also be different. Therefore, timing attacks would fail in measuring the time differences.

8. SECURITY ANALYSIS OF RFID DELEGATION PROTOCOLS

RFID tag delegation enables a centralized back-end server to delegate the right to identify and authenticate a tag to specified readers. This should be used to mitigate the computational load on the server side and also to solve the issues in terms of latency and dependency on network connectivity.

In this chapter, we describe a basic RFID delegation architecture and then under this model we investigate the security of some RFID delegation protocols. Specifically, we focus our analysis on a new delegation protocol which we call SMD (Song Mitchell Delegation) due to name of the authors. We point out security flaws that have gone unnoticed in the design and present two attacks namely, a tag impersonation attack and a de-synchronization attack against it. We also discover a subtle flaw by which a delegated entity can still keep its delegation rights after the expire of them – this infringes security policy of the scheme. More precisely, we show that the protocol will be still vulnerable to previously mentioned attacks, even if the back-end server ends the delegation right of a delegated reader and update the secrets of the delegated tags. After giving the security weaknesses of the SMD, we claim that some presented flaws are an inherent and can thus be easily generalized for some other delegation protocols. Then, we outline some countermeasures that should be taken into consideration in designing an RFID delegation protocol. Finally, to eliminate the vulnerabilities of SMD we propose our revisions.

8.1. RFID Tag Delegation

Most existing RFID schemes have a common problem that in order to identify or authenticate a responder tag, the server has to perform an exhaustive search over the list of all tag entries in the database requiring a linear computational complexity. Such a tedious search procedure brings up the so called scalability problem that many systems have suffered from. Delegation is one possible solution to reduce the cost of

exhaustive search and enhance the scalability of the system. It enables a back-end server to delegate the right to identify and authenticate a tag to a specified entity, such as a reader, i.e. the delegated entity is permitted by the back-end server to identify a number of tags without referring to the online database [47, 74]. In addition to reducing the computational load on the server, delegation mechanism can be a solution for the case that the back-end server is unavailable for some reason, such as network connectivity failure and all the reading operations of the tags relying on back-end server are stopped [75].

Delegation may be permanent or temporarily: In the first case, the readers have permanent delegation for the tags in their read range and the back-end server is contacted only when a new tag arrives or an old tag leaves the system [74]. In the second case, the back-end server temporarily transfers the right to interact with a set of tags for a limited number of queries or a limited time period and updates or revokes the delegation capability according to a delegation policy [47].

8.2. The Basic RFID Delegation Protocol

Our basic RFID system is composed of four components: A centralized back-end server, a set of on-line readers, a set of off-line readers and RFID tags as shown in Figure 8.1. Below, we give main functions of each unit.

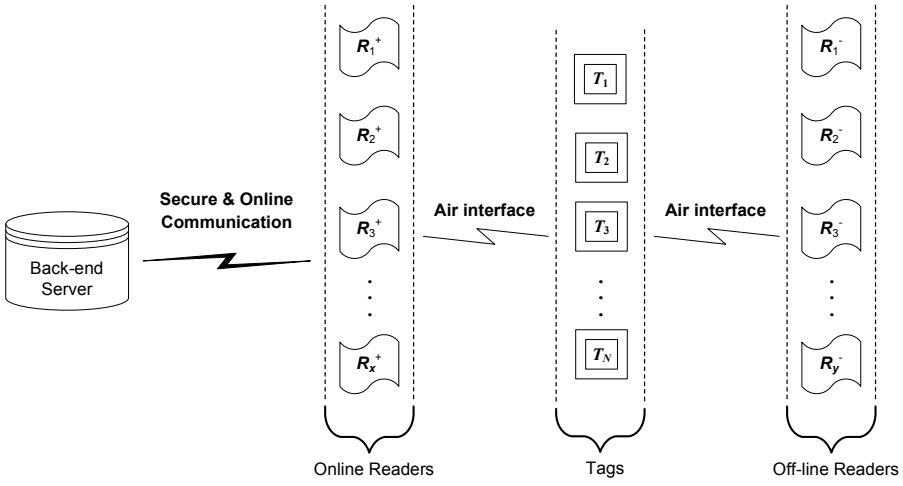

Figure 8.1. The basic RFID delegation architecture.

The Back-end Server: It stores related information of the RFID tags in the system such as product and price information etc. and also their secrets. It receives data from the readers and provides certain services like secret key update and authentication of the tags.

The Online Reader: This unit is directly connected to the back-end server with a secure channel. It communicates with the RFID tags through air interface and may transmit response from tag to the back-end server and from server to the tag.

The Off-line Reader: The off-line readers have the same properties of online readers except that they do not necessarily have a direct connection to the back-end server. Also, they can provide some limited services of the back-end server such as tag identification. In the rest of the paper we assume that a delegated reader is an off-line reader.

The Tag: In our architecture we suppose the RFID tags are passive RFID tags that are low cost microchips with weak computational capabilities and an antenna providing wireless data transmission. They transmit information through the RF interface when powered by a reading device.

According to our basic RFID delegation architecture, a registered tag can be authenticated by the delegated readers that have right to authenticate or by the back-end server itself through one of the online readers as illustrated in Figure 8.1.

8.3. The Song–Mitchell RFID Delegation Protocol

In [47], Song and Mitchell have proposed a scalable RFID authentication protocol which claims to assure the untraceability, authentication, and robustness against replay

and spoofing attacks. They propose that this scheme can also be used as an RFID delegation protocol which we call it as the SMD protocol. Besides, the authors assume that the channel between the reader and the back-end server is secure, hence $\mathcal{DB}$ is considered as a single entity consisting of the back-end server and the online reader. The operation of the protocol can be divided into three cases:

- Case 1: For each of the first $m-1$ queries of a tag, where m is a positive integer determined by the server, the protocol process requires just two message flows and only involves tag authentication. To authenticate a tag, the reader/server searches a look-up table, thereby taking only $O(1)$ work to identify and authenticate a tag, without needing a linear search. Indeed, thanks to this feature a delegated reader or the server can authenticate a registered tag in constant time and hence the scheme provides desirable scalability properties.
- Case 2: On the mth query of a tag, as in Case 1 the tag is authenticated in constant time by the server. However, the server realizes that all identifiers in the look-up table for the corresponding tag were used in the previous queries. In order to maintain Case 1 operation, the server needs to update the secrets of the tag and generate new identifiers. Also following the update process, the server sends an additional message to initiate a secret update process at the tag side. At the end of these steps, for the first $m'-1$ queries of the tag, the protocol again operates as in Case 1, where m' is new value of m.
- Case 3: If a tag is queried more than m times, which should not normally happen, then tag produces two messages with demanding a secret update. In this case, the authentication of the tag imposes a linear search with complexity $O(N)$, where N is number of tags in the back-end database. After this, as in Case 2, the server updates the secrets of the tag and replies to the tag with an additional message to invoke a secret update procedure at the tag side.

A summary of SMD protocol is seen in Figure 8.2, where an in depth description could be find in [47]. Initially, a secret l-bit string, s_j, and its hash value (computed by the server), $k_j = h(s_j)$, is attached to the tag entries $\mathcal{T}_j \in \mathcal{DB}$. Moreover, for every tag $\mathcal{T}_j$, $\mathcal{DB}$ stores a hash-chain $\{x_0, x_1, \cdots, x_m\}$ where m is a positive integer, $x_t = e_k(x_{t-1})$

Data Base / Reader $[\hat{s}, \hat{k}, s, k, x_0, \cdots, x_i \cdots, x_m]$		**Tag** $[k, x, c]$
Generate r_R	r_R ⟶	**If** $c \neq 0$, $M_T = f_k(r_R \| x)$ $x \leftarrow e_k(x)$, $c \leftarrow c - 1$
	r_R, x, M_T ⟵	
Case 1: Search for $x_t = x$ in the DB Check $M_T = f_k(r_R \| x_{t-1})$ **Case 2:** **If** $x = x_m$, $M_S = g_k(r_R \| M_T) \oplus (s \| k' \| m')$ Update secrets for Tag $\hat{s} \leftarrow s, \hat{k} \leftarrow k, s \leftarrow s', k \leftarrow k', x_0 \leftarrow x$ $x_t(1 \leq t \leq m) \leftarrow x'_t(1 \leq t \leq m')$	r_R, M_S ⟶	$(s \| k' \| x'_{m'}) = M_S \oplus g_k(r_R \| M_T)$ **If** $h(s) = k$, $k \leftarrow k'$ and $c \leftarrow m'$
Case 3: Search for $x = x_m$ or x_0 for which $M_1 = f_k(r_R \| (M_2 \oplus x))$ $r_T = M_2 \oplus x$ **If** $x = x_m$, $M_S = g_k(r_R \| r_T) \oplus (s \| k' \| m')$ **If** $x = x_0$, $M_S = g_{\hat{k}}(r_R \| r_T) \oplus (\hat{s} \| k \| m)$ Update secrets for Tag $s \leftarrow s', k \leftarrow k', x_0 \leftarrow x$ $x_t(1 \leq t \leq m) \leftarrow x'_t(1 \leq t \leq m')$	r_R, M_1, M_2 ⟵ *SecReq* r_R, M_S ⟶	**If** $c = 0$, Generate r_T $M_1 = f_k(r_R \| r_T)$ $M_2 = (r_T \oplus x)$ $(s \| k' \| m') = M_S \oplus g_k(r_R \| r_T)$ **If** $h(s) = k$, $k \leftarrow k'$ and $c \leftarrow m'$

Figure 8.2. The Song Mitchell scalable RFID authentication protocol.

for $1 \leq t \leq m$ and x_0 is a random l-bit string. In case of a need to resynchronize the tag, $\mathcal{DB}$ further stores $\{\hat{s}_j, \hat{k}_j\}$ as the most recent secrets assigned to $\mathcal{T}_j$. On the other hand, each $\mathcal{T}_j$ stores k, x and a counter c, where x is initially set to x_0 and c is set to m.

In case of $c \neq 0$ whenever $\mathcal{T}$ receives r_R, the tag calculates $M_T = f_k(r_R||x)$ and updates its identifier x with $e_k(x)$ and its counter c to $c-1$. Then, $\mathcal{T}$ transmits r_R, x and M_T to $\mathcal{DB}$. If the updated c is equal to 0, $\mathcal{T}$ waits for $\mathcal{DB}$ response, keeping r_R and M_T in short term memory.

On the other side, when $\mathcal{DB}$ receives $\{r_R, x, M_T\}$ payload, firstly, $\mathcal{DB}$ searches its look-up table for a value x_t matching to the received x. If nothing matches, the session terminates. Otherwise, $\mathcal{DB}$ identifies the tag and checks whether $f_k(r_R||x_{t-1})$ equals the received M_T, where k is the identified tag $\mathcal{T}$'s key. If this verification succeeds and $x \neq x_m$, then the authentication session terminates successfully. However, if $x = x_m$, then the server starts a regular secret update process in which $\mathcal{DB}$ picks a random l-bit string s' and an integer m', and computes a key $k' = h(s')$ and a sequence of m' identifiers $x'_t = e_{k'}(x'_{t-1})$ for $1 \leq t \leq m'$, where x'_0 is set to x. Then, $\mathcal{DB}$ computes $M_S = g_k(r_R||M_T) \oplus (s||k'||m')$, and sends r_R and M_S to $\mathcal{T}$. Finally, $\mathcal{DB}$ updates the set of stored values for $\mathcal{T}$ from $(\hat{s}, \hat{k}, s, k, x_0, x_1, \cdots, x_m)$ to $(s, k, s', k', x, x'_1, \cdots, x'_{m'})$.

In the normal state, if the tag $\mathcal{T}$ receives $\{r_R, M_S\}$ payload, $(s||k'||m') = M_S \oplus g_k(r_R||M_T)$ is evaluated. Next, $\mathcal{T}$ authenticates $\mathcal{DB}$ and updates k and c with k' and m' respectively, if $h(s)$ is equal to k. The secret update session is then ended successfully.

On the other hand, the process is slightly different if the tag is in a rare abnormal state which is the case $c = 0$ when $\mathcal{T}$ receives the nonce r_R. The process in this state which is also illustrated in the lower box of Figure 8.2 starts with the generation of a random number r_T and values $M_1 = f_k(r_R||r_T)$ and $M_2 = r_T \oplus x$. The tag, then sends r_R, M_1 and M_2 back to $\mathcal{DB}$ with a request for an update of the shared secrets as *SecReq* and waits for the server response, keeping r_R, r_T and M_1 in its short term memory.

On the server side, once $\mathcal{DB}$ receives the $\{r_R,\ M_1,\ M_2, SecReq\}$ payload, the server initiates an irregular secret update process. Firstly, $\mathcal{DB}$ searches its look-up table for a value $x = x_m$ or $x = x_0$ for which $M_1 = f_k(r_R||(M_2 \oplus x))$. Observe that this search corresponds to the usual linear search and particularly carried in the abnormal state. If such a value is found, $\mathcal{DB}$ authenticates the tag. Otherwise, the session terminates.

In this step, $x = x_m$ means that even if $\mathcal{T}$ sent $x = x_m$ to $\mathcal{DB}$ in the previous session, $\mathcal{DB}$ would not receive it correctly, hence, the shared secrets are not fresh. In this case, $\mathcal{DB}$ performs the following steps:

- $\mathcal{DB}$ chooses a random l-bit string s' and an integer m', and computes a key $k' = h(s')$ and a sequence of m' identifiers $x'_t = e_{k'}(x'_{t-1})$ for $1 \leq t \leq m'$, where x'_0 is set to x.
- $\mathcal{DB}$ computes $r_T = M_2 \oplus x$ and $M_S = g_k(r_R||r_T) \oplus (s||k'||m')$ and sends r_R and M_S to $\mathcal{T}$.
- $\mathcal{DB}$ assigns the set of values $\{s, k, s', k', x, x'_1, \cdots, x'_{m'}\}$ to the tag $\mathcal{T}$, updating the previously stored values $\{\hat{s}, \hat{k}, s, k, x_0, x_1, \cdots, x_m\}$.

However, if $x = x_0$ it means that M_S did not reach $\mathcal{T}$ correctly in the previous session, and thus $\mathcal{T}$ could not update its secrets, although $\mathcal{DB}$ did. In this case, the following steps are performed by the server. $\mathcal{DB}$ computes $r_T = M_2 \oplus x$ and $M_S = g_{\hat{k}}(r_R||r_T) \oplus (\hat{s}||k||m)$, and sends r_R and M_S to $\mathcal{T}$. On the tag side, since M_1 and r_T were kept previously, when $\mathcal{T}$ receives r_R and M_S, it calculates $(s||k'||m') = M_S \oplus g_k(r_R||r_T)$. Following this step, if $h(s)$ is equal to k, $\mathcal{T}$ authenticates $\mathcal{DB}$ and ends the session successfully after updating k and c to k' and m' respectively.

For delegation of a tag $\mathcal{T}$ to a reader $\mathcal{R}_i^-$, the server firstly transfers the secret k and the identifiers $\{x_0, x_1, \cdots, x_m\}$ for $\mathcal{T}$ to the $\mathcal{R}_i^-$ via a secure channel. After this, $\mathcal{R}_i^-$ has right to authenticate $\mathcal{T}$ a maximum of m times. The authentication steps of the SMD scheme is described as given below:

i) $\mathcal{R}_i^-$ transmits a random nonce r_R to $\mathcal{T}$.

ii) If $c \neq 0$, $\mathcal{T}$ computes $M_T = f_k(r_R||x)$, $x \leftarrow e_k(x)$, $c \leftarrow c - 1$ and transmits r_R, x and M_T to $\mathcal{R}_i^-$. However if $c = 0$, it means that $\mathcal{R}_i^-$ has no more right to authenticate the tag. Thus, secrets and identifiers of $\mathcal{T}$ need to be refreshed. In this case, $\mathcal{T}$ initiates the secret update process.

iii) $\mathcal{R}_i^-$ searches its look-up table for a value x_t equal to the received value of x. If such a value is found, $\mathcal{R}_i^-$ identifies $\mathcal{T}$ and checks that $f_k(r_R||x_{t-1})$ equals the received value of M_T, where k is the key belonging to the identified tag $\mathcal{T}$. If this verification succeeds, then $\mathcal{R}_i^-$ authenticates $\mathcal{T}$. Otherwise, the session is terminated.

According to the SMD protocol, the delegated reader can authenticate $\mathcal{T}$ a maximum of m times, but the delegated reader cannot update the tag secrets, as it does not know s. If $\mathcal{T}$ is queried m times, then the server will need to update $\mathcal{T}$'s secret and identifiers and, if necessary, the server can now delegate the right to query the tag again. Moreover, multiple delegations of a tag $\mathcal{T}$ are also possible. If the server transfers the secret k and the identifiers $\{x_0, x_1, \cdots, x_m\}$ for $\mathcal{T}$ to multiple readers, then these entities can all authenticate $\mathcal{T}$ during the same limited period, that is, until $x = x_m$ is reached [47].

8.4. The Attacks on the Song-Mitchell Delegation Protocol

In this section, we first give the assumptions and the adversary model used in our attacking strategies. Then, we detail the attacks to the SMD protocol.

8.4.1. Assumptions and the Description of the Attacks

Before we elaborate on our attacking strategies, we will first state a set of reasonable assumptions about our attack model to address the security flaws of SMD. The delegated readers can be mobile embedded devices like notebooks, PDAs, mobile phones etc. that have an intermittent access to the back-end server and perform their tasks even in off-line mode. Hence, it is reasonable to assume that an adversary can compromise a delegated reader and obtain all the data stored in this reader e.g. by

stealing it or with the help of a dishonest reader. Indeed, due to these potential risks the protocols usually use temporarily delegation model such that the delegated reader can identify the tags with a limited number of times and cannot update secrets of the tags. In 2009, Avoine et al. introduced the problem of compromised readers [76] and pointed out the possibility and risk of such a situation. Throughout our attacks we assume that the adversary can initiate communication with the delegated readers and the tags. In addition to this, he can compromise a delegated reader and obtain all the data stored in it. Besides, we assume that SMD is used in an environment which has the properties of the basic RFID delegation architecture, i.e. the system includes a back-end server, online readers, delegated (off-line) readers and RFID tags.

Definition 8.1. *$S_{\mathcal{T}_i}(\pi)$ denotes the set of delegated readers that have right to identify tag $\mathcal{T}_i$ at instance π.*

Definition 8.2. *$S_{\mathcal{R}_i^-}(\pi)$ denote the set of tags that the delegated reader $\mathcal{R}_i^-$ has right to identify at instance π.*

Definition 8.3. *An RFID delegation protocol is called Multiple Delegated denoted with MD if $\exists(S_{\mathcal{R}_i^-(\pi)} \cap S_{\mathcal{R}_j^-(\pi)} \neq \varnothing)$, where $\mathcal{R}_i^-$ and $\mathcal{R}_j^-$ are two delegated readers. It is equivalent to say that for an MD protocol, one tag can be delegated to more than one reader.*

Note that SMD protocol provides multiple delegation by transferring the secret k and the identifiers $\{x_0, x_1, \cdots, x_m\}$ for $\mathcal{T}$ to multiple readers, then these entities can all authenticate $\mathcal{T}$ during the same limited period. Hence SMD scheme is an MD protocol.

As an initial condition of the attacks, we suppose that an adversary $\mathcal{A}$ compromised a delegated reader $\mathcal{R}_i^-$, so $\{k, x_0, x_1, \cdots, x_m\}$ tuple of each tag $\mathcal{T}_j \in S_{\mathcal{R}_i^-}(\pi)$ are available to him.

8.4.2. Tag Impersonation Attack

For SMD protocol, once the adversary compromised a reader $\mathcal{R}_i^-$, he gets all the secrets of $S_{\mathcal{R}_i^-}(\pi)$. Obviously, he can impersonate any tag in the set. This attack can be launched as follows:

i) $\mathcal{A}$ selects a tag $\mathcal{T}_0$ and a legitimate delegated reader $\mathcal{R}_j^-$ such that $\mathcal{T}_0 \in \{S_{\mathcal{R}_i^-}(\pi) \cap S_{\mathcal{R}_j^-}(\pi)\}$.
ii) $\mathcal{A}$ initiates a communication with $\mathcal{R}_j^-$.
iii) $\mathcal{R}_j^-$ transmits some random nonce r_R to $\mathcal{A}$.
iv) $\mathcal{A}$ generates $M_T = f_k(r_R||x_0)$ and transmits r_R, x_1 and M_T to $\mathcal{R}_j^-$.
v) $\mathcal{R}_j^-$ searches its look-up table for a value x_t equal to the received value of x_1. It finds the match and identifies as $\mathcal{T}_0$. Then $\mathcal{R}_j^-$ authenticates $\mathcal{A}$ as $\mathcal{T}_0$, since $f_k(r_R||x_{t-1}) = M_T$ is verified.

By using this attack, the adversary fools another delegated reader to accept a fake tag as valid. Indeed, if we replace $\mathcal{R}_j^-$ with an online reader $\mathcal{R}_j^+$, the attack still works but now $\mathcal{DB}$ authenticates fake tags. In other words, instead of interacting with other delegated readers, $\mathcal{A}$ can communicate with online readers/$\mathcal{DB}$ and impersonate authorized tags.

There is also another unnoticed security breach in SMD: The expire of delegation is only controlled by the tag via control of the counter c and this is not checked by the reader. Hence, an adversary can impersonate a tag to an authorized reader several times until the server recognizes the compromised reader and refreshes the tag secrets.

8.4.3. De-synchronization Attack

Most RFID authentication protocols include an update operation for the tag secrets and its ID at the end of a successful protocol run. Since this operation is

performed at the back-end server as well as at the tag, synchronization of secret information between the database and the tag is crucial for future authentications. In a de-synchronization attack, which is a particular form of denial-of-service attacks, an adversary intentionally makes a tag out of synchronization with the server to prevent subsequent authentications of the tag. In this part, it is shown that the SMD protocol is vulnerable to a de-synchronization attack, in which a tag loses synchronization with the back-end server by updating its secrets based on false information generated by the adversary. The attack is consisted of two phases: In the first phase of the attack the adversary impersonates the target tag to an online reader, whereas in the second phase the adversary pretends to be the back-end server. Steps of the attack are given below:

In Phase 1, the adversary impersonates tag $\mathcal{T}_0$ to an online reader

i) $\mathcal{A}$ selects a tag $\mathcal{T}_0$ such that $\mathcal{T}_0 \in S_{\mathcal{R}_i^-}(\pi)$.

ii) $\mathcal{A}$ initiates a communication with an online reader $\mathcal{R}_j^+$.

iii) $\mathcal{R}_j^+$ transmits some random nonce r_R to $\mathcal{A}$.

iv) $\mathcal{A}$ generates a random nonce r_T and computes $M_1 = f_k(r_R||r_T)$ and $M_2 = r_T \oplus x_m$. Then he transmits r_R, M_1, M_2 and $SecReq$ to show a request for an update of the shared secrets.

v) $\mathcal{DB}$ searches its look-up table for a value $x = x_m$ or $x = x_0$ for which $M_1 = f_k(r_R||(M_2 \oplus x)$. It finds the match for $x = x_m$ and authenticates $\mathcal{A}$ as $\mathcal{T}_0$.

vi) $\mathcal{DB}$ chooses a random l-bit string s' and an integer m', and computes a key $k' = h(s')$ and a sequence of m' identifiers $x'_t = e_{k'}(x'_{t-1})$ for $1 \leq t \leq m'$, where x'_0 is set to x.

vii) $\mathcal{DB}$ computes $r_T = M_2 \oplus x$ and $M_S = g_k(r_R||r_T) \oplus (s||k'||m')$ and sends r_R and M_S to $\mathcal{A}$.

viii) $\mathcal{DB}$ assigns the set of values $\{s, k, s', k', x, x'_1, \cdots, x'_{m'}\}$ to the tag $\mathcal{T}_0$, updating the previously stored values $\{\hat{s}, \hat{k}, s, k, x_0, x_1, \cdots, x_m\}$.

ix) $\mathcal{A}$ receives r_R and M_S. It calculates $(s||k'||m') = M_S \oplus g_k(r_R||r_T)$ and obtains s of $\mathcal{T}_0$.

In Phase 2, the adversary impersonates the server to the tag and desynchronizes it:

i) $\mathcal{A}$ initiates communication with $\mathcal{T}_0$ and transmits some random nonce r_A.
ii) $\mathcal{A}$ repeats the previous step till $\mathcal{T}_0$ responds $\bar{M}_1$, $\bar{M}_2$ with $SecReq$.
iii) $\mathcal{A}$ obtains $\bar{r}_T$ from $M_2 \oplus x_m$. Since s is available to him from first phase of the attack, he computes $\bar{M}_S = g_k(r_A||\bar{r}_T) \oplus (s||\bar{k}||\bar{m})$ where $\bar{k}$ and $\bar{m}$ are randomly produced values.
iv) $\mathcal{T}_0$ computes $\bar{M}_S \oplus g_k(r_A||\bar{r}_T)$ and obtains $(s||\bar{k}||\bar{m})$. Next, $h(s) = k$ is verified and the modified secrets $\bar{k}$ and $\bar{m}$ are accepted by the tag.

At the end, $\mathcal{DB}$ and $\mathcal{T}_0$ have different secret keys. To be specific, the server stores $(s, k, s', k', x, x'_1, \cdots, x'_{m'})$, whereas the tag stores $\bar{k}$ and $c = \bar{m}$. Therefore, the readers and the server would be in a desynchronized state with the tag where future authentication of the tag is hindered.

8.4.4. Tracking Attack

According to SMD protocol, tag does not authenticate delegated reader, so it cannot know whether its interacting entity is a compromised reader or a legitimate one. As a result, an adversary can easily trace the tags whose secrets are available to him due to a compromised reader and unless the secrets of tags are refreshed, the adversary threatens the privacy of the system.

8.4.5. Future Security Weakness of SMD

Let $\mathcal{T}_0$ and $\mathcal{R}_i^-$ be a tag and a delegated reader respectively in SMD. Suppose that $\mathcal{R}_i^- \in S_{\mathcal{T}_0}(\pi)$ and $|S_{\mathcal{T}_0}(\pi)| > 1$. Also assume that at instance $\pi + 1$, $\mathcal{DB}$ updates

the secret k and the identifiers of $\mathcal{T}_0$ and decides to end the $\mathcal{R}_i^-$'s delegation status for this tag. According to SMD delegation policy, $\mathcal{R}_i^-$ has no more delegation right for $\mathcal{T}_0$, but we show that if the entity $\mathcal{R}_i^-$ wants, he can impersonate, desynchronize and trace $\mathcal{T}_0$. In other words, if all data of $\mathcal{R}_i^-$ at instance π are available to an adversary, he can succeed in all of the previously described attacks, although secrets of the tags are renewed by the server at instance $\pi + 1$.

Let tag secrets of $\mathcal{T}_0$ at instance π and $\pi + 1$ be $\{s^\pi, k^\pi, x_0^\pi, x_1^\pi, \cdots, x_m^\pi\}$ and $\{s^{\pi+1}, k^{\pi+1}, x_0^{\pi+1}, x_1^{\pi+1}, \cdots, x_m^{\pi+1}\}$ respectively. On the other hand, the database $\mathcal{DB}$ stores $\{s^\pi, k^\pi, s^{\pi+1}, k^{\pi+1}, x_0^{\pi+1}, x_1^{\pi+1}, \cdots, x_m^{\pi+1}\}$ at instance $\pi + 1$, where s^π and k^π are kept to prevent a de-synchronization case. Below, we give steps of the attack after the tag's secrets are refreshed at instance $\pi + 1$.

i) $\mathcal{A}$ selects a tag $\mathcal{T}_0$ such that $\mathcal{T}_0 \in S_{\mathcal{R}_i^-}(\pi)$.

ii) $\mathcal{A}$ initiates a communication with an online reader $\mathcal{R}_j^+$.

iii) $\mathcal{R}_j^+$ transmits some random nonce r_R to $\mathcal{A}$.

iv) $\mathcal{A}$ generates a random nonce r_T and computes $M_1 = f_{k^\pi}(r_R||r_T)$ and $M_2 = r_T \oplus x_m^\pi$. Then he transmits r_R, M_1, M_2 and $SecReq$ to indicate a request for an update of the shared secrets.

v) $\mathcal{DB}$ searches its look-up table for a value $x = x_m^{\pi+1}$ or $x = x_0^\pi$ for which $M_1 = f_{k^\pi}(r_R||(M_2 \oplus x)$.

vi) $\mathcal{DB}$ finds the match for $x = x_0^\pi$ and believes that M_S did not reach $\mathcal{T}_0$ correctly in the previous session, and thus $\mathcal{T}_0$ did not update its secrets, although it did.

vii) $\mathcal{DB}$ computes $r_T = M_2 \oplus x$ and $M_S = g_{k^\pi}(r_R||r_T) \oplus (s^\pi||k^{\pi+1}||m^{\pi+1})$ and sends r_R and M_S to A.

viii) $\mathcal{A}$ receives r_R and M_S. It calculates $(s^\pi||k^{\pi+1}||m^{\pi+1}) = M_S \oplus g_{k^\pi}(r_R||r_T)$ and obtains $k^{\pi+1}$ and $m^{\pi+1}$ of $\mathcal{T}_0$.

ix) After the above steps $\mathcal{A}$ obtains secrets of $\mathcal{T}_0$ for the instance $\pi + 1$. Notice that once the adversary gets the secrets of $\pi + 1$, it is a trivial matter for him to carry out the tag impersonation, the de-synchronization and the tag tracking attacks.

It is clearly seen that $\mathcal{DB}$ cannot accomplish to revoke the delegation of an entity, although $\mathcal{DB}$ thinks the delegation rights of the corresponding entity is ended after secret update process for the tag.

8.5. Analyzing the Security Flaws of SMD

In the previous part, we presented different attacks against the SMD protocol and show that under our RFID delegation model these attacks undermine the security claims of the scheme. In this section, the design flaws are treated in more detail. Also, we claim that this is an inherent problem and can thus be easily generalized for other delegation protocols. To show this, we investigate security of two other delegation protocols by taking the following issues into consideration.

Remark 8.4. *The SMD protocol is vulnerable to the described attacks because of the following reasons:*

i) *The SMD protocol does not provide a mechanism such that a reader can verify whether its interacting entity is a legitimate tag or another delegated reader. The reason is apparent: there is no secret information, used in authentication, that the tag knows but the delegated reader does not.*

ii) *The delegation of the reader is limited depending on the counter value stored in the tag, i.e. only the tag controls expire of the reader's access, not the reader. As a consequence, an adversary can impersonate a tag to a delegated reader or the back-end server several times until the server recognizes the compromised reader and refreshes the tag secrets.*

iii) *Suppose a dishonest delegated entity $\mathcal{R}_i^-$ gives its secret information to an adversary $\mathcal{A}$. Next, another delegated reader $\mathcal{R}_j$ is fooled by $\mathcal{A}$ such that some tag $\mathcal{T}_0$ is impersonated. Afterwards, even if the attack is realized, the SMD protocol cannot clarify whether entity $\mathcal{R}_j^-$ is malicious or not, because according to the protocol description $\mathcal{R}_i^-$ and $\mathcal{R}_j^-$ have same rights and same secrets. By considering this scenario, one can say that the protocol cannot thwart the risk of untrustworthy*

delegated entities.

In the following part, we investigate security of two RFID delegation schemes proposed in the literature: That of Fouladgar and Afifi [75] and the model proposed by Lim and Kwon (LK) [77]. Because the main purpose of this part is addressing the common security problems in the RFID delegation protocols, we shortly give the protocol descriptions and omit whole details of the protocol processes (interested readers may refer to [75] and [77, 78] for the detailed description).

8.5.1. The Fouladgar's Protocol and Analysis

An RFID delegation protocol was proposed in [75] by Fouladgar and Afifi. In this scheme, the server stores the following values for each tag $\mathcal{T}_i$: The tag identifier $ID_{\mathcal{T}_i}$, the tag's pseudonym key K_{P_i} and the tag's update key K_{U_i}. To thwart desynchronization problems, old values of these keys are also kept by the server. On the other side, each tag stores the pseudonym key K_{P_i} and the update key K_{U_i}. In addition, the tag embeds a counter C which is initially set to zero and incremented at each query.

To delegate a tag $\mathcal{T}_i$ to a reader $\mathcal{R}_x^-$, the server only gives the pseudonym key K_{P_i} and $ID_{\mathcal{T}_i}$ to the reader. After this, the reader is allowed to identify the tag $\mathcal{T}_i$. We can briefly describe the steps of the delegation protocol and the secret update procedure as given below:

i) $\mathcal{R}_x^-$ generates a random nonce N_R and transmits to $\mathcal{T}_i$.

ii) $\mathcal{T}_i$ first increments the counter C and then generates a fresh nonce N_T. If $C < C_{max}$, then the tag computes $P = f_{K_{P_i}}(N_T \oplus N_R)$, where C_{max} is counter's maximum value and f is a symmetric key cryptographic function. However, if $C \geq C_{max}$ $\mathcal{T}_i$ uses the update key K_{U_i} in computation of $P = f_{K_{U_i}}(N_T \oplus N_R)$. Finally, the tag sends P to $\mathcal{R}_x^-$. Note that the counter mechanism permits to limit readers delegation to a number of C_{max} queries for a given tag.

iii) $\mathcal{R}_x^-$ computes $f_{K_P}(N_T \oplus N_R)$ for all the keys it possesses until it finds the right key and the corresponding $ID_{\mathcal{T}}$. If a match is obtained, $\mathcal{R}_x^-$ identifies the tag and the session is ended. However, if $\mathcal{T}_i$ is not in the set of $\mathcal{R}_x^-$'s delegated tags or if the P is computed with K_{U_i} by the tag, then obviously $\mathcal{R}_x^-$ will not find a match. In this case, the reader forwards the tags pseudonym P along with its credentials $Cred_R$ to the server.

iv) The server checks $Cred_R$ of the reader. If it is not valid, the protocol is ended. If the reader has the rights for tag identification, then the server initiates the key update mechanism: It produces a random value α and updates K_{U_i} and K_{P_i} as $K_{U_i} = K_{U_i} \oplus N_T \oplus \alpha$, $K_{P_i} = K_{P_i} \oplus N_T \oplus \alpha$ while keeping old values of K_{P_i} and K_{U_i}.

v) The server replies to $\mathcal{R}_x^-$ with $f_{K_{U_i}}(N_T||\alpha)$, using the old value of K_{U_i}.

vi) $\mathcal{R}_x^-$ forwards the received message to the tag.

vii) $\mathcal{T}_i$ decrypts the received message with K_{U_i} and gets α. Next it updates K_{U_i} and K_{P_i} in the same way than the server. After the tag has updated its keys, previously delegated readers have to refer to the database in order to extend their delegation rights.

In [75], Fouladgar and Afifi mentions that a delegated reader can be compromised. Hence, it is reasonable to analyze security of their protocol under the assumption of a compromised reader scenario. As it can be seen from the Fouladgar's protocol description if we revisit Remark 1, we can see that all items are also valid for the scheme: First, a delegated reader cannot realize whether its interacting entity is a delegated tag in its set or another delegated reader, because all delegated readers for the same tag possess the same tag pseudonym key K_{P_i}. Second, a tag access limit is not controlled by a delegated reader, instead it is only checked by the tag via counter value C. Last, no information, e.g. a reader identifier, to differentiate readers is used in tag identification procedure and this matches with third item of Remark 1. By considering these facts, we point out that the Fouladgar's protocol is vulnerable to a tag impersonation attack (similar to tag impersonation for SMD, but with customizing the messages according to protocol flow). Hence an adversary having the secrets of a

delegated reader can easily convinces another delegated reader to accept fake tags as valid for several times until the security violation is recognized by the server.

8.5.2. The LK Protocol and Analysis

In [77], Lim and Kwon proposed an RFID protocol which can also be used to delegate access to tags to particular readers, i.e. as an RFID delegation scheme. The initialization phase of the protocol is as follows:

- $\mathcal{DB}$ chooses a random secret K_i for each tag $\mathcal{T}_i$ and $m-1$ evolutions of K_i, $K_i^0 = K_i$ and $K_i^j = g(K_i^{j-1})$ for $1 \leq j \leq m-1$, where where g is a pseudorandom function. Then it computes $t_i^j = ext(K_i^j, l_2)$ for $1 \leq j \leq m-1$, where l_2 is some appropriate bit length, $ext(x, l)$ denotes a simple extract function returning l bits out of x. Notice that within this key chain, desynchronization up to m times can be resolved.
- $\mathcal{DB}$ also chooses a random u_i for each tag $\mathcal{T}_i$ and computes a key chain $\{w_i^j\}_{j=0}^{n-1}$ of length n, such that $w_i^n = u_i$ and $w_i^j = h(w_i^{j+1})$ for $0 \leq j < n$, where h is a pseudorandom function. This key chain is used in reverse order to authenticate the server and to trigger a refresh of a tag secret.
- Each tag $\mathcal{T}_i$ stores $\langle K_i, w_{i,T} \rangle$, where $w_{i,T} = w_i^0$ and sets a failure counter $c_i = 0$.
- $\mathcal{DB}$ makes two entries D_{old} (initially empty) and D_{new} for each tag $\mathcal{T}_i$ in its database. D_{new} has entries of the form $\langle K_i, \{t_i^j\}_{j=0}^{m-1}, u_i, n_i, w_{i,T}, w_{i,S} \rangle$, where $n_i = n$ and $w_{i,S} = w_i^1$ thus $w_{i,T} = h(w_{i,S})$.

A normal protocol run of LK protocol is depicted in Figure 8.3, where f is a pseudorandom function. In each session, the tag secret K_i is evolved using a one-way key chain in two different ways: If the authentication succeeds, the tag secret is refreshed probabilistically as $K_i = g(K_i \oplus w_{i,T} || r_1 || r_2)$. On the other hand if the protocol fails, the secret is updated deterministically as $K_i = g(K_i)$. Besides, the reverse key chain $\{w_i^j\}_{j=0}^{n-1}$ is used to validate the server at the tag side.

In [77], it is proposed that the LK protocol can also be used to provide tag delegation. For delegation of a tag $\mathcal{T}_i$ to a reader $\mathcal{R}_x^-$, the server delivers only the forward key chain for the tag secret and the tag ID to the reader as $\langle K_i, \{t_i^j\}_{j=0}^{m-1}, ID_{\mathcal{T}_i}\rangle$, but not the backward key chain, i.e. only the server can refresh tag secrets. After this, the reader may identify the delegated tags from the first two protocol flows. Note that $\mathcal{R}_x^-$ cannot produce a valid answer in the third flow and it responds with an arbitrary dummy message as σ_2. Nevertheless, the protocol works still correctly even if the σ_2 is not valid: The tag secret is changed deterministically and eventually remains static after m readings. Thus, this enables a delegated reader $\mathcal{R}_x^-$ to authenticate $\mathcal{T}_i$ until the server refreshes the secrets of the tag. The authentication steps of the LK delegation protocol is described as given below:

i) $\mathcal{R}_x^-$ transmits a random nonce r_1 to $\mathcal{T}_i$.

ii) $\mathcal{T}_i$ chooses another random nonce r_2 and computes $t_i = ext(K_i, l_2)$ and $\sigma_1 = ext(f(K_i, r_1||r_2), l_1)$.

iii) $\mathcal{T}_i$ sends $\{t_i, r_2, \sigma_1\}$ to $\mathcal{R}_x^-$.

iv) $\mathcal{R}_x^-$ searches its look-up table to find an entry containing the received key identifier t_i. If such a value is found as $t_i = t_i^j$ for some j, $\mathcal{R}_x^-$ identifies $\mathcal{T}_i$ and computes $K_i' = g(K_i)^j$ by using corresponding K_i in the list. Then it checks that $ext(f(K_i', r_1||r_2), l_1)$ equals the received value of σ_1. If this verification succeeds, then $\mathcal{R}_x^-$ authenticates $\mathcal{T}_i$. Afterwards, it sends a dummy message σ_2 to the tag.

v) $\mathcal{T}_i$ realizes that σ_2 is not valid and sets $c_i = c_i + 1$. If $c_i < m$, then the tag updates K_i as $K_i = g(K_i)$.

In the original study, existence of potentially untrustworthy readers is taken into account in designing the scheme. Hence, it is reasonable to analyze security of LK delegation protocol under the assumption of a compromised or untrustworthy reader scenario. As can be inferred from the above description, all of the security flaws given in Remark 1 are also valid for the LK delegation protocol. First of all, a reader cannot distinguish whether its interacting entity is a legitimate tag or another delegated reader, since the secret K_i is available to the adversary and he can easily generate the forward key chain by using it. This security flaw apparently makes the LK delegation protocol

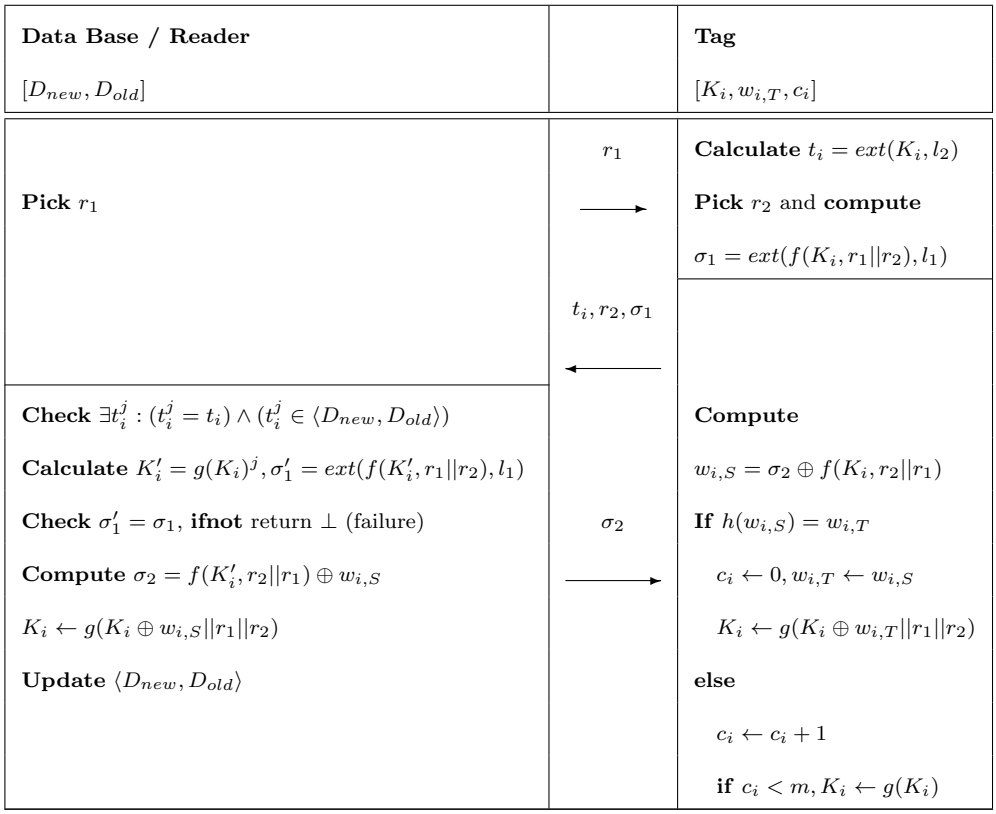

Figure 8.3. The LK RFID authentication protocol.

vulnerable to a tag impersonation attack by which an adversary having the secrets of a delegated reader can easily fools another delegated reader to accept a fake tag as valid. In addition to this, the LK protocol does not provide a limited delegation mechanism: A delegated reader can authenticate a tag as long as the secret of the tag is not refreshed by the server. Therefore an adversary can impersonate a tag to an authorized reader several times until the server becomes aware of a security problem and refreshes the tag secrets. Last, the LK protocol does not assign unique IDs or secrets to readers and this makes it relevant to the third item of the Remark 1.

8.5.3. Countermeasures against Security Vulnerabilities of RFID Delegation Schemes

In the light of our previous analysis, it would be necessary to take some precautions against potential compromise of delegated readers and the relevant flaws given

in Remark 1. Thus, we define below some security requirements that a basic RFID delegation protocol should take into account:

- Mainly the weakness in 1st item of Remark 1 comes from the fact that all secret information necessary to authenticate tags are transferred to delegated readers and this information is enough to impersonate/simulate tags. Therefore, for a delegation protocol it is of paramount importance to provide a mechanism such that a reader can verify whether its interacting entity is a legitimate tag or another delegated reader. This should be done by introducing information asymmetry in the tags and the delegated readers. For example, embedding a unique secret, shared by the server but not with the reader, can be used as a tool to differentiate between tags and other delegated readers. Such a solution will be effective in preventing tag impersonation by a compromised delegated reader.

- For a temporary delegation scheme, limited access delegation of a reader is usually controlled only by the tag e.g. via a counter. However as stressed in Section 8.4.1, in case of the tag impersonation attack a legitimate reader can be fooled by an adversary for several times. Hence, the expire time of the delegation should be controlled by both of the tag and the delegated reader. Although it is an auxiliary issue, a double check mechanism makes a protocol more complete and secure.

- Delivering same secrets, information and data to different delegated readers may result in some security problems: For example if one of the delegated readers is untrustworthy, then it is not an easy job for the authority to find which reader. Thus, the server must diversify delegated readers: Each of the delegated reader must have a unique ID assigned by the server. Also, different secrets or identifiers/pseudonoyms should be assigned for each reader. Moreover, reader ID should be taken as an input in computing the tag response message, i.e. tag's response varies depending on delegated reader ID.

8.6. Improving Security of the SMD Protocol

8.6.1. The Improved Song Mitchell Delegation Protocol

In this part, we enhance the security of the SMD protocol and make the resulting protocol immune against the security risks with proposing an improved model which we call ISMD.

Let $d()$ be a keyed hash function and z be a secret of l-bit string shared only by $\mathcal{DB}$ and the tag $\mathcal{T}$ such that z is different for each tag. Considering the last item of Remark 1, it is critically important to assign a unique ID to each of the readers. So, in our model each reader has a unique ID denoted as ID_R. Then the revisedimproved protocol is described as follows:

Initially, a secret l-bit string, s_j, and its hash value (computed by the server), $k_j = h(s_j)$, is assigned to the tag entries $\mathcal{T}_j \in \mathcal{DB}$. Moreover, for every tag $\mathcal{T}_j$, $\mathcal{DB}$ stores a hash-chain $\{x_0, x_1, \cdots, x_m\}$ where m is a positive integer, $x_t = e_k(x_{t-1}||z)$ for $1 \leq t \leq m$ and x_0 is a random l-bit string.

For delegation of a tag $\mathcal{T}$ to a reader $\mathcal{R}_i^-$, the server firstly transfers the secret k, the identifiers $\{x_0, x_1, \cdots, x_m\}$ and a stamp δ for $\mathcal{T}$ to the $\mathcal{R}_i^-$ via a secure channel, where δ is computed by $\mathcal{DB}$ as:

$$\delta = d_z(ID_{R_i^-}||k).$$

The authentication process between a delegated reader and a tag is described below[5] and it is also summarized in Table 8.1 :

[5]The same authentication steps are also done between an online reader and a tag as long as there exits corresponding tag identifers in the server list and $c \neq 0$.

1) $\mathcal{R}_i^-$ transmits a random nonce r_R and $ID_{R_i^-}$ to $\mathcal{T}$.

2) If $c \neq 0$, $\mathcal{T}$ computes $\delta = d_z(ID_{R_i^-}||k)$, $M_T = f_k(r_R||x||\delta)$, $x \leftarrow e_k(x||z)$, $c \leftarrow c-1$ and transmits r_R, x and M_T to $\mathcal{R}_i^-$. Otherwise, $\mathcal{R}_i^-$ has no more right to authenticate the tag so secrets and identifiers of $\mathcal{T}$ need to be refreshed.

3) $\mathcal{R}_i^-$ searches its look-up table for a value x_t equal to the received value of x. If such a value is found, $\mathcal{R}_i^-$ identifies $\mathcal{T}$ and checks that $f_k(r_R||x_{t-1}||\delta)$ equals the received value of M_T, where k and δ belong to the identified tag $\mathcal{T}$. If this verification succeeds, then $\mathcal{R}_i^-$ authenticates $\mathcal{T}$. Next, it reduces the corresponding tags' identifiers $\{x_0, x_1, \cdots, x_m\}$ to $\{x_t, \cdots, x_m\}$.

Table 8.1. Authentication steps for the improved model.

$\mathcal{R}_i^- \rightarrow \mathcal{T}$	: r_R, $ID_{R_i^-}$
$\mathcal{T} \rightarrow \mathcal{R}_i^-$	: $M_T = f_k(r_R||x||\delta)$, x
$\mathcal{R}_i^-$	: Searches for a value $x_t = x$ in the list.
	: If $f_k(r_R||x_{t-1}||\delta) = M_T$, authenticates $\mathcal{T}$ and updates identifer list.

Notice that compared with the original delegation protocol SMD, our improved delegation model ISMD needs one more execution of the keyed hash function from the tag, while the computational complexity remains the same for the delegated reader.

On the other side, ISMD protocol uses the same secret update procedure of SMD except that $M_1 = f_k(r_R||r_T)$ is replaced with $M_1 = f_k(r_R||r_T||\delta)$ and at the last step instead of sending message M_S, messages M_{S1} and M_{S2} are transmitted after the following computation:

$$M_{S1}=g_z(k||r_R||r_T||M_1) \oplus (s||k'||m'),$$
$$M_{S2}=d_z(k||M_{S1}).$$

Consequently, on the tag side $\mathcal{T}$ firstly calculates $d_z(k||M_{S1})$ to confirm that M_{S1} is produced by the back-end server and the message integrity is verified. Next, it computes $(s||k'||m') = M_{S1} \oplus g_z(k||r_R||r_T||M_1)$. Following this step, if $h(s)$ is equal to k, $\mathcal{T}$ authenticates $\mathcal{DB}$ and ends the session successfully after updating k and c to k' and m' respectively. The secret update procedure of the proposed model is illustrated in Table 8.2.

Table 8.2. Secret update procedure for the improved model.

$\mathcal{R}_j^+ \rightarrow \mathcal{T}$	: r_R, $ID_{R_j^+}$
$\mathcal{T} \rightarrow \mathcal{R}_j^+$	: $M_1 = f_k(r_R||r_T||\delta)$
$\mathcal{R}_j^+ \rightarrow \mathcal{T}$	: $M_{S1} = g_z(k||r_R||r_T||M_1) \oplus (s||k'||m')$,$M_{S2} = d_z(k||M_{S1})$

Before giving security analysis of ISMD, we want to stress a critical distinction between our protocol and SMD. Considering the SMD initial phase, the only difference of our proposed protocol is involvement of secret term z in generation of identifiers x_t as $x_t = e_k(x_{t-1}||z)$. In our case a delegated reader cannot produce the identifiers, because computation of identifiers requires knowledge of z which is only known by $\mathcal{DB}$ and the corresponding tag. As a consequence, a delegated reader can authenticate a tag until all given identifiers are used. By means of this modification, a back-end server may assign different length of delegation rights to different readers for the same tag. For example, suppose that $\mathcal{DB}$ generated 1000 identifiers as $\{x_0, x_1, \cdots, x_{1000}\}$ for a tag $\mathcal{T}_0$. Then, it sends $\{x_0, x_1, \cdots, x_{500}\}$ to a delegated reader R_i^- and $\{x_0, x_1, \cdots, x_{1000}\}$ to another one R_j^-. In this case, R_i^- has right to authenticate the tag $\mathcal{T}_0$ for 500 times while this is 1000 for R_j^-. Thus, after the tag is authenticated 500 times, R_i^- can no longer be able to authenticate $\mathcal{T}_0$, whereas R_j^- can still do. Although it is an implementation issue and does not directly pertain to security of the protocol, we believe such a change is necessary regarding disparate applications and makes a delegation protocol more flexible.

8.6.2. Security Analysis of the Improved Model

In this section, we analyze the security of the proposed protocol with respect to the previously described attacks under the same assumptions and we illustrate that it

resolves the existing security problems of SMD.

Resistance to Tag Impersonation Attack: If one analyzes the ISMD, it can be seen that an adversary cannot impersonate a tag $\mathcal{T}$ to another delegated reader as $\mathcal{R}_j^-$ by using the previously described tag impersonation attack. The reason is that only the registered tag can compute $\delta = d_z(ID_{R_j^-}||k)$. It is equivalent to say that another delegated reader $\mathcal{R}_i^-$ cannot generate $d_z(ID_{R_j^-}||k)$, where $i \neq j$, since it does not know z of the corresponding tag. Besides, a delegated reader $\mathcal{R}_i^-$ can use the stored term δ only in verification process, not in an impersonation attack because this value is calculated depending on the interacted reader ID. Thus, these facts repair the security weaknesses 1 and 3 of Remark 1. Moreover, after successful authentication the delegated reader removes used identifiers from its look-up table which prevents the security flaw mentioned in Remark 1's second item.

Resistance to De-synchronization Attack: The proposed model prevents an adversary to realize the first phase of the described de-synchronization attack, because only an entity which has z can convince the back-end server to initiate a secret update process. Also due to unknown z information, the adversary cannot benefit from the second phase of the attack neither, i.e. cannot recover s or k' from M_{S1}. Therefore, the scheme provides resistance against the presented de-synchronization attack. Moreover, in our protocol another message M_{S2} is used to verify integrity of M_{S1}. If M_{S2} were not involved in the last message flow, then the following desynchronization attack would be applied: After a tag $\mathcal{T}$'s secret update request, an adversary may intercept and block the message from $\mathcal{DB}$ to the tag. At this point, the adversary sends a forged message $\{r_R, \bar{M}_S\}$ to the tag with $\bar{M}_S = M_S \oplus (0||r_1||r_2)$ where r_1, r_2 are non-zero strings. Since $\mathcal{T}$ computes $\bar{M}_S \oplus g_k(r_R||r_T)$ and obtains $(s||\bar{k}||\bar{m})$, where $\bar{k} = k' \oplus r_1$ and $\bar{m} = m' \oplus r_2$, $h(s) = k$ is verified and the modified secrets are accepted by the tag. Therefore, $\mathcal{DB}$ and $\mathcal{T}$ set their secrets to different values and future authentications of the tag is prevented. Indeed this attack is possible on the SMD protocol. However, such an attack based on alteration of M_{S1} for our proposed protocol fails, because the

integrity verifier message M_{S2} is also transmitted.

Resistance to Tracking Attack: In the off-line authentication the delegated reader authenticates the tag, but the tag does not. As a result for the ISMD protocol, an adversary can still trace the tags whose secrets are available to him due to a compromised reader, unless the secrets of tags are refreshed. However, after secret update procedure an adversary will no longer be able to trace tags.

Analysis of Future Security: As mentioned earlier, an adversary without knowledge of z cannot succeed in a secret update at the server side and also cannot capture neither $k^{\pi+1}$ nor $m^{\pi+1}$. Hence, following a secret update if $\mathcal{DB}$ decides to revoke delegation of a reader then it has no more delegation right.

9. CONCLUSIONS

To conclude this dissertation, we firstly summarize the main contributions of the study and then points out some future work.

9.1. Summary

In this thesis, we have investigated different aspects of RFID authentication protocols; in particular security, privacy, and scalability features.

After giving an overview of RFID systems and their application fields in Chapter 2, we have addressed security, privacy and performance requirements in Chapter 3.

In Chapter 4, we have reviewed some well known RFID protocols through following a classification based on computational complexity taken by the server for its overall computations in tag identification process. Also, we have discussed security weakness, privacy flaws or other shortcomings of each protocol and then pointed out the open problem: designing a secure fully privacy–preserving and scalable RFID authentication protocol.

In Chapter 5, we have noticed that solving this open problem has motivated protocol designers to construct new schemes whose common characteristics is allowing tags to be in different states such that the server authenticates the tag in constant/sublinear time in a more frequent normal state and needs a linear search in a rare abnormal states. Thanks to this feature, such a protocol can satisfy security, privacy and scalability criterions. We have called this type of protocols as Unbalanced Authentication Protocol and showed, however, that these schemes are traceable for an adversary who can access the side channel information of the computational complexity of the backend database. More precisely, we have applied our result to examine the privacy of some existing protocols and showed that they fail to fulfill untraceability property. We have believed that this highlighted privacy issue facilitates development of stronger

schemes and should be taken into account as a design criteria for RFID authentication protocols. Hence, unbalanced states shall not be used anymore in RFID systems to prevent such a weakness.

In Chapter 6, we have presented security or privacy flaws in some recent RFID protocols that have received no attacks yet. Also, we have proposed some revisions, if possible, to eliminate weakness in these target schemes.

In Chapter 7, we have stated that exhaustive search process is crucial in RFID authentication protocols. Although the protocol might satisfy the necessary security requirements of the RFID system specifications, careless deployments of database search mechanisms could jeopardize the security of the whole system. Therefore, it should not be left to user's choice and has to be described precisely in the system specifications. We have believed that our attempt would point out this salient missing link in RFID security protocols and address the potential pitfalls or side channels in realizations. In order to support our claim, through a careful analysis we have given the minimum index difference of selected two tags in database such that the attacker succeed. In addition, the success probability of the proposed attack model is derived in terms of the number of tags in the system, number of cryptographic operations carried out by the server, the computational power of the server and the sensitivity of the attacker in timing. Moreover, as a countermeasure for the timing attack we have proposed a dynamic search process which would fail in measuring the running time differences for different tag searches. We have claimed that choosing non-identical random query sequences for all search items gives the desired protection for the described attacks.

In Chapter 8, we have discussed security of RFID delegation systems and in particular analyzed security of a newly proposed RFID delegation protocol which we call SMD. We have introduced a tag impersonation and a de-synchronization attack against it. Additionally, we have found a subtle flaw in this protocol: A delegated entity can keep its delegation rights though they are revoked by the back-end server. Indeed, this makes the tag impersonation and the de-synchronization attacks still applicable even if the secrets of tag are updated. Furthermore, we have identified possible threats

faced by such delegation protocols and have also proposed some solutions. To overcome these weaknesses, we have described a revised version for SMD without violation of any other security properties. Also, we have conducted a security analysis on the revised protocol and claimed that it meets its security requirements even though a delegated reader is compromised.

9.2. Future Work

We suggest some directions for further research relating to this study.

- The wide spread deployment of RFID technology makes privacy a major concern in RFID systems. In this thesis, we analyzed security and privacy of some existing RFID protocols and presented different attacks under some adversary models. However, an adversary sometimes may know very few about the used RFID protocol. Hence a new adversary model should be constructed such that according to this model the RFID protocol is a black box for the adversary and he can only access some side-channel information. Thus, RFID security protocols should be analyzed further under this model in order to show what an adversary can do even with a very limited attacker capabilities.

- In this thesis, we presented some revisions for an RFID delegation protocol. However, also for this scheme, a tag do not store session transactions indicating which delegated reader authenticate it for a certain session, because the tag is storage limited. Thus, some solutions should be developed and the back-end server can obtain knowledge of which delegated reader authenticated a specific tag.

REFERENCES

1. Chien, H.Y. and C. H. Chen, "Mutual Authentication Protocol for RFID Conforming to EPC Class 1 Generation 2 Standards," *Computer Standars & Interfaces*, vol. 29, no. 2, pp. 254–259, February 2007.

2. Ohkubo, M., K. Suzuki, and S. Kinoshita, "Cryptographic Approach to 'Privacy-Friendly' Tags," *Proceedings of RFID Privacy Workshop*, November 2003.

3. Rhee, K., J. Kwak, S. Kim, and D. Won, "Challenge-Response based RFID Authentication Protocol for Distributed Database Environment," *Proceedings of International Conference on Security in Pervasive Computing, Lecture Notes in Computer Science*, vol. 3450, pp. 70–84, April 2005.

4. Nguyen Duc, D., J. Park, H. Lee, and K. Kim, "Enhancing Security of EPCglobal Gen-2 RFID Tag against Traceability and Cloning," *Proceedings of Symposium on Cryptography and Information Security*, January 2006.

5. Song, B. and C. J. Mitchell, "RFID Authentication Protocol for Low-Cost Tags," *Proceedings of the 1st ACM Conference on Wireless Network Security*, pp. 140–147, March–April 2008.

6. Dimitriou, T., "A Lightweight RFID Protocol to Protect Against Traceability and Cloning Attacks," *Proceedings of Conference on Security and Privacy for Emerging Areas in Communication Networks*, September 2005.

7. Henrici, D. and P. Müller, "Hash-based Enhancement of Location Privacy for Radio-Frequency Identification Devices Using Varying Identifiers," *Proceedings of International Workshop on Pervasive Computing and Communication Security*, pp. 149–153, March 2004.

8. Molnar, D. and D. Wagner, "Privacy and Security in Library RFID: Issues, Prac-

tices, and Architectures," *Proceedings of Conference on Computer and Communications Security*, pp. 210–219, October 2004.

9. Ha, J., S. Moon, J. M. G. Nieto, and C. Boyd, "Low-Cost and Strong-Security RFID Authentication Protocol," *Proceedings of the 2007 Conference on Emerging Direction in Embedded and Ubiquitous Computing, Lecture Notes in Computer Science*, vol. 4809, pp. 795–807, 2007.

10. Tsudik, G., "A Family of Dunces: Trivial RFID Identification and Authentication Protocols," Cryptology ePrint Archive, Report 2006/015, IACR, 2007.

11. Shaoying, C., Y. Li, T. Li, and R. Deng, "Attacks and Improvements to an RFID Mutual Authentication Protocol and Its Extensions," *Proceedings of the 2nd ACM Conference on Wireless Network Security*, pp. 51–58, March 2009.

12. Burmester, M., B. de Medeiros, and R. Motta, "Anonymous RFID Authentication Supporting Constant-Cost Key-Lookup against Active Adversaries," *International Journal of Applied Cryptography*, vol. 1, no. 2, pp. 79–90, 2008.

13. Song, B. and C. J. Mitchell, "Scalable RFID Authentication Protocol," *Proceedings of 3rd International Conference on Network ans System Security*, pp. 216–224, October 2009.

14. Ha, J., J. Ha, S. Moon, and C. Boyd, "LRMAP: Lightweight and Resynchronous Mutual Authentication Protocol for RFID System," *Proceedings of International Conference on Ubiquitous Convergence Technology, Lecture Notes in Computer Science*, vol. 4412, pp. 80–89, 2007.

15. Liu, Y., "An Efficient RFID Authentication Protocol for Low-Cost Tags," *Proceedings of IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, pp. 706–711, 2008.

16. Peris-Lopez, P., J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Rib-

agorda, "LMAP: A Real Lightweight Mutual Authentication Protocol for Low-Cost RFID Tags," *Proceedings of Workshop on RFID Security*, July 2006.

17. Chien, H.Y., "SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 4, pp. 337–340, December 2007.

18. Peris Lopez, P., J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "EMAP: An Efficient Mutual Authentication Protocol for Low-Cost RFID Tags," *Proceedings of OTM Federated Conferences and Workshop*, 2006.

19. Li, T. and G. Wang, "SLMAP – A Secure Ultra-Lightweight RFID Mutual Authentication Protocol," *Proceedings of Chinacrypt'07*, pp. 19–22, 2007.

20. Peris-Lopez, P., J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "M2AP: A Minimalist Mutual-Authentication Protocol for Low-cost RFID Tags," *Proceedings of International Conference on Ubiquitous Intelligence and Computing, Lecture Notes in Computer Science*, vol. 4159, pp. 912–923, September 2006.

21. Erguler, I., M. Akgun, and E. Anarim, "Cryptanalysis of a Lightweight RFID Authentication Protocol – LRMAP," *Proceedings of Western European Workshop on Research in Cryptology*, July 2009.

22. Erguler, I. and E. Anarim, "Scalability and Security Conflict for RFID Authentication Protocols," *Wireless Personal Communications*, vol. 59, no. 1, pp. 43–56, 2011.

23. Erguler, I., S. Aktas, E. Anarim, and G. Saldamli, "Breaking the Spacing Based Authentication Protocol," *Information Journal*, accepted.

24. Erguler, I. and E. Anarim, "Practical Attacks and Improvements to an Efficient RFID Authentication Protocol," *Concurrency and Computation: Practice and Ex-*

perience, accepted.

25. Erguler, I., C. Unsal, E. Anarim, and G. Saldamli, "Security Analysis of an Ultra-lightweight RFID Authentication Protocol–SLMAP*," *Security and Communication Networks*, 2011, doi:10.1002/sec.347.

26. Erguler, I. and E. Anarim, "Attacks on an Efficient RFID Authentication Protocol," *Proceedings of 10th IEEE International Conference on Computer and Information Technology, CIT 2010*, pp. 1065–1069, June 2010.

27. Erguler, I., E. Anarim, and G. Saldamli, "A Salient Missing Link in RFID Security Protocols," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 541283:10.1155/2011/541283, 2011.

28. Erguler, I. and E. Anarim, "Security Flaws in a Recent RFID Delegation Protocol," *Personal and Ubiquitous Computing*, 2011, doi:10.1007/s00779-011-0393-1.

29. Juels, A., "RFID Security and Privacy: A Research Survey," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 381–394, February 2006.

30. Vacca, J. R., *Computer and Information Security Handbook*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009.

31. Peris-Lopez, P., "Lightweight Cryptography in Radio Frequency Identification (RFID) Systems," Ph.D. dissertation, Computer Science Department, Carlos III University of Madrid, November 2008.

32. Song, B., "RFID Authentication Protocols using Symmetric Cryptography," Ph.D. dissertation, Royal Holloway, Univeristy of London, Egham, Surrey, United Kingdom, December 2009.

33. Schneider, M., "Radio Frequency Identification (RFID) Technology and its Applications in the Commercial Construction Industry," Technical Report, University of Kentucky Civil Engineering Department, 2003.

34. Langheinrich, M., "A Survey of RFID Privacy Approaches," *Personal and Ubiquitous Computing*, vol. 13, pp. 413–421, August 2009.

35. Chawla, V. and D. S. Ha, "An Overview of Passive RFID," *IEEE Applications and Practice*, vol. 45, no. 9, pp. 11–17, 2007.

36. Roussos, G. and V. Kostakos, "RFID In Pervasive Computing: State-Of-The-Art and Outlook," *Pervasive and Mobile Computing*, vol. 5, pp. 110–131, February 2009.

37. Want, R., "The Magic of RFID," *Queue*, vol. 2, pp. 40–48, October 2004.

38. Garfinkel, S., A. Juels, and R. Pappu, "RFID Privacy: An Overview of Problems and Proposed Solutions," *IEEE Security and Privacy*, vol. 3, no. 3, pp. 34–43, May–June 2005.

39. Alomair, B. and R. Poovendran, "Privacy versus Scalability in Radio Frequency Identification Systems," *Computer Communication, Elsevier*, vol. 33, no. 18, pp. 2155–2163, 2010.

40. Burmester, M. and B. de Medeiros, "RFID Security: Attacks, Countermeasures and Challenges," *Proceedings of 5th RFID Academic Convocation, The RFID Journal Conference*, July 2007.

41. Kang, J. and D. Nyang, "RFID Authentication Protocol with Strong Resistance against Traceability and Denial of Service Attacks," *Proceedings of European Workshop on Security and Privacy in Ad hoc and Sensor Networks, Lecture Notes in Computer Science*, vol. 3813, pp. 164–175, July 2005.

42. van Deursen, T. and S. Radomirović, "Security of RFID Protocols - A Case Study," *Electronic Notes in Theoretical Computer Science*, vol. 244, pp. 41–52, 2009.

43. Avoine, G., "Adversarial Model for Radio Frequency Identification," Cryptology ePrint Archive, Report 2005/049, IACR, 2005.

44. Vaudenay, S., "On Privacy Models for RFID," *Proceedings of Asiacrypt 2007, Lecture Notes in Computer Science*, vol. 4833, pp. 68–87, 2007.

45. Juels, A. and S. Weis, "Defining Strong Privacy for RFID," *Proceedings of International Conference on Pervasive Computing and Communications*, pp. 342–347, March 2007.

46. Ryu, E.-K. Ryu and T. Takagi, "A Hybrid Approach for Privacy-Preserving RFID Tags," *Computer Standards & Interfaces*, vol. 31, no. 4, pp. 812–815, 2009.

47. Song, B. and C. J. Mitchell, "Scalable RFID Security Protocols Supporting Tag Ownership Transfer," *Computer Communication, Elsevier*, vol. 34, no. 4, pp. 556–566, March 2011.

48. Avoine, G. and P. Oechslin, "RFID Traceability: A Multilayer Problem," *Proceedings of Financial Cryptography, Lecture Notes in Computer Science*, vol. 3570, pp. 125–140, 2005.

49. Karygiannis, A., T. Phillips, and A. Tsibertzopoulos, "RFID Security: A Taxonomy of Risk," *Proceedings of 1st International Conference on Communications and Networking*, pp. 1–7, October 2006.

50. van Deursen, T. and S. Radomirović, "Attacks on RFID Protocols," Cryptology ePrint Archive, Report 2008/310, IACR, 2008.

51. Mitrokotsa, A., M. R. Rieback, and A. S. Tanenbaum, "Classifying RFID Attacks and Defenses," *Information Systems Frontiers*, vol. 12, no. 5, pp. 491–505, 2010.

52. Weis, S., S. Sarma, R. Rivest, and D. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," *Proceedings of International Conference on Security in Pervasive Computing, Lecture Notes in Computer Science*, vol. 2802, pp. 454–469, 2003.

53. Tsudik, G., "A Family of Dunces: Trivial RFID Identification and Authentication

Protocols," *Proceedings of Workshop on Privacy Enhancing Technologies, Lecture Notes in Computer Science*, vol. 4776, pp. 45–61, June 2007.

54. Cheon, J. H., J. Hong, and G. Tsudik, "Reducing RFID Reader Load with the Meet-in-the-Middle Strategy," Cryptology ePrint Archive, Report 2009/092, IACR, 2009.

55. Li, T., R. H. Deng, and G. Wang, "The Security and Improvement of an Ultra-Lightweight RFID Authentication Protocol," *Security and Communication Networks*, vol. 1, no. 2, pp. 135–146, 2008.

56. Phan, R. C.-W., "Cryptanalysis of a New Ultralightweight RFID Authentication Protocol - SASI," *IEEE Transactions on Dependable and Secure Computing*, vol. 99, no. 1, 2008.

57. Li, T. and G. Wang, "Security Analysis of Two Ultra-Lightweight RFID Authentication Protocols," *Proceedings of IFIP TC-11 22nd International Information Security Conference*, vol. 232, pp. 109–120, May 2007.

58. Hernandez-Castro, J. C., J. E. Tapiador, P. Peris-Lopez, J. A. Clark, and E.-G. Talbi, "Metaheuristic Traceability Attack against SLMAP, an RFID Lightweight Authentication Protocol," *Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium*, May 2009.

59. Chang, J.-C. and H.-L. Wu, "A Hybrid RFID Protocol against Tracking Attacks," *Proceedings of Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 865–868, 2009.

60. Ouafi, K. and R. C. W. Phan, "Privacy of Recent RFID Authentication Protocols," *Proceedings of 4th International Conference on Information Security Practice and Experience, Lecture Notes in Computer Science*, vol. 4991, pp. 263–277, 2008.

61. Avoine, G., I. Coisel, and T. Martin, "Time Measurement Threatens Privacy-

Friendly RFID Authentication Protocols," *Proceedings of Workshop on RFID Security, Lecture Notes in Computer Science*, vol. 6370, pp. 138–157, 2010.

62. Burmester, M., T. v. Le, and B. de Medeiros, "Provably Secure Ubiquitous Systems: Universally Composable RFID Authentication Protocols," *Proceedings of Conference on Security and Privacy for Emerging Areas in Communication Networks*, pp. 1–10, August–September 2006.

63. Jin, Z.-A., Z.-Q. Cheng, and K.-Y. Yoo, "Spacing Based Authentication Protocol for Low-Cost RFID," *Proceedings of the 2008 Second International Conference on Future Generation Communication and Networking*, pp. 160–163, 2008.

64. An, Y. and S. Oh, "RFID System for Users Privacy Protection," *Proceedings of Asia-Pacific Conference on Communications*, pp. 516–519, 2005.

65. Osaka, K., T. Takagi, K. Yamazaki, and O. Takahashi, "An Efficient and Secure RFID Security Method with Ownership Transfer," *Proceedings of Computational Intelligence and Security, Lecture Notes in Computer Science*, vol. 4456, pp. 778–787, 2006.

66. Lee, S., T. Asano, and K. Kim, "RFID Mutual Authentication Scheme Based on Synchronized Secret Information," *Proceedings of Symposium on Cryptography and Information Security*, January 2006.

67. Peris-Lopez, P., J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "An Efficient Authentication Protocol for RFID Systems Resistant to Active Attacks," *Proceedings of International Workshop on Security in Ubiquitous Computing Systems, Lecture Notes in Computer Science*, vol. 4809, pp. 781–794, December 2007.

68. Fouladgar, S. and H. Afifi, "A Simple Privacy Protecting Scheme Enabling Delegation and Ownership Transfer for RFID Tags," *Journal of Communications*, vol. 2, no. 6, pp. 6–13, 2007.

69. Chien, H. Y. and C. W. Huang, "A Lightweight RFID Protocol Using Substring," *Proceedings of Embedded and Ubiquitous Computing, Lecture Notes in Computer Science*, vol. 4808, pp. 422–431, December 2007.

70. Lim, T., T. Li, and T. Gu, "Secure RFID Identification and Authentication with Triggered Hash Chain Variants," *Proceedings of 14th International Conference on Parallel and Distributed Systems*, pp. 583–590, January 2008.

71. Kocher, P., "Timing Attacks on Implementations of Diffie–Hellman, RSA, DSS and Other Systems," *Proceedings of Advances in Cryptology, Lecture Notes in Computer Science*, vol. 1109, pp. 104–113, 1996.

72. Brumley, D. and D. Boneh, "Remote Timing Attacks are Practical," *Proceedings of 12th Usenix Security Symposium*, pp. 1–14, 2004.

73. Avoine, G., E. Dysli, and P. Oechslin, "Reducing Time Complexity in RFID Systems," *Proceedings of Selected Areas in Cryptography, Lecture Notes in Computer Science*, vol. 3897, pp. 291–306, August 2005.

74. Zhang, Y. and P. Kitsos, *Security in RFID and Sensor Networks*, 1st ed. Boston, MA, USA: Auerbach Publications, 2009.

75. Fouladgar, S. and H. Afifi, "An Efficient Delegation and Transfer of Ownership Protocol for RFID Tags," *Proceedings of First International EURASIP Workshop on RFID Technology*, September 2007.

76. Avoine, G., C. Lauradoux, and T. Martin, "When Compromised Readers Meet RFID," *Proceedings of Workshop on RFID Security*, July 2009.

77. Lim, C. H. and T. Kwon, "Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer," *Proceedings of International Conference on Information and Communications Security, Lecture Notes in Computer Science*, vol. 4307, pp. 1–20, December 2006.

78. Ouafi, K and R. C. W. Phan, "Traceable Privacy of Recent Provably-Secure RFID Protocols," *Proceedings of the 6th International Conference on Applied Cryptography and Network Security, Lecture Notes in Computer Science*, vol. 5037, pp. 479–489, June 2008.