SECURITY AND PRIVACY IN RADIO FREQUENCY IDENTIFICATION

by

Mete Akgün

B.S, Computer Engineering, Bahçeşehir University, 2005

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

Graduate Program in Computer Engineering

Boğaziçi University

2009

ACKNOWLEDGEMENTS

This thesis is dedicated to my mother and my father, Sıddıka Akgün and Mustafa Akgün.

I would like to thank my thesis advisor Prof. Mehmet Ufuk Çağlayan and my co-advisor Prof. Emin Anarım, for their guidance, endless support and contributions throughout the preparation of my thesis. They always support and encourage me during my studies.

I would like to express my gratitude to Dr. Hüseyin Demirci, Dr. Mahmut Şamil Sağıroğlu, Dr. Nezih Geçkinli and Pınar Kavak for their patient reviews of my studies.

I also would like to thank Shaoying Cai from Singapore Management University and Li Lu from Hong Kong University of Science and Technology for their support.

Finally, I wish to thank to my family for their endless love, support, patience and encouragement. I especially want to thank my wife Yasemin Akgün for her love, patience and understanding to me.

ABSTRACT

SECURITY AND PRIVACY IN RADIO FREQUENCY IDENTIFICATION

This thesis studies security and privacy issues of Radio Frequency Identification (RFID) technology that enhances ubiquitous computing environment. RFID technology is used to identify many types of objects. Some of the main applications are asset management, tracking, access control and automated payment. Therefore, in the near future, this technology will replace the barcode technology.

However, privacy is one of main issues to adopt RFID technology in daily use. Due to resource constraints of low cost RFID tags in terms circuit size, power consumption and memory size, it is very restricted to design a private authentication protocol based on existing cryptographic functions. Therefore new private authentication protocols based on lightweight cryptography are required to use low cost RFID tags in RFID applications.

In this thesis, we focus on low cost RFID tags. Our contributions are as follows. Firstly, we propose a new strong privacy model called ACAP for RFID systems and analyze the privacy of a former authentication protocol based on our privacy model. Former proposal assumes that the adversary should miss any reader-to-tag communication flows and claims that their protocol is secure against forward traceability only in such communication environment. We show that even under such an assumption, the former proposed protocol is not secure. We propose a new RFID authentication protocol called ACA. We analyze its security based on our privacy model. The proposed protocol provides better protection against privacy and security threats than those be-

fore. It is resistant to server impersonation attack without any assumption and secure against forward traceability, if the adversary misses any reader-to-tag communication flows. Furthermore, we analyze the performance of ACA. It has low computational load on both the tag and the server side.

Secondly, we analyze the privacy and security of another former proposal SAPA (Storage Awareness Private Authentication) and discover that SAPA does not provide location and information privacy between successful authentication sessions and does not resist denial of service attacks, forward traceability, and server impersonation. We analyze the weaknesses of SAPA and propose a new tree based RFID authentication protocol called ACAT. ACAT provides better protection against privacy and security threats than those before. It provides tag information privacy and tag location privacy, and resists replay attacks, denial of service attacks, backward traceability, forward traceability (under an assumption), and server impersonation with an efficient key-lookup. Furthermore, ACAT has the least computation and communication load on both the tag and the server side compared to other tree based protocols.

ÖZET

RADYO FREKANSLI TANIMLAMADA GÜVENLİK VE GİZLİLİK

Bu tez her yerde birden bulunan hesaplama ortamını geliştiren, radyo frekanslı tanımlamanın (RFID) güvenlik ve gizlilik konuları üzerinde durmaktadır. RFID teknolojisi birçok tipte nesneyi tanımlamada kullanılmaktadır. Varlık yönetimi, izleme, erişim kontrol ve otomatik ödeme RFID teknolojisinin kullanıldığı bazı temel uygulamalardır. Bu nedenle yakın gelecekte RFID teknolojisinin barkod teknolojisinin yerini alacaktır.

Bununla beraber, RFID teknolojisini günlük kullanıma uygun hale getirmedeki en önemli mesele gizliliktir. Düşük maliyetli RFID etiketlerinin devre boyutu, güç tüketimi ve hafıza boyutu açısından kaynak sınırlamaları olduğu için, varolan kriptografik fonksiyonlara dayanarak gizli kimlik denetim protokolleri tasarlamak çok güçtür. Bu nedenle düşük maliyetli RFID etiketleri RFID uygulamalarında kullanmak için hafif kriptografiye dayanan yeni gizli kimlik denetim protokolleri gerekmektedir.

Biz bu tezde, düşük maliyetli etiketler üzerinde odaklandık. İlk olarak, RFID sistemleri için yeni güçlü bir güvenlik modeli (ACAP) öneriyoruz ve yeni önerilmiş bir kimlik denetim protokolünün güvenliğini önerdiğimiz modele göre analiz ediyoruz. Önerilen protokolde, düşmanın en az bir iletişim akışını kaçırması gerektiği varsayılıyor ve bu varsayım altında protokolün ileri izlenebilirliğe karşı güvenli olduğunu iddaa ediliyor. Biz önerilen protokolün bu varsayım altında güvenli olmadığını gösteriyoruz. RFID sistemleri için yeni bir kimlik denetim protokolü (ACA) öneriyoruz ve önerdiğimiz protokolün güvenliğini kendi güvenlik modelimizi kullanarak analiz ediyoruz. Önerdiğimiz protokol gizlilik ve güvenlik tehditlerine karşı önceki protokollerden daha iyi bir ko-

ruma sağlıyor. Önerdiğimiz protokol sunucu taklit etme atağına karşı her hangi bir varsayım olmadan koruma sağlıyor ve ileri izlenebilirliğe karşı düşmanın en az bir iletişim akışını kaçırması gerektiği varsayımı altında koruma sağlıyor. Bununla beraber, önerdiğimiz protokolün performansını analiz ettik ve daha önceki çalışmalarla karşılaştırdık. Önerdiğimiz protokol sunucu ve etiket tarafında düşük hesaplama yüküne sahiptir.

İkinci olarak, diğer bir kimlik denetim protokolünün SAPA (Storage Awareness Private Authentication) güvenliğini analiz ediyoruz ve SAPA'nın başarılı doğrulama oturumları arasında lokasyon ve bilgi gizliliği sağlamadığını ve SAPA'nın ileri izlenebilirliğe, servis reddi saldırısına ve sunucu taklit etme saldırısına dayanıklı olmadığını gösteriyoruz. RFID sistemleri için ağaç yapısına dayanan yeni bir kimlik denetim protokolü (ACAT) öneriyoruz. ACAT gizlilik ve güvenlik tehlikelerine karşı daha önceki protokollerden daha iyi koruma sağlıyor. ACAT bilgi gizliliği ve lokasyon gizliliği sağlıyor ve tekrar saldırılarına, servis reddi saldırılarına, ileri izlenebilirliğe (varsayım altında), geri izlenebilirliğe ve sunucu taklit etme saldırısına direnç gösterebiliyor. Diğer ağaç yapısına dayanan protokollerle kıyasladığımızda, etiket ve sunucu tarafında ACAT en az hesaplama ve iletişim yüküne sahiptir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$Adv_P^{UNT}(\mathcal{A})$	Advantage of adversary
$\mathcal{A}$	Adversary
$f()$	Pseudorandom number generator
$g()$	Hash function
$\mathcal{G}$	Attack game
$h()$	Hash function
I	The length of interaction
l_{ref}	The security parameter
l_{ch}	The security parameter
M	Message
N	The number of tags
$\mathcal{P}$	RFID protocol
r	Pseudorandom number generated by PRNG
$\mathcal{R}$	RFID reader
$seed_i$	The seed of value of the server validator val_i
$\mathcal{T}$	RFID tag
T_i	The i-th Tag
x_{new}	The refreshed value of x
x_{old}	The most recent value of x
val_i	The server validator of T_i
$\omega_i(\mathcal{T})$	The result of query
$\Omega_I(\mathcal{T})$	Results of the set of queries
$\gg$	Right circular shift operator
$\ll$	Left circular shift operator

$\oplus$	XOR operator
$\|\|$	Concatenation operator
$\in$	The random choice operator

ACA	Akgun-Caglayan-Anarim Protocol
ACAP	Akgun-Caglayan-Anarim Privacy Model
ACAT	Akgun-Caglayan-Anarim Tree Based Protocol
AES	Advanced Encryption Standard
CR	Challenge Response
CRC	Cyclic Redundancy Code
DB	Database
DoD	Department of Defense
DoS	Denial of Service
EAS	Electronic Article Surveillance
EPC	Electronic Product Code
HF	High Frequencies
ID	Identifier
LF	Low Frequencies
PRF	Pseudo Random Function
PRNG	Pseudo Random Number Generator
RA	Restricted Access
RF	Radio Frequency
RFID	Radio Frequency Identification
SAPA	Storage Awareness Private Authentication
UA	Universal Access
UHF	Ultra High Frequencies
XOR	Exclusive-or

1. INTRODUCTION

1.1. Radio Frequency Identification

Generally, radio transmissions are considered as Radio Frequency Identification (RFID) if they contain information related to identification. RFID usage is increasing rapidly. Wal-Mart's top 100 suppliers use more than one billion tags on cases and pallets of inventory per year. RFID technology is already used for different purposes such as warehouse management and logistics, railroad car tracking, product identification, library books check-in/check-out and, asset tracking [2]. Michelin embedded RFID tags in tires for its tire tracking system. For more sensitive area, European Central Bank (ECB) determined to embed RFID tags on its banknotes from 2005 for special purposes like banknote tracking and strong forgery resistance as well as user privacy protection. In near future several countries will start to use passports, credit cards and other payment devices that contain RFID chips [3].

RFID is a technology that sends or receives identification information by using radio signals. RFID systems fundamentally consist of three elements: RFID tags, RFID readers and a back-end server. The tag is the basic building block of RFID systems. Each tag consists of an antenna and a small silicon chip that includes a radio receiver and a radio modulator, a control logic, some amount of memory and a power system. RFID tags can take its power from incoming RF signal [4]. The reader is a device that retrieves information from tags [2]. It sends a radio signal to a tag and listens for the tag's response. The tag detects this signal and sends back a response that contains its identification information [4]. After retrieving the tag's response, the reader sends it to the back-end server. The server can retrieve detailed information of the tag by querying its database with the tag's response.

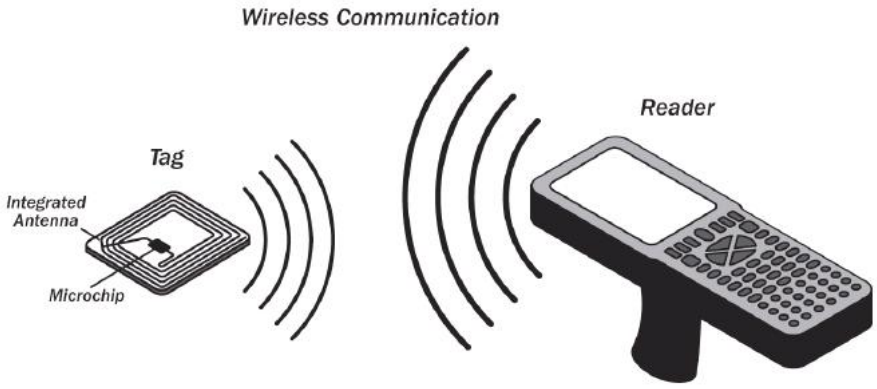


Figure 1.1. RFID architecture

1.2. RFID Applications

The first RFID application used by the British Royal Air Force during World War II is "Friend or Foe Transponder Identification System". In the late 1960s, the first commercial RFID application, the Electronic Article Surveillance (EAS) system was developed. EAS uses 1-bit tags to prevent shoplifting. In the 1980s and 1990s other RFID commercial applications, such as livestock tagging and toll road payment systems were developed [5].

Millions of people around the world are aware of RFID applications by the end of the 20^{th} century. However, they are not aware of inner workings of these applications. In the beginning of the 21^{th} century, the loss of privacy in RFID technology makes many people worried about if RFID usage becomes more widespread. We must admit that these people have right on their side because the business world wants to use RFID in wide variety of industries for commercial purposes. Business managers and theirs experts are learning about RFID to improve and optimize their companies' processes.

While technology is becoming cheaper, it is more available and easier to use. After the price of silicon chips used in manufacture of RFID tags has dropped, more

companies start to use RFID applications. The first major commercial use of RFID was started by Wal-Mart in June 2003. Wal-Mart made the use of RFID on the cases and pallets shipped from its top 100 suppliers mandatory. In October 2003, the US Department of Defense (DoD) wanted the use of RFID from its suppliers. Some of the biggest retailers in the world Albertsons, Metro, Target, Tesco have said that they plan to use EPC technology to track goods in their supply chain. The pharmaceutical, tire, defense and other industries are also moving to adopt the technology.

RFID uses provide better visibility in supply chain systems and RFID technology can reduce costs and increase efficiencies of companies. Despite the benefits of RFID applications, RFID applications in supply chain systems are still in their early stages. In the near future, contactless payment systems and smart cards will become the most widespread markets for RFID

The applied uses of RFID are listed[2]:

- Supply Chains : Wal-Mart, DoD, Target, Metro Group
- Item Level Tagging : Marks & Spencer testing of consumer apparel in limited number of stores
- Toll Payment Systems : OGS in Turkey
- Smart Cards : Philips MIFARE technology
- Contactless Payment Systems : Mastercard PayPass
- Logistics : Kimberly Clark
- Automobile Keyless Start Systems : Toyota, Lexus, Audi
- Sports : Tracking marathon runners
- Ticketing : RFID embedded tickets, used in 2008 Olympics
- Access Control : Controlling access to campuses, buildings and rooms
- Pet Microchipping : Inserting glass-encased RFID tags under the skin of pets
- Livestock and Wildlife Tagging : Tagging livestock and wildlife for conservation

and tracking purposes

- People Tagging : Tagging people for medical and security purposes
- Luggage Tracking : Hong Kong Airport, Delta Airlines
- Passport and Border Control : Japan, Norway, Pakistan
- Libraries : Berkeley library
- File Management : 3M's file tracking systems for law offices
- Pharmaceutical Anti-drug Counterfeiting

1.3. Design Considerations on Authentication Protocols

RFID technology is considered as "the next big thing" in management because it enables the improvement and optimization of multiple business processes, automation and elimination of existing processes and creation of smart processes that automatically trigger actions or events [3]. However, RFID systems have many security risks that are arising from insecure wireless communication between the tag and the reader, accessibility of tags by any reader, tampering tags and design flaws in reader implementations due to the cost constraint [6]. The resulting vulnerabilities are eavesdropping, scanning, cloning, man-in-the-middle attack, denial of service, physical attack and tracking. Making RFID system secure is a challenging issue because of its limited storage and computation capability.

Authentication protocol is the measure of the security for RFID systems. We can easily solve security and privacy problems of RFID system with a well-designed authentication protocol based on strong cryptographic solutions. However, these solutions can not be directly applied to RFID systems because standard cryptographic algorithms need more storage and computation resources than that are available in low cost RFID tags to be implemented. Computation performance of the server is also important issue for scalability of the RFID systems. Another important factor is asymmetric communication channel. Signals from reader to tag are more vulnerable

to eavesdropping than signals from tag to reader. Therefore the protocol designed for RFID authentication must consider not only security and privacy threats but also storage and computation capabilities of RFID tags, servers and properties of protocol environments.

An authentication protocol must do the followings to achieve perfect privacy and security:

1. Authentication protocol must provide mutual authentication between a tag and a server. For each session, the tag must have the fresh server validator to authenticate the server and the server must have fresh secrets of the tag to authenticate it.
2. Authentication protocol must resist to adversaries trying to identify, trace or link tags. Each entity must update its shared secrets after all successful authentication sessions to remove traceability. Messages exchanged between two entities must be anonymous to resist distinguishability and linkability.
3. Authentication protocol must resist DoS attacks. Each entity must keep themselves synchronizing with other entity. The server must store previous secrets of each tag to resist DoS attacks.
4. Each communication flow in the protocol must contain sensitive information to update shared secrets and the server validator. Therefore an adversary knowing internal state of the tag has to eavesdrop all communication flows to compute the new internal state of the tag and to trace the tag in the future.

1.4. Contributions

RFID is not a new and exciting technology. All of us can see RFID applications implemented in different purposes in our daily life. However, privacy and security issues of RFID system are still the most prominent obstacles to become widespread this

technology in various areas. A bewildering number of privacy and security proposals have been proposed to eliminate security and privacy threats to RFID systems.

In this thesis, we try to give a brief introduction to RFID systems and an overview of some proposed security and privacy solutions and their attributes. Furthermore, we propose a strong privacy model (ACAP) for RFID systems and analyze security of a former proposal[7] based on this model. We propose a new secure authentication protocol (ACA) for RFID systems to protect privacy and resist security vulnerabilities. Our main difference from former proposed protocols is to have resistance to server impersonation. In [7], server impersonation is considered and a protocol is proposed under an assumption that an adversary does not have access any reader-to-tag communication flows. Our protocol does not need any assumption to resist server impersonation. Furthermore, in [7], forward untraceability is provided with an assumption that the adversary misses a single flow from tag-reader interactions after the tag compromised. However, this protocol does not provide forward untraceability as claimed. Our protocol makes all communication flows mandatory to trace future transactions. We show that it provides more security features than previously proposed solutions. It also achieves efficient and effective identification with low storage and computation needs.

Another work in this thesis is that we examine another former proposal (SAPA)[8]. We discover that SAPA does not provide both location and information privacy. We also show that SAPA does not resist denial of service attacks, forward traceability, and server impersonation. After analyzing the weaknesses of SAPA, we propose a new secure authentication protocol (ACAT) for RFID systems to protect privacy and resist security vulnerabilities. The new protocol provides tag information privacy, and tag location privacy and resists tag impersonation, replay attack, denial of service attacks, backward traceability, forward traceability (under an assumption), and server impersonation. In our protocol, we prevent location tracking and information leakage by hiding the path key XORing it with the temporary secret that is created by the tag.

We prevent the denial of service attack by keeping the leaf node storing the most recent state of the tag in the tree. Furthermore, the key search complexity on the server side is nearly logarithmic, while the computation on the tag side is largely reduced. As a result, our scheme achieves efficient and effective identification with low storage, computation and communication needs.

1.5. Outline

The remainder of the thesis is organized as follows:

- In Chapter 2 we present RFID system primer. The components of RFID systems are described. We give the identified security and privacy requirements of RFID systems. Attacks related to privacy of tags are described. Further, attacks related to security vulnerabilities of tags and readers are introduced.
- In Chapter 3 we introduce some of related proposals and their attributes. We explain the basic principles of each proposal and their security vulnerabilities discovered in other works.
- In Chapter 4 we propose a new RFID privacy model (ACAP). We explain the details of the model. This work is accepted to Workshop of Information Security and Practice (WISTP) 2009.
- In Chapter 5 we analyze the security and privacy of a former RFID protocol. We explain its weaknesses based on our RFID privacy model. We propose a new RFID authentication protocol (ACA). ACA uses only XOR operation and one-way hash functions to meet the scarce resources of RFID tags. It is the first protocol that provides resistance against server impersonation attack without any assumption. It does also not need any extra computation burden on the tag to provide enhanced security and privacy features. We analyze the security and privacy of ACA by comparing its security and privacy features with that of some related works. We give the performance analysis of ACA by comparing its

storage and computation requirements with that of some related works. This work appears in proceedings of IEEE International Parallel and Distributed Processing Symposium (IPDPS) 2009.

- In Chapter 6 we analyze the security and privacy of an another former RFID protocol. We explain its weaknesses. We propose an another RFID authentication protocol (ACAT). ACAT achieves security and privacy goals with high efficiency. We compare security and privacy features of ACAT with that of some related works. We give the performance analysis of ACAT by comparing its storage, computation and communication requirements with that of some related works. This work appears in proceedings of IEEE 2009 Global Communications Conference (GLOBECOM 2009).
- In Chapter 7 we conclude this thesis.

2. OVERVIEW OF RFID SYSTEMS AND THEIR SECURITY AND PRIVACY ISSUES

2.1. RFID Systems

RFID system fundamentally consists of three elements: an RFID tag, an RFID reader and a back-end server. Basically, the reader queries the tag, supplies energy to it, collects its response, and then forwards its response to the back-end server. Generally, the back-end server has a database that contains the tag identifier and information related to the tag.

2.1.1. RFID Tags

RFID tag is a radio device consists of an antenna and a small silicon chip with some data storage, a control logic, a transmitter, a receiver and a power supply. The size of the tag depends on the size of the antenna, which increases with the range of the tag and decreases with the frequency. RFID tags can be categorized based on three main criteria [9].

Figure 2.1. RFID tags

1. **The Frequency of Operation:**

   Low Frequency (LF) Tags: Frequency range is 125-134.2 KHz. LF tags are less affected by metal or liquids. They are more expensive than other tags because they require copper in its production.

   High Frequency (HF) Tags: Frequency is 13.56 MHz. HF tags are cheaper than LF tags. However, they can be affected by metal or liquids.

   Ultra High Frequency (UHF) Tags: Frequency range is 860-960 MHz. They are cheaper to manufacture. However, they can be affected easily by metal or liquids.

   Microwave Tags: Frequencies are 2.4 GHz and 5.8 GHz. Microwave tags require line of sight for long distance communication.

2. **Powering Technique:** The power system of a tag can be completely powered by the incoming RF signal or it can have its own battery. Table 2.1 shows three categories of RFID tags [6].

   Table 2.1. Comparison of passive, semi-passive and active tags

	Passive	**Semi-Passive**	**Active**
Powering	Electromagnetic field	Battery	Battery
Transmission	Electromagnetic field	Electromagnetic field	Battery
Read Range	3 m -5 m (UHF)	10 m -20 m	1000 m

   Passive Tags: Passive tags are smaller and cheaper than active tags because they do not have a battery. However, they must be very close to the antenna to work because they completely obtain its power from incoming RF signal by coupling the electromagnetic field of the antenna.

   Semi-Passive Tags: Semi-passive tags are between the passive and active tags. They have a battery, but they rely on power obtained from the electromagnetic field of the antenna to transmit a message.

   Active Tags: Active tags have their own battery. They do not need to be powered by the electromagnetic field of the reader's antenna. They are able

to send or receive data over longer distances [2].

3. **Reprogrammability:**

   Read Only Tags: "Read only" tags are programmed by the manufacturer.

   Write-Once, Read Many Tags: "Write-once, read many" tags are programmed by the customer at application level but they are not reprogrammed.

   Rewritable Tags: The customer can program "rewritable" tags more than once.

2.1.2. Electronic Product Code (EPC)

The Electronic Product Code (EPC) is a scheme designed for universal object identification. The EPC is a unique number that is stored on an RFID tag to identify a specific item. At the beginning, EPC was developed by the Auto-ID Center created in October 1999 at the MIT Department of Mechanical Engineering. The Auto-ID Center completed its work in October 2003 and EPCglobal Inc took over its technology [9]. EPCglobal is a not-for-profit worldwide organization.

EPC tag uses the EPC tag data standard. A binary format of EPC tag data standard is shown in Table 2.2. The Header identifies the EPC format used by the tag, the General Manager Number identifies the company or organization, the Object Class is used to break down products into groups and the Serial Number is the unique identifier for each object[6].

Table 2.2. A binary format of EPC tag data standard

For 96 bit format			
Header	General Manager Number	Object Class	Serial Number
8-bit	28-bit	24-bit	36-bit

2.1.2.1. Classification of RFID tags. The Auto-ID Center has classified RFID tags based on their functionality. Table 2.3 gives the tag class hierarchy. The tag class hierarchy is important because environmental, functional, computational, security and privacy requirements are varying depending on the RFID application. RFID tags with different specifications can be developed based on this hierarchy.

Table 2.3. RFID tag class hierarchy

Class 1	Class 1 RFID tags are passive tags that have read only write once EPC code and a password. They also carries a CRC for transmission verification.
Class II	Class II RFID tags are passive tags with limited read range. In addition to abilities of Class I, they have write many ability and provision for security and privacy.
Class III	Class III tags contain a battery. They can be active or semi passive tags.
Class IV	Class IV tags can communicate with other Class IV tags.

2.1.3. RFID Readers

Another component of a basic RFID system is the reader composed of a separated or an integrated antenna, a power supply, a cryptographic encoding and decoding circuitry and a control unit. The reader is a device that retrieves information from a tag [2]. It sends a pulse of radio energy to query the tag and listens for the tag's response. The tag detects this signal and sends back a response that contains the tag's identification information [4]. After retrieving tag's response, the reader sends it to the back-end server. The server can retrieve detailed information of the tag by querying its database with the tag's response. RFID readers have more computation and storage capabilities than those of tags. They are capable of carrying out all kind of cryptographic operations. Depending on the application, RFID readers can be man-

ufactured in different size changing from the size of postage stamp to the size of a desktop personal computer.

Readers come in different prices, depending on their functionality. Basic readers have on board computing power and are able to filter data, store information and run applications. Advanced readers can communicate with tags using a variety of protocols and read tags using different frequencies.

Readers can have serial, Ethernet, Wi-Fi or USB ports. They use one or more ports for connecting separated antennas. They are also able to connect to external devices, a computer and a network.

2.1.4. Middleware

Middleware is the interface needed between the reader and the back-end server. Each event must be managed in RFID systems to keep information in synchronization between the back-end server and tags. Middleware provides the right information to readers and the back-end server. For example, RFID readers can query the same tag several times per second. The middleware takes raw data from the reader, filters it and sends the useful data to the back-end server.

Middlewares have different functionality. Basic middlewares perform a basic filtering. Advanced middlewares also perform additional functions to filter data. Some middlewares manage RFID readers: They monitor their health, configure them, send software updates and so on.

2.1.5. Back-End Server

The back-end database that stores all required information for tags can be a standard commercial database such as SQL and Oracle. The communication between a reader and the database is assumed to be secure. Depending on the application, the back-end server can run on a single PC or multiple mainframes networked together via global communication systems[2].

2.2. Security and Privacy Issues

There are numerous vulnerabilities for low cost RFID systems. These vulnerabilities lead researchers to consider security and privacy issues because the acceptance of RFID technology depends on addressing them.

2.2.1. Privacy Issues

The privacy issues in RFID systems are related to the privacy of tags. Tags contain information about a person, an object or a product to which they are attached. When they are communicating with the reader, they send their information in some way. According to the interest of an adversary, two types of privacy can be defined: data privacy and behavior privacy. The adversary may want to get information stored in the tag or information about the location of an individual or a product. In [7], they are defined as tag information privacy and tag location privacy.

2.2.1.1. Information Privacy. If tags store sensitive information, a passive adversary could get this information by simply querying tags. For example, when you borrow a book in a library, an attacker having a reader find out what kind of book you like, the name of the book and the name of the author, if a tag attached to your book does not authenticate the legitimate reader and sends messages in plaintext. This type of

attack can be avoided by encrypting messages.

2.2.1.2. Location Privacy. If tags send their static ID in plaintext or send same encrypted message for each challenge, a passive adversary can distinguish a specific tag from others. Therefore, an attacker having a reader can trace the location of an individual. For example, if you have a travel card with RFID chip, the attacker can find out locations you travel because usually RFID chips in travel cards give same response to all queries. Anonymous answers that are generated by encrypting static ID with some random values can avoid this type of attack.

2.2.2. Security Issues

Several common types of attacks can be applied to RFID systems; eavesdropping, replay attack, man-in-the-middle attack, and denial of service (DoS) attack.

Eavesdropping: Communication channel between the tag and the reader can be observed by a passive adversary. The reader-to-tag channel (forward channel) can be monitored from a long distance. The signal transmitted by the tag is relatively weak; therefore the tag-to-reader channel (backward channel) can be monitored from a short distance.

Replay attack: A passive adversary successfully eavesdrops a conversation between the tag and the reader and retransmits these conversations to query the tag or the back-end server to impersonate the valid tag or the valid reader. The most suitable way to avoid replay attack in RFID protocols is to use a nonce [9]. Cai[10] shows an example of replay attack by discovering vulnerabilities of an RFID protocol that result from $\oplus$ and $\gg$ operations.

Man-in-the-middle attack: An adversary monitors a conversation between the tag and the reader and alters exchanged messages and sends them to the reader or the

tag to obtain information. Inexpensive security operations such as AND, OR, $\oplus$, $\gg$ and $\ll$ gives the opportunity of creating valid messages from messages created by the valid reader or the valid tag.

DoS attack: DoS attack can be implemented by placing a large number of fake tags for identification by the reader or corrupting large batch of tags[6]. Moreover, an adversary can prevent the tag or the reader from reading messages. This results in some synchronization problems between the tag and the server.

Cloning attack: Device that is designed specially can impersonate a tag with data obtained from the valid tag. The reader can not verify that whether it is communicating with the valid or the fake tag [6]. For example, a thief may replace an expensive item attached to the valid tag with a cheap item attached to the fake tag.

Compromising Attack: RFID tags offer no tamper resistance because tamper-proofing is an expensive option for low cost tags [6] so the content of tags is vulnerable to physical attacks. A strong adversary can compromise a valid tag to retrieve its stored secrets and impersonate the valid tag to the legitimate reader.

Backward Untraceability: The concept of forward security is introduced in [11]. Forward security means that a strong adversary compromising a tag at time t can not trace the past interactions of the tag that occurred at time $t' < t$. The forward security is referred as *backward untraceability* in [12, 7]. Generally, forward security is provided by updating secrets of the tag with one-way hash function.

Forward Untraceability: The concept of forward untraceability is introduced in [12]. Forward untraceability means that a strong adversary compromising a tag at time t can not trace the future interactions of the tag that occurred at time $t' > t$. There is not a proposed solution that achieves forward untraceability without any assumption.

Server Impersonation: The concept of server impersonation is introduced in [7]. In this attack, a strong adversary knowing the internal state of a tag can impersonate

the valid server to the tag. There are also some weak proposals such as protocol in [7] that a passive adversary can impersonate the valid server

2.2.3. Challenges

RFID security and privacy are the challenging research areas. Contemporary cryptographic solutions can not be used because of the scarce power and storage recourses of RFID tags. The limited memory and silicon area on a tag should be kept as minimum as possible to make the cost of the tag cheaper. Transmit power restrictions, available bandwidth, time available for computations, power consumptions, and power disruptions must be considered when designing a security system for low cost RFID [6].

If the price of a tag becomes cheaper, the RFID technology could be more widespread easily. The price of a tag can be reduced by using less silicon or another material cheaper than silicon in chip production.

An RFID chip can be designed up to $1mm^2$ in size. Stephen Weis says that 10000 gates can be squeezed into this area [13, 14]. Martin Feldhofer estimates the number of gates squeezed into this area as 20000. He also says that 5000 gates can be used for security purposes [15]. In [15], he implements encryption only AES block cipher in an RFID chip using 3628 gates.

Passive tags take its power from electromagnetic field of reader signal. The power supplied to tags can be increased by using low frequency instead of high frequency, increasing the transmission strength or using a larger antenna [13].

Computation performance of the tag and the server are also important issues for scalability of RFID systems. When the reader receives response of a tag, it forwards

this response to the back-end server. The back-end server identifies the tag using the list of identifiers. In a large scale RFID system, there are billions of tags. The back-end server may be queried for multiple tags at the same time so it should answer these queries with high efficiency.

2.2.4. Hash Function

In this section, we give general information about the cryptographic hash functions.

A hash function H takes a variable size bit string M and produces a fixed-size bit string h as output $h = H(M)$. For example, MD5 developed by Ron Rivest produces 128-bit message digest. Another famous example of hash functions is SHA-1 takes a bit string with length less than 2^{64} bits and produces 160-bit digest. In cryptographic protocols, hash functions provide data integrity and serve as authenticator to provide message authentication.

A hash function must have the following properties to be useful for RFID authentication [16]:

1. Given any size of data as input the hash function produces fixed-length of output.
2. Given an input M, it is relatively easy to compute output $H(M)$.
3. Given an output $h = H(M)$, it is computationally infeasible to find M such that $H(M) = h$. In the literature, this is referred as one-way property.
4. It is computationally infeasible to find (M, N) pair satisfying $H(M) = H(N)$. This is referred as strong collision avoidance.

If a hash function does not have any of these properties, it does not resist first preimage attacks, second preimage attacks and collision attacks.

The length of the hash code is important to resist Birthday attacks. For example, if the length of the hash code is 64-bit, an attacker would have to try 2^{32} messages to find a collision. Oorschot and Wiener [17] designed a collision search machine for MD5 that could find a collision in 24 days. Therefore, 128-bit length is not secure, hash functions should use the hash code longer than 160-bit.

In RFID authentication protocols, generally hash functions are used to update secrets of the tag and to create authenticators for the tag and the server. They also provide data integrity for exchanged messages. RFID authentication protocols take advantage of one-way property of hash functions.

2.2.5. Random Number Generator

Random numbers play an important role in RFID authentication. They are used to prevent replay attacks, to create an authenticator and to update secrets of the tag.

First requirement for a sequence of random numbers is randomness[16]. If the distribution of numbers in the sequence is uniform we can say that the sequence is random. Another requirement is unpredictability. Each number can not be predicted using previous numbers. Therefore each number must be statistically independent from other numbers.

Source of true random numbers are physical noise generators, gas discharge tubes, leaky capacitors and etc...[16]. In network security applications, true random generators are difficult to build, so cryptographic algorithms are used to produce pseudo random numbers. However, these algorithms are deterministic and produce statistically random looking distribution. They are called as Pseudo Random Number Generator (PRNG). The goodness of an algorithm is tested with many tests of randomness.

3. OVERVIEW OF AUTHENTICATION IN RFID

Many solutions have been proposed to achieve private authentication in RFID systems. However, in this chapter, we only describe some authentication protocols that provide some improvements to RFID authentication. We aim to give the fundamental principles and critical reviews of each protocol instead of giving their full details.

3.1. Weis, Sarma, Rivest and Engels

HashLock, the simplest form of cryptographically controlled access is proposed in [18]. This protocol is low cost since it requires only a hash function. The tag avoids illegal ID scanning and ID leakage by sending $metaID$ that it stores. The

Table 3.1. The HashLock protocol

Server $[k, metaID, ID]$		**Reader**		**Tag** T_i $[metaID_i, ID_i]$
			$request$ $\longrightarrow$	
	$metaID_i$ $\longleftarrow$		$metaID_i$ $\longleftarrow$	
Search for $metaID_i$ if it is found,	k_i', ID_i' $\longrightarrow$		k_i' $\longrightarrow$	
			ID_i $\longleftarrow$	if $h(k_i') = metaID_i$

tag receives a request from the reader and sends $metaID$ to it. The reader looks up for $metaID$ in the database and then sends the key k related to $metaID$ to the tag. The tag calculates $h(k)$ and checks equality between its $metaID$ and the calculated

$h(k)$. If equality holds, the tag responds to the reader with its ID. The HashLock protocol offers data privacy but a passive adversary can track a tag via its $metaID$, since $metaID$ is fixed. Another drawback of this protocol is that the key k is sent in plaintext over the forward channel which can be easily eavesdropped.

Randomized HashLock protocol is proposed in [14]. This is an extension of the HashLock type protocol. It requires only a hash function and a pseudo-random generator in the tag. In this protocol, after receiving authentication request, the tag calculates hashed value of ID and random number r as M. The tag then responds to the reader with M and r. The reader forwards M and r to the back-end server. The reader computes $h(ID, r)$ for each stored key. After identifying the correct key, the server sends it to the reader. The search complexity of the randomized HashLock protocol is linear to the number of tags in the system. The randomized HashLock does not provide forward secrecy because a passive adversary can obtain the location history of a tag if the key k is exposed.

Table 3.2. The randomized HashLock protocol

Server $[ID]$		**Reader**		**Tag** T_i $[ID_i]$
			$request$ →	
				$r \in \{0,1\}^l$ $M = h(r, ID_i))$
	r, M ←		r, M ←	
Search for ID_i if $h(r, ID_i') = M$,				
	ID_i' →		ID_i' →	
				if $ID_i' = ID_i$ unlocks itself

Many approaches have been proposed to avoid brute force searches. These approaches are classified by Juels [19] into tree type.

3.2. Synchronization Approaches

One solution to achieve private authentication in RFID systems is synchronization approach. In this approach, both the tag and the reader keep a counter in synchronization and the tag uses this counter in its replies. The reader keeps a few possible counter values for each tag and updates these values after each successful authentication.

3.2.1. Ohkubo, Suzuki and Kinoshita

Ohkubo et al. propose to use hash-chains and stores initial values of hash chains in the back-end server [20]. The tag responds to an identification request with the hashed value of its current identifier M and updates its identifier using another hash function after each identification request. The reader constructs a hash chain from each initial value until it finds response of the tag M. While this protocol provides backward untraceability, it is vulnerable to replay attacks. The modified version of Ohkubo's

Table 3.3. The OSK protocol

Server $[ID, S^1]$		**Reader**		**Tag** T_i $[S_i^k]$
			$request$ $\longrightarrow$	
	M $\longleftarrow$		M $\longleftarrow$	$M = g(S_i^k)$ $S_i^{k+1} = h(S_i^k)$
Search for S_i^1 for $j = 1 : m$ if $g(h^j(S_i^1)) = M$,	$ID_i^{'}$ $\longrightarrow$			

protocol [11] is proposed to reduce the computation cost of the back-end server and

tags. The authors propose not to update the identifier after each identification request. They use a counter c and define upper bound for it. The counter is augmented after each identification request. If the counter reaches its upper bound, the tag updates its identifier. However, this protocol degrades forward security because last c interactions of the tag can be traced without corrupting it and unsuccessful authentications make this protocol vulnerable to replay attacks.

3.2.2. Henrici and Müller

In protocol proposed by Henrici [21], the tag counts the read attempts since the last successful authentication and sends this value to the reader. This eliminates the vulnerability of [11] to replay attacks. A passive adversary can track a tag by artificially augmenting the number of read attempts. In [22], Avoine shows that a passive adversary can easily break the synchronization between a tag and the back-end server by replacing r value with 0 and M_3 value with M_2 that is eavesdropped from the second communication flow of the protocol.

Table 3.4. Henrici and Müller's protocol

Server $[ID, k, k_{last}]$		**Reader**		**Tag** T_i $[ID_i, k, k_{last}]$
			$request$ $\longrightarrow$	
	$M_1, M_2, \triangle k$ $\longleftarrow$		$M_1, M_2, \triangle k$ $\longleftarrow$	$k = k + 1, \triangle k = k_{last} - k$ $M_1 = h(ID_i), M_2 = h(ID_i \oplus k)$
Search for ID_i if $h(ID_i = M_1)$ $k^{'} = k_{last} - \triangle k$ if $h(ID_i \oplus k^{'}) = M_2$ picks r $M_3 = h(ID_i \oplus k^{'} \oplus r)$	M_3, r $\longrightarrow$		M_3, r $\longrightarrow$	
$ID_i = r \oplus ID_i$				if $M_3 = h(ID_i \oplus k \oplus r)$ $ID_i = r \oplus ID_i, k_{last} = k$

3.2.3. Dimitriou

In [23], Dimitriou eliminates the issue of desynchronization entirely by using mutual authentication of the tag and the reader. He designed a protocol with both tag-to-reader and reader-to-tag authentication in mind; unless both types of authentication are applied.

Table 3.5. Dimitriou's protocol

Server $[ID]$		**Reader**		**Tag** T_i $[ID_i]$
		$r_1 \in \{0,1\}^l$	r_1 ⟶	
	r_1, r_2, M_1, M_2 ⟵		r_2, M_1, M_2 ⟵	$r_2 \in \{0,1\}^l$ $M_1 = h(ID_i)$ $M_2 = h_{ID_i}(r_1, r_2)$
Search for ID_i if $h(ID_i') = M_1$, if $h_{ID_i'}(r_1, r_2) = M_2$, Computes ID_{i+1} $M_3 = h_{ID_{i+1}}(r_1, r_2)$	M_3 ⟶		M_3 ⟶	
				Computes ID_{i+1} $M_3 == h_{ID_{i+1}}(r_1, r_2)$

Dimitriou's protocol is based on the use of a secret shared between the tag and the database that is updated to avoid tag tracing. This is done in such a way so that efficiency of identification is not sacrificed. The drawback of this protocol is that the tag ID stays constant between two identification sessions. During this time interval, a passive adversary can track a tag.

3.3. Tree Based Approaches

In tree-based protocols, the reader has a tree structure to store secrets of tags.

3.3.1. Molnar and Wagner

First tree-based approach is proposed by Molnar and Wagner [24]. Tree structure is constructed by the reader using unique keys. Every node in the tree except the root stores an unique key. Tags are considered as leaves in a balanced tree. Each tag stores $\log N$ keys corresponding to the root to a leaf associated with itself. When a tag needs to authenticate the reader, it uses a challenge-response protocol for each key. The reader walks the tree from the root to leaves by checking whether the tag contains a key in the left hand or the right hand of the tree. If the tree has a depth d and a branching factor α, the tree can support N=α^d tags. For one authentication, the reader searches $\alpha \cdot d$ keys so this protocol has logarithmic lookup properties O($\log_\alpha N$). Figure 3.1 represent a tree with $\alpha = 2$ and $n = 8$. The depth of the tree is 3. Keys of T_4 are $[k_0, k_{1,1}, k_{2,2}, k_{3,4}]$. One round of the protocol is depicted in Table 3.6. This protocol does not provide backward untraceability due to the lack of key-updating mechanism. Updating keys is a challenging issue because the updating mechanism leads to the reader work in the number of possible time periods, but the reader does not know at which time period the tag is[25].

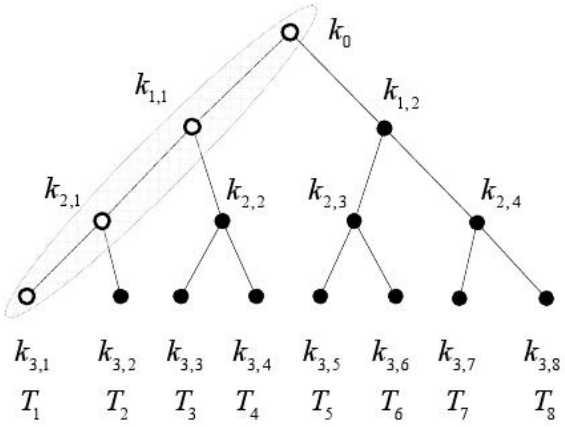


Figure 3.1. The balanced tree of Molnar and Wagner's protocol [1]

Table 3.6. Molnar and Wagner's protocol

Server The key tree		**Reader**		**Tag** T_i $k_i^0, k_i^1, k_i^2, k_i^3]$
		$r_1 \in \{0,1\}^l$	r_1 →	
	r_1, r_2, M_1 ←		r_2, M_1 ←	$r_2 \in \{0,1\}^l$, $k = k_i^0$ $M_1 = k \oplus f(0, r_1, r_2)$
Search for k_i^0 in the child nodes of the root if $M_1 = k' \oplus f(0, r_1, r_2)$, $M_2 = k' \oplus f(1, r_1, r_2)$	M_2 →		M_2 →	
root=node of k'				$M_2 == k \oplus f(1, r_1, r_2)$

3.3.2. Lu et al.

In [1], a strong and lightweight RFID Private Authentication protocol, SPA, which enables dynamic key-updating for tree based authentication approaches is proposed. This protocol achieves forward secrecy without degrading key search efficiency and reduces key relationships between compromised tags and uncompromised ones by recursively updating keys in the tree.

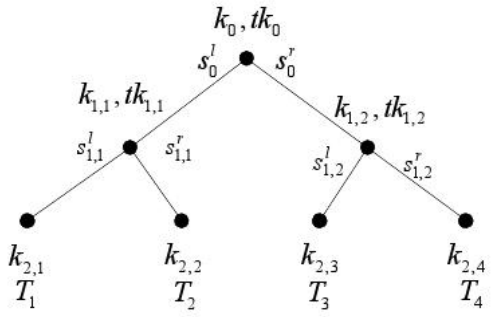


Figure 3.2. The key tree of Lu et al.'s protocol [1]

The basic idea behind the updating mechanism is that each non-leaf node uses a number of state bits to reflect the key-updating status of its children. Once a child has

updated its key, the corresponding bit is set to 1. Each node updates its own key when all its state bits become 1. The authors claim that their protocol outperforms existing designs in defending against both passive and active attacks, including compromising attack. However, this solution reduces key relationships as a whole but local key relationships are still kept as static. Therefore compromising a tag still imperils the privacy of its brother tag seriously.

3.3.3. Wang et al.

Wang et al.[8] propose storage awareness RFID authentication protocol based on sparse tree structure (SAPA) which provides backward untraceability and reduces the space of a tag for storing a key sequence. This protocol removes all key relationships by considering the path of the tag as an individual secret. In key-updating phase, the reader removes a branch corresponding to the path key, updates the path key, inserts a branch corresponding to it and then sends a synchronization message to the tag. Upon receiving the synchronization message, the tag updates its path key. However, this protocol is susceptible to tag location tracking and tag information leaking because a hash-chain calculated by the tag does not provide any security to hide the path of the tag.

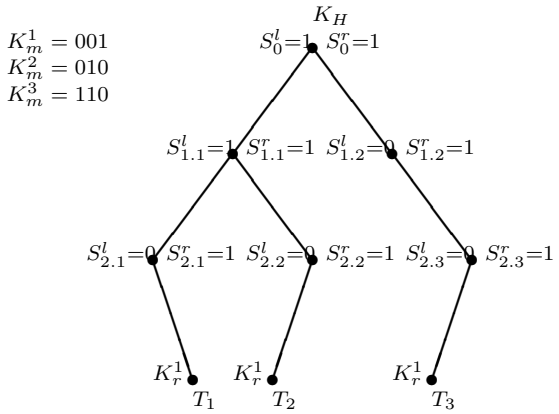


Figure 3.3. The sparse tree of SAPA

3.4. Time-Space Trade-off Approaches

3.4.1. Avoine and Oechslin

Avoine et al. [26, 27] modify Ohkubo et al.'s protocol [20] as depicted in Table 3.7. He uses a fresh challenge r_1 sent by the reader to avoid replay attacks and add the third message created with a fixed and public non-zero binary string w to provide reader authentication. Furthermore, Avoine et al. [27] observe that the key search problem in RFID systems is nearly identical with the problem of breaking symmetric keys. In [28], Hellman studies the key-breaking problem and claims that recovering a symmetric key k from a ciphertext needs pre-computation and storing $O(N^{2/3})$ possible keys. Avoine applies a variant of the Hellman technique to Ohkubo et al.'s protocol in [20] to enhance lookup costs considerably. Because of needing pre-computation of the Hellman table, like the protocol in [20], protocols in [26, 27] searches for keys over a pre-determined window of time steps [19].

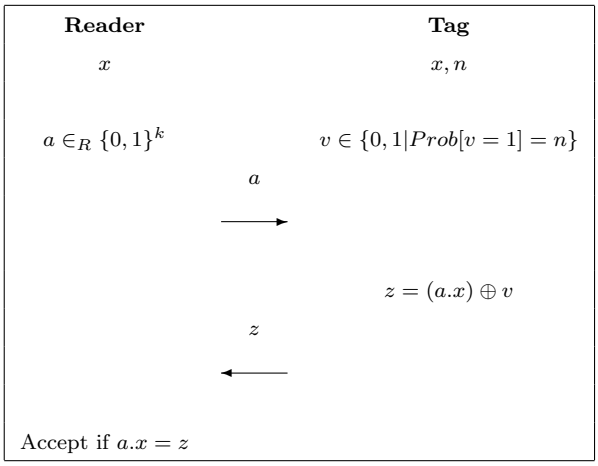
Table 3.7. Ohkubo's modified protocol

Server $[ID, S^1, w]$		**Reader**		**Tag** T_i $[S_i^k, w]$
		$r_1 \in \{0,1\}^l$		
			r_1 ⟶	
	M_1 ⟵		M_1 ⟵	$M_1 = g(S_i^k \oplus r_1)$ $S_i^{k+1} = h(S_i^k)$
Search for S_i^1 by using Hellman technique for $j = 1 : m$ if $g(h^j(S_i^1) \oplus r_1) = M_1$, $M_2 = g(S_i^{k+1} \oplus w)$	M_2 ⟶		M_2 ⟶	
				$g(S_i^{k+1} \oplus w) == w$

3.5. HB Based Protocols

In 2001, Hopper and Blum[29] proposed an authentication protocol to secure authentication and identification for unassisted humans known as the HB protocol. The security of HB protocol relies on the computation hardness of the Learning Parity with Noise (LPN) problem. The HB protocol is very lightweight because it uses only dot products of binary vectors and a random noise bit. The authors say that their solution is impractical for humans. In 2005, Juels and Weis[30] investigated that RFID

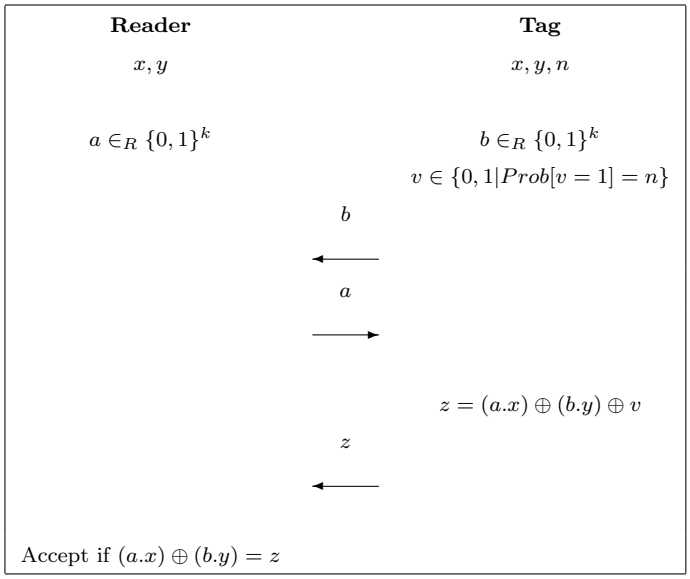
Table 3.8. HB protocol round



tags share similar capabilities with another weak computing device: people. They adopt HB protocol to the pervasive computing setting and propose a new symmetric authentication protocol with a simple, low-cost implementation called HB$^+$ protocol. The authors prove the security of the protocol against active adversaries. In [31], the authors show that HB$^+$ is vulnerable to an efficient man-in-the-middle attack with linear computational and communication complexity. In 2006, Bringer et al. [32] proposed a modified version of HB$^+$ called HB^{++}. The HB^{++} protocol removes the vulnerabilities of HB$^+$ against active attacks.

Piramuthu [33] evaluates HB and its variants for RFID tag/reader authentication.

Table 3.9. HB$^+$ protocol round



He also shows the vulnerability of HB^{++}, and present another modified version of HB$^+$ protocol. Although the proposed method is not secure against all types of attack by an adversary, it is reasonably secure against those that were considered while maintaining lightweight characteristic of the protocol.

In 2007, Duc and Kim [34] presented a variation of HB+ protocol called HB*, which is resistant to Gilbert et al.'s attack [12]. Piramuthu [35] breaks HB* and proposes his modified protocol.

Munilla and Peinado [36] propose a new lightweight authentication protocol called HB-MP. The HB-MP protocol has a fresh way of exchanging messages and improved attack resistance to active attacks and the man-in-the-middle attack applied to HB and HB$^+$

In 2008, Gilbert et al. [37] proposed two new lightweight authentication protocols called random-HB$^\#$ and HB$^\#$. Both random-HB$^\#$ and HB$^\#$ offer practical improve-

ments over HB^+. Random-$HB^\#$ is provably secure in the detection-based model, just like HB^+, but it is also provably resistant to a broader class of active attacks. $HB^\#$ trades some of the theoretical underpinnings to random-$HB^\#$ and attains a truly practical performance profile.

Quafi et al. [38] present a general man-in-the-middle attack against $HB^\#$ and random-$HB^\#$ which can also be applied to all HB based protocols, that recovers a shared secret in 2^{25} or 2^{20} authentication rounds for $HB^\#$ and 2^{34} or 2^{28} for random-$HB^\#$.

Hammori and Sunar [39] propose a light-weight authentication protocol called PUF-HB for low-power devices. The PUF-HB protocol provides low-cost tamper-resilient challenge-response authentication using physically unclonable functions and provides provable cryptographic strength against passive adversaries by using the principles of the HB protocol. The authors rigorously prove the security of the proposed protocol against active attacks except man-in-the-middle attacks.

Leng et al. [40] propose an enhanced version of the HB-MP authentication protocol, called the HB-MP^+ protocol. The HB-MP^+ protocol overcomes the man-in-the-middle attack to which the basic HB-MP protocol is vulnerable. The authors analyze the security and performance improvements of HB-MP+ protocol.

3.6. Karthikeyan and Nesterenko

In [41] a protocol based XOR operations and a simple matrix multiplication is proposed. This protocol does not need extensive cryptographic operations. Security is based on the difficulty of recovering the multiplier from the product of two matrices [7]. This protocol can not resist DoS attack, replay attack and individual tracing [42]

Table 3.10. Karthikeyan and Nesterenko's protocol

Reader		**Tag**
K, M_1^{-1}, M_2		K, M_1, M_2^{-1}
	request $\longrightarrow$	
		$X = KM_1$
	X $\longleftarrow$	start timer
Identify tag by matching X		
$Y = (K_1 \oplus K_1 \oplus ... \oplus K_l)M_2$	Y, Z	
Picks unique X_{new}	$\longrightarrow$	stop timer
$K_{new} = X_{new}M_1^{-1}$		
$Z = K_{new}M_2$		if $YM_2^{-1} == (K_1 \oplus K_1 \oplus ... \oplus K_l)$
		$K = ZM_2^{-1}$

3.7. Duc et al.

A synchronization-based communication protocol for the EPCGlobal Class 1 Gen-2 RFID tag is proposed in [43]. It uses a pseudo-random number generator (PRNG)

Table 3.11. Duc et al.'s protocol

Server $[EPC, PIN, K_i]$		**Reader**		**Tag** T_i $[EPC, PIN, K_i]$
			request $\longrightarrow$	
				$r \in \{0,1\}^l$
	M_1, r, C		M_1, r, C	$M_1 = CRC(EPC\|\|r) \oplus K_i$
	$\longleftarrow$		$\longleftarrow$	$C = CRC(M_1 \oplus r)$
Search each tuple				
if $M_1 \oplus K_i = CRC(EPC\|\|r)$				
if $C = CRC(M_1 \oplus r)$	M_2		M_2	
$M_2 = CRC(EPC\|\|PIN\|\|r) \oplus K_i$	$\longrightarrow$		$\longrightarrow$	
$K_{i+1} = f(K_i)$				if $M_2 \oplus K_i = CRC(EPC\|\|PIN\|\|r)$
				$K_{i+1} = f(K_i)$

and a cyclic redundancy code (CRC). It can not resist DoS attack and key guessing attack. In addition, it does not provide backward untraceability if the tag is compromised [42].

3.8. Chien and Chen

Chien et al. propose a mutual authentication protocol based on the EPC Class 1 Generation 2 standards[42]. This protocol prevents replay attacks using challenge-response technology. The server database stores two sets of authentication key and access key to resist DoS attacks. This allows the server to authenticate the tag and re-synchronize itself with the tag. Secrets of the tag are updated after each successful session in order to give backward untraceability. However, the protocol still permits backward and forward traceability because a strong adversary compromising a tag can identify the tag's past interactions from the previous communications and can also read the tag's future transactions [7]. In addition, this protocol can also not resist server impersonation, if the tag is compromised.

Table 3.12. Chien's protocol

Server $[EPC_i, K_{new}, K_{old}]$		**Reader**		**Tag** T_i
		$r_1 \in \{0,1\}^l$	r_1 →	
	M_1, r_1, r_2 ←		M_1, r_2 ←	$r_2 \in \{0,1\}^l$ $M_1 = CRC(EPC_x \| \| r_1 \| \| r_2)$
Search each tuple $I_{new} = M_1 \oplus K_{new}, I_{old} = M_1 \oplus K_{old}$ if I_{old} or $I_{new} = CRC(EPC_x \| \| r_1 \| \| r_2)$ $M_2 = CRC(EPC_x \| \| r_2) \oplus P_{old}$ or P_{new}	M_2 →		M_2 →	
$K_{old} \leftarrow K_{new} \leftarrow PRNG(K_{new})$ $P_{old} \leftarrow P_{new} \leftarrow PRNG(P_{new})$				if $M_2 \oplus P_{x_i} = CRC(EPC_x \| \| r_2)$ $K_{x_i+1} \leftarrow PRNG(K_{x_i})$ $P_{x_i+1} \leftarrow PRNG(P_{x_i})$

3.9. Lim and Kwon

Lim and Kwon introduce the concept of forward untraceability and its importance in designing RFID security protocols and based on these observations he proposes a new authentication protocol[12]. This protocol provides both forward and backward untraceability and also enables perfect ownership transfer. If an authentication session fails, it updates secrets of the tag using a deterministic process (one-way key chain). If the authentication succeeds, then both the tag and the database update secrets of the tag probabilistically using the exchanged random numbers.

Lim and Kwon's protocol provides forward untraceability under an assumption that a strong adversary compromising a tag can not eavesdrops all the future interactions of the tag. Secrets of the tag are updated after each successful authentication session and thus a compromised tag becomes untraceable from the moment that the adversary misses even a single valid session of the tag. It uses two hash key chains: a forward key chain for tag secret evolution, and a backward key chain, used in reverse order, for the server validation. This protocol is secure against server impersonation. However, the server performs too many hash operations to compute the new value of server validator because the backward key chain used for server validation is in reverse order. In [44, 45], authors show how to trace a tag in this protocol without corrupting it so this protocol does not provide location privacy.

3.10. Song and Mitchell

In [7], Song and Mitchell introduce the concept of server impersonation as a novel type strong attack and she proposes a new authentication protocol. Forward untraceability and resistance to server impersonation are provided under an assumption. This protocol provides improved performance of the protocol in memory space, computation time, communication overhead with same security and privacy features in [12]. In [46],

the authors find a replay attack on tag authentication, so this protocol does not resist replay attacks.

Table 3.13. Song and Mitchell's protocol

Server $[(u_i, t_i)_{new}, (u_i, t_i)_{old}]$		**Reader**		**Tag** T_i $[t_i]$
		$r_1 \in \{0,1\}^l$	r_1 →	
	M_1, M_2 ←		M_1, M_2 ←	$r_2 \in \{0,1\}^l$ $M_1 = t_i \oplus r_2$ $M_2 = f_{t_i}(r_1 \oplus r_2)$
Search for t_i $r_2 = M_1 \oplus t_i$ if $M_2 = f_{t_i}(r_1 \oplus r_2)$, $M_3 = u_i \oplus (r_2 \gg l/2)$	M_3 →		M_3 →	
$u_{i(old)} = u_i$ $t_{i(old)} = t_i$ $u_{i(new)} = (u_i \ll l/4)$ $\oplus(t_i \gg l/4) \oplus r1 \oplus r2$ $t_{i(new)} = h(u_{i(new)})$				$u_i^{'} = M_3 \oplus (r_2 \gg l/2)$ if $h(u_i^{'}) = t_i$ $t_i = h((u_i^{'} \ll l/4)$ $\oplus(t_i \gg l/4) \oplus r1 \oplus r2))$

3.11. Cai et al.

In 2009, Cai et al. [10] analyze Song et al.'s protocol [7] and present reader impersonation attack, which enable an adversary to impersonate any legitimate reader. This attack can result in de-synchronized updating of the secret information between the tag and the server. Inexpensive security operations such as $\ll$ and $\oplus$ are extensively used in the design of Song's protocol. Although these operations can help to reduce the cost of RFID tags, they lead to various security vulnerabilities. The authors also propose a revision to eliminate the vulnerabilities. They increase the computation cost of the tag to eliminate proposed attack.

Table 3.14. Cai et al.'s protocol

Server $[(u_i, t_i)_{new}, (u_i, t_i)_{old}]$		**Reader**		**Tag** T_i $[t_i]$
		$r_1 \in \{0,1\}^l$	r_1 →	
	M_1, M_2 ←		M_1, M_2 ←	$r_2 \in \{0,1\}^l$ $M_1 = t_i \oplus r_2$ $M_2 = f_{t_i}(r_1 \| r_2)$
Search for t_i $r_2 = M_1 \oplus t_i$ if $M_2 = f_{t_i}(r_1 \| r_2)$, $M_3 = u_i \oplus h(r_2)$	M_3 →		M_3 →	
$u_{i(old)} = u_i$ $t_{i(old)} = t_i$ $u_{i(new)} = (u_i \ll l/4)$ $\oplus(t_i \gg l/4) \oplus r1 \oplus r2$ $t_{i(new)} = h(u_{i(new)})$				$u_i^{'} = M_3 \oplus h(r_2)$ if $h(u_i^{'}) = t_i$ $t_i = h((u_i^{'} \ll l/4)$ $\oplus(t_i \gg l/4) \oplus r1 \oplus r2))$

4. ACAP: A NEW RFID PRIVACY MODEL

RFID privacy model is formal definition of RFID protocols covering their security and privacy and the abilities of an adversary. RFID protocols must have security properties to provide tag-reader authentication and must have privacy properties to resist adversaries aiming to identify, trace or link tags. Privacy models are used to determine the privacy level of RFID protocols. They can have several privacy levels according to the abilities of the adversary.

4.1. Proposed RFID Privacy Models

There are several proposed RFID privacy models in the literature. In this section, we give a brief information about these models.

- **Avoine's Model:** Avoine has proposed a strong cryptographic model and defined the strong privacy notion of untraceability in RFID protocols [27]. This model has different level of privacy and defines different abilities for an adversary. Privacy is formalized by the ability to distinguish two known tags. This model excludes the availability of side-channel information.
- **Juels and Weis's Model:** Juels and Weis extend Avoine's model [47]. In this model, side-channel information is used and tags are chosen by an adversary. Juels and Weis find a powerful desynchronization attack on Avoine's model. Avoine shows OSK protocol[20] is secure under his model. In this model, this protocol is considered as insecure.
- **Lim and Kwon's Model:** Lim and Kwon propose a security model for untraceability by making Avoine's model more general and flexible by considering various possible restriction in terms of access time of the adversary and frequency that the adversary uses[12]. This model defines the strong privacy notion of un-

traceability especially forward and backward untraceability.

- **Ouafi and Phan's Model:** Ouafi and Phan provide a general untraceable RFID privacy model[44, 45]. They show that recently proved secure RFID protocols do not achieve the notion of untraceable privacy. Lim and Kwon consider the protocol in [12] as secure according to their model. Ouafi and Phan show that this protocol is not secure.
- **Vaudenay's Model:** Vaudenay proposes a new model that classifies privacy in RFID[48]. This model has eight classes of privacy levels. Vaudenay shows that strong privacy is not possible and point out an open question whether forward privacy with symmetric-key cryptography, is possible under a strong privacy model.

4.2. Detailed Description of ACAP

We define a privacy model mainly based on the models in [27, 12, 44, 45]. Our model differs from these models in some restriction on RFID systems in terms of accessing possibility of an adversary to each communication round. Generally, privacy models assume that the adversary accesses all communication rounds of a protocol if it calls an execute query. We know that the adversary possibly misses some communication rounds, therefore, we can say that it has some limitations in the number communication of rounds of a valid session that can be read by the adversary.

In our model, we do not follow Vaudenay's strong privacy model because it is not realistic to consider traceable privacy of a valid tag between two valid sessions. A strong adversary can only trace its own interactions with the valid tag between two valid sessions. It is not important in the privacy notion of untraceability. After each valid session, the adversary can not continue to trace the tag because the tag updates its secrets.

We have two access models: universal access model (UA) and restricted access model (RA). Our limitations on an adversary define our restricted access model (RA) like in [12].

In the model, a protocol $\mathcal{P}$ determines how protocol parties a $\mathcal{T} \in Tags$ and a $\mathcal{R} \in Readers$ interact each other in protocol sessions. An adversary $\mathcal{A}$ is a probabilistic algorithm and it controls the communications between the tag $\mathcal{T}$ and the reader $\mathcal{R}$ by calling the following queries:

$Execute(\mathcal{R}, \mathcal{T}, i)$: This query models attacks where an adversary $\mathcal{A}$ passively eavesdrops on the communication channel between $\mathcal{T}$ and $\mathcal{R}$ in the protocol session i and obtains the messages exchanged on both the forward and the backward channels.

$Execute^*(\mathcal{R}, \mathcal{T}, i, j)$: This query models attacks where an adversary $\mathcal{A}$ passively eavesdrops on the communication channel between $\mathcal{T}$ and $\mathcal{R}$ in the protocol session i and obtains the exchanged messages on communication round j.

$Send(U_1, U_2, i, m)$: This query models an adversary $\mathcal{A}$ actively impersonating a tag $U_1 \in Tags$ or a reader $U_1 \in Readers$ in protocol session i. It sends a message m to a reader $U_2 \in Readers$ or a tag $U_2 \in Tags$

$Corrupt(\mathcal{T}, K)$: This query models an adversary $\mathcal{A}$ learning the stored secret $K^{'}$ and which further sets the stored secret to K. This query is only used during the time interval of $\mathcal{A}$'s training phase.

We use the following notations in the model:

$\omega_i(\mathcal{T})$: The result of the query $Q \in Execute, Execute^*, Corrupt$ executed on a tag $\mathcal{T}$ in the protocol session i.

I : The length of interactions is $I = [a, b]$. We use the notation $I < J$ to denote I precedes J which means $I = [a, b], J = [c, d]$. and $a \leq b < c \leq d$

$\Omega_I(\mathcal{T})$: Interactions of a tag $\mathcal{T}$ are defined as the execution results of a set of queries on the tag $\mathcal{T}$ where $\Omega_I(\mathcal{T}) = \{\omega_i(\mathcal{T})|i \in I\}$.

l_{ref} : The security parameter controlling the length of interactions that the adversary can execute during the learning phase.

l_{ch} : The security parameter controlling the length of interactions that the adversary can execute during the challenge phase.

Untraceability is formally described using the game $\mathcal{G}$ between an adversary and a set of readers and tags. We describe below the notion of untraceability.

Phase 1 (Learning): $\mathcal{A}$ is able to send any Execute, $Execute^*$, Send, and Corrupt queries to a given tag $\mathcal{T}$ during the chosen time interval $I < l_{ref}$ and gets $\Omega_I(\mathcal{T})$.

Phase 2 (Challenge):

a. Depending on a randomly chosen bit $b \in 0, 1$, $\mathcal{A}$ is given a tag $\mathcal{T}_b$ such that $\mathcal{T}_b = \mathcal{T}$ from a set $\{\mathcal{T}_0, \mathcal{T}_1\}$.

b. $\mathcal{A}$ is able to send any Execute , $Execute^*$, Send and Corrupt queries to the tags $\mathcal{T}_0$ and $\mathcal{T}_1$ during the chosen time intervals $I_0 < l_{ch}$ and $I_1 < l_{ch}$ such that $(I_0 \cup I_1) \cap I = \oslash$ and gets $\Omega_{I_0}(\mathcal{T}_0)$ and $\Omega_{I_1}(\mathcal{T}_1)$

Phase 3 (Guessing): Eventually, $\mathcal{A}$ terminates the game simulation and outputs a bit $b^{'}$ which is its guess of the value of b.

If $\mathcal{A}$ does not use the Corrupt query during the time interval of challenge phase, the advantage of $\mathcal{A}$ in distinguishing whether $\mathcal{A}$ receives $\mathcal{T}_0$ or $\mathcal{T}_1$ is defined by $Adv_P^{UNT}(\mathcal{A}) = 2Pr(b^{'} = b) - 1$ [27]. If $\mathcal{A}$ can use the Corrupt query during the time interval of challenge phase, the untraceability is determined by the choice of experiment time interval. If the choice of time intervals is that $I > I_0$ and $I > I_1$, then the protocol is resistant to backward traceability. In [12], the minimum restriction for forward untraceability is defined such that there should exist some non-empty gap not accessible by $\mathcal{A}$ between the time of Corrupt query and the attack time. If $\mathcal{A}$ chooses the time intervals such

that $I < I_0$ and $I < I_1$ and there are time intervals J_0 and J_1 such that $I < J_0 < I_0$ and $I < J_1 < I_1$ to which $\mathcal{A}$ does not access, then the protocol is resistant to forward traceability.

In some protocols, such as the protocol in [7] described in Chapter 5.1, after corrupting secrets of the tag, $\mathcal{A}$ does not need to access some exchanged messages between protocol parties since these messages do not consist of information required to refresh or update secrets of the tag. If a single communication round of the protocol consists of such messages entirely, $\mathcal{A}$ is not under the necessity of accessing this communication round of the protocol. This analysis reveals a nice design criterion for RFID authentication protocols. This design criterion is that all communication rounds of the protocol must contain necessary information to refresh or update secrets of the tag. Thus, $\mathcal{A}$ have to access all communication rounds of the protocol to trace the tag. Now, we can change the minimum restriction for forward untraceability. That is; there should exist at least a single session which $\mathcal{A}$ does not access at least a single communication round of it between the time of Corrupt query and the attack time. Formally, to provide forward untraceability we only need protocol sessions S_0 and S_1 such that $I < S_0 < I_0$ and $I < S_1 < I_1$ in which $\mathcal{A}$ does not access at least a single communication round of them.

5. ACA: A NEW RFID AUTHENTICATION PROTOCOL

5.1. Detailed Description of Song and Mitchell's Protocol (S-M) and Its Vulnerabilities

5.1.1. Description of the S-M Protocol

The S-M protocol is proposed at WISEC'08 [7]. In the protocol, the following notations are used: $h()$ $\{0,1\}^l \rightarrow \{0,1\}^l$ is a hash function; $f_k()$ $\{0,1\}^l \times \{0,1\}^l \rightarrow \{0,1\}^l$ is a keyed hash function with key k; $x \gg y$ is a right circular shift operator, which rotates all bits of x to the right by y bits; $x \ll y$ is a left circular shift operator, which rotates all bits of x to the left by y bits; l is the length of the parameters in the protocol. The S-M protocol is described in Table 5.1.

Table 5.1. Song and Mitchell's protocol

Server $[(u_i,t_i)_{new},(u_i,t_i)_{old}]$		**Reader**		**Tag** T_i $[t_i]$
		$r_1 \in \{0,1\}^l$	r_1 →	$r_2 \in \{0,1\}^l$
	M_1, M_2 ←		M_1, M_2 ←	$M_1 = t_i \oplus r_2$ $M_2 = f_{t_i}(r_1 \oplus r_2)$
Search for t_i $r_2 = M_1 \oplus t_i$ if $M_2 = f_{t_i}(r_1 \oplus r_2)$, $M_3 = u_i \oplus (r_2 \gg l/2)$	M_3 →		M_3 →	
$u_{i(old)} = u_i$ $t_{i(old)} = t_i$ $u_{i(new)} = (u_i \ll l/4)$ $\oplus(t_i \gg l/4) \oplus r1 \oplus r2$ $t_{i(new)} = h(u_{i(new)})$				$u_i^{'} = M_3 \oplus (r_2 \gg l/2)$ if $h(u_i^{'}) = t_i$ $t_i = h((u_i^{'} \ll l/4)$ $\oplus(t_i \gg l/4) \oplus r1 \oplus r2))$

For each tag T_i, the reader R stores its current identifier and its previous identifier

$(u_i, t_i)_{new}, (u_i, t_i)_{old}$. u_i is an unique secret for T_i and $t_i = h(u_i)$. T_i only stores the value t_i.

5.1.2. Tag Impersonation Attack on the S-M Protocol

Deursen et al. show how an active adversary impersonates a tag to a legitimate server when the adversary does not have any of the tag's secrets [46]. This attack does not result in de-synchronized updating of the secret information between the tag and the server. The adversary can fool the legitimate reader. The attack works as follows:

1. At time a, an adversary $\mathcal{A}$ initiates a session with $\mathcal{T}_i$ with a Send($r_1^{'}$) query. $\mathcal{T}_i$ replies with $M_1^{'}, M_2^{'}$.
2. At time a, the legitimate reader queries $\mathcal{A}$ with r_1.
3. After receiving r_1, $\mathcal{A}$ calls a Send(M_1, M_2) query to the legitimate reader where $M_1 = M_1^{'} \oplus r_1 \oplus r_1^{'}$ and $M_2 = M_2^{'}$. $\mathcal{A}$ does not know the value of t_i and the value of $r_2 = r_2^{'} \oplus r_1 \oplus r_1^{'}$
4. The legitimate reader forwards r_1, M_1, M_2 to the server. The server computes $r_2 = M_1 \oplus t_i$ and checks $M_2 = f_{t_i}(r_1 \oplus r_2)$ for all secrets stored in its database. These equations must hold for $\mathcal{T}_i$ since $M_2 = M_2^{'}$ and $f_{t_i}(r_1 \oplus r_2) = f_{t_i}(r_1 \oplus r_2^{'} \oplus r_1 \oplus r_1^{'})$.

5.1.3. Server Impersonation Attack on the S-M Protocol

Cai et al. present a server impersonation attack on the S-M protocol at WISEC'09. In this attack an active adversary can impersonate a legitimate server to a tag when the adversary does not have any of the tag's secrets. This attack results in de-synchronized updating of the secret information between the tag and the server. The adversary can fool the legitimate reader. The attack works as follows:

1. In the session at time a, an adversary $\mathcal{A}$ calls an Execute query, eavesdrops values

of r_1, M_1, M_2, M_3.

2. Right after interaction between $\mathcal{T}_i$ and $\mathcal{R}$ at time $b = a + 1$, $\mathcal{A}$ initiates a session with $\mathcal{T}_i$ with a Send(r_1) query. $\mathcal{T}_i$ replies with $M_1^{'}, M_2^{'}$. At this point, $\mathcal{A}$ computes $M_3^{'}$ using the following equations:

$$[M_3^{'}]_L = [M_1]_R \oplus [M_3]_L \oplus [M_1^{'}]_R$$
$$[M_3^{'}]_R = [M_1]_L \oplus [M_3]_R \oplus [M_1^{'}]_L$$

3. $\mathcal{A}$ calls a Send(M_3) query. It impersonates the valid server because M_3 is accepted by the tag $\mathcal{T}_i$ successfully.
4. After this session, $\mathcal{A}$ can trace all future interactions of the tag $\mathcal{T}_i$ and the valid readers can not successfully interact with the tag $\mathcal{T}_i$.

Cai et al. also propose their revised protocol that eliminates the vulnerabilities of the S-M protocol. They revise $M_2 = f(r_1 \oplus r_1)$ to $M_2 = f(r_1||r_1)$ and $M_3 = u_i \oplus (r_2 \gg l/2)$ to $M_3 = u_i \oplus h(r_2)$. However, this revised protocol does not resist to the attacks that we will describe in Section 5.1.4 and Section 5.1.5

5.1.4. Violating the Forward Untraceability

In the S-M protocol, the forward untraceability is preserved under an assumption that a strong attacker misses M_3 just once in a single successful authentication session after compromising T_i's secret. That is, if the attacker cannot prevent T_i from receiving the last message M_3, or does not have access to all the values r_1 , r_2 and u_i that are needed to update t_i, then it can not compute the new identifier and track future transactions. Authors claim that the only potential attack to trace future transactions is to access all the values r_1 , r_2 and u_i. Formally, the attack works as follows.

1. **Learning:** An adversary chooses a time interval $I = [a, b]$. Firstly $\mathcal{A}$ issues a

Corrupt query to a tag $\mathcal{T}_0$. During the time interval I, $\mathcal{A}$ calls an Execute query, gets r_1, M_1, M_3 and computes $\mathcal{T}_0$'s new identifier $t_0^{'}$.

2. **Challenge:** $\mathcal{A}$ chooses a time interval $J = [c]$ where $c = b + 1$, sends r_1 to the tag $\mathcal{T}_b \in \{\mathcal{T}_0, \mathcal{T}_1\}$ and obtains its response M_1^*, M_2^*.
3. **Guessing:** A computes $r_2^{'} = M_1^* \oplus t_0^{'}$ and checks if $f_{t_0^{'}}(r_1, r_2^{'}) = M_2^*$. If so, the adversary knows $.\mathcal{T}_b = \mathcal{T}_0$ else knows $\mathcal{T}_b = \mathcal{T}_1$

Now, we show that after compromising T_i 's secret, the attacker does not need to eavesdrop M_3 and access u_i value for each session to track future transactions. Formally, the attack works as follows:

1. **Learning:** An adversary chooses a time interval $I = [a, b]$. An adversary $\mathcal{A}$ issues a Corrupt query to the tag $\mathcal{T}_0$. In the first session a, $\mathcal{A}$ calls an Execute query, eavesdrops r_1, M_1, M_3, gets the values r_1, r_2, u_i and computes the current values of $u_i = (u_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2$, $t_i = h(u_i)$. During the time interval $[a + 1, b]$, $\mathcal{A}$ calls an $Execute^*(1)$ query to eavesdrop r_1 and an $Execute^*(2)$ query to eavesdrop M_1, gets the values r_1, r_2 and computes the current values of $u_i = (u_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2$, $t_i = h(u_i)$.
2. **Challenge:** $\mathcal{A}$ chooses a time interval $J = [c]$ where $c = b + 1$, sends r_1 to the tag $\mathcal{T}_b \in \{\mathcal{T}_0, \mathcal{T}_1\}$ and obtains its response M_1^*, M_2^*.
3. **Guessing:** A computes $r_2^{'} = M_1^* \oplus t_0^{'}$ and checks if $f_{t_0^{'}}(r_1 \oplus r_2^{'}) = M_2^*$. If so, the adversary knows, $\mathcal{T}_b = \mathcal{T}_0$ else knows $\mathcal{T}_b = \mathcal{T}_1$

$\mathcal{A}$ gets M_3 in the session a and computes the new identifier t_0 and the new value of u_0. It does not need M_3 to compute the current value of identifier during the time interval $[a + 1, b]$ because it already knows the current value of u_0 sent in M_3. We can see that $\mathcal{A}$ misses M_3 during the time interval $[a + 1, b]$ (not only in a single session) but the S-M protocol does not preserve the forward untraceability as claimed.

5.1.5. Server Impersonation Attack on the S-M Protocol After Corrupting Tag

An adversary impersonating the valid server can trace the future transactions of the tag without eavesdropping the all future transactions. The attack works as follows:

1. An adversary $\mathcal{A}$ issues a Corrupt query to a tag $\mathcal{T}_i$.
2. In the first session after corrupting, $\mathcal{A}$ calls an Execute query, eavesdrops r_1, M_1, M_3 and gets the values r_1, r_2, u_i.
3. $\mathcal{A}$ computes the current values of $u_i = (u_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2$, $t_i = h(u_i)$ because it knows the previous values of t_i, u_i and the latest values of r_1, r_2.
4. Right after interaction between $\mathcal{T}_i$ and $\mathcal{R}$, $\mathcal{A}$ initiates a session with $\mathcal{T}_i$ with a Send query. Since it knows the current values of t_i, u_i, it impersonates the valid server and passes the check by the tag $\mathcal{T}_i$ successfully.
5. After this session, $\mathcal{A}$ can trace all future interactions of the tag $\mathcal{T}_i$ and the valid readers can not successfully interact with the tag $\mathcal{T}_i$.

We can see that a strong adversary $\mathcal{A}$ compromising the tag can easily impersonate the valid server and the valid server becomes invalid for the tag. This is because the value u_i used for server validation is updated with values t_i, r_1 and r_2 known by $\mathcal{A}$. Thus, $\mathcal{A}$ can behave like the valid server.

5.2. Detailed Description of ACA

5.2.1. Main Idea

The goal of the proposed protocol is to meet all security requirements and improve the efficiency of the tag and the server side operations.

Our protocol uses a classical challenge-response approach. The protocol involves four messages in three rounds. It is designed to provide maximum security with minimum cryptographic operations. The tag holds a secret identifier and a randomly chosen server validator. The server stores an identifier and a seed of the server validator of each tag. Moreover, it stores the old value of the identifier and the seed of the server validator of each tag to prevent DoS attacks with the tag [42, 21, 12, 7].

Our protocol provides forward and backward untraceability by updating secrets of the tag. In an authentication session, a tag creates a temporary secret by using shared random numbers and its identifier. After the server identifies the tag, it accesses the seed of server validator and then sends it combined with the temporary secret to the tag. After a successful authentication, the tag and the server use shared random numbers and the seed of server validator to update the identifier of the tag. Further, the tag stores a random number sent by reader as its new server validator. If a strong adversary compromising a tag and knowing the identifier and the server validator must gain access to all exchanged messages in each communication round to trace the tag in the future because it does not know the value of the seed of the server validator.

The important point is that the server validator of each tag is determined by the server randomly. Unlike the S-M protocol, there is no computational association among old values of the server validator and its new value. Moreover, knowing the current value of the identifier and old values of the identifier does not help to compute the new value of the server validator. As a result, an active adversary compromising a tag and gaining access all messages in each communication round cannot impersonate the valid server. The protocol is summarized in Table 5.4.

5.2.2. Specifications

RFID systems consist of three elements: a tag, a reader and a server. The communication channel between the tag and the reader is insecure and vulnerable to eavesdropping. The communication channel between the reader and the server is secure and authentic. The server communicates only with non-compromised readers.

Table 5.2. Notations in ACA

N	:	The number of tags.
T_i	:	The i-th Tag ($1 \leq i \leq N$).
id_i	:	The secret of T_i.
val_i	:	The server validator of i-th tag.
$seed_i$	:	The seed of val_i.
k	:	The lookup property $1 \leq k \leq 2N$
l	:	The bit-length of tag secrets.
x_{new}	:	The updated value of x.
x_{old}	:	The most recent value of x.

Table 5.3. Functions in ACA

f	:	A pseudorandom function $\{0,1\}^l \times \{0,1\}^{2l} \rightarrow \{0,1\}^{2l}$.
g	:	A hash function $\{0,1\}^l \rightarrow \{0,1\}^l$.
h	:	A hash function $\{0,1\}^l \rightarrow \{0,1\}^l$.
$\oplus$	:	XOR operator.
$\|\|$	:	Concatenation operator.
$\in$	:	The random choice operator.
F	:	A general notation for functions f, g and h.

The tags are resource constraint. However, they store a tag secret id_i serves as

an identifier and a server validator val_i. They are vulnerable to cloning and physical attacks. They can perform hash functions and a pseudorandom function.

The server has a secure database system that stores the identifier and the server validator of each tag. The reader can perform hash operations and generates random numbers.

Table 5.4. ACA authentication protocol

Server $[(id_i)_{new}, (id_i)_{old}, (seed_i)_{new}, (seed_i)_{old}]$		**Reader**		**Tag** T_i $[id_i, val_i]$
		$seed_{r_1} \in \{0,1\}^l$ $r_1 = h(seed_{r_1})$		
			$r_1 \longrightarrow$	
				$r_2 \in \{0,1\}^l$ $p_1, p_2 = f(r_1, r_2 \| id_i)$
	$r_1, r_2, p_1, seed_{r_1} \longleftarrow$		$r_2, p_1 \longleftarrow$	
Search for id_i $p_1^{'}, p_2^{'} = f(r_1, r_2 \| id_i)$ if $p_1^{'} = p_1$, $p_3 = seed_i \oplus p_2^{'}$				
	$p_3 \longrightarrow$		$p_3 \longrightarrow$	
$(id_i)_{old} = id_i$ $(seed_i)_{old} = seed_i$ $(id_i)_{new} = g(r_1 \oplus r_2 \oplus seed_i \oplus id_i)$ $(seed_i)_{new} = seed_{r_1}$				$seed_i^{'} = p_2 \oplus p_3$ if $h(seed_i^{'}) = val_i$ $id_i = g(r_1 \oplus r_2 \oplus seed_i^{'} \oplus id_i)$ $val_i = r_1$

5.2.3. Initialization

For each tag T_i, the back-end server makes the following steps:

1. The server chooses random secrets id_i and $seed_i$ for the tag T_i and computes $val_i = h(seed_i)$.
2. The server creates an entry $[id_i, id_i, seed_i, seed_i]$ for the tag T_i and stores it in its DB. Initially $(id_i)_{new}$ is equal to $(id_i)_{old}$ and $(seed_i)_{new}$ is equal to $(seed_i)_{old}$.
3. The tag T_i stores an identifier id_i and a server validator val_i.

5.2.4. Authentication

1. The reader generates a random bit string $seed_{r_1} \in \{0,1\}^l$ as a seed of the server validator and computes $r_1 = h(seed_{r_1})$
2. The reader sends a request message r_1 to the tag T_i.
3. Upon receiving r_1, T_i generates a random bit string $r_2 \in \{0,1\}^l$, computes $p_1, p_2 = f(r_1, r_2||id_i)$
4. T_i sends (r_2, p_1) to the reader.
5. Upon receiving (r_2, p_1) the reader queries the server with $(r_1, r_2, p_1, seed_{r_1})$.
6. The server searches its DB using (r_1, r_2, p_1).
   (a) The server picks id_i from between $(id_i)_{new}$ and $(id_i)_{old}$ and picks $seed_i$ from between $(seed_i)_{new}$ and $(seed_i)_{old}$ and computes $p_1^{'}, p_2^{'} = f(r_1, r_2||id_i)$ for each entry in its DB until it finds the response of the tag T_i.
   (b) If the server finds the response of the tag T_i, the server identifies and authenticates the tag, else stops the session.
   (c) The server computes $p_3 = seed_i \oplus p_2^{'}$.
   (d) The server sends p_3 to the reader.
   (e) The server updates values in the related entry as follows:

$$(id_i)_{old} = id_i$$
$$(id_i)_{new} = g(r_1 \oplus r_2 \oplus seed_i \oplus id_i)$$
$$(seed_i)_{old} = seed_i$$
$$(seed_i)_{new} = seed_{r_1}$$

7. The reader forwards p_3 to the tag T_i.
8. T_i computes $seed_i^{'} = p_3 \oplus p_2$.
9. If $h(seed_i^{'}) = val_i$, the tag authenticates the server.
10. The tag updates its secrets as follows:

$$id_i = g(r_1 \oplus r_2 \oplus seed_i^{'} \oplus id_i)$$
$$val_i = r_1$$

5.3. Security Analysis of ACA

In this section, we present how our protocol resists security and privacy attacks.

5.3.1. Privacy Attacks

In privacy attacks, a passive adversary wants to learn the content of a tag T_i. Then it queries tags. In each session, the tag uses a pseudorandom function f to create a message p_1 and responds to the reader with (r_2, p_1). We know that output of the pseudorandom function can be seen as a random bit string. Only valid server can access the information associated with the tag and so can extract the correct id_i from the message (r_2, p_1). An adversary knowing r_1, r_2 and p_1 can not obtain id_i. Thus, our protocol provides information privacy for the tag.

In another form of privacy attacks, a passive adversary tries to distinguish a tag T_i from other RFID tags over time to track T_i. In our protocol, the tag creates an anonymous message p_1 for each challenge. The adversary tries to track the tag by using the same r_1, but p_1 is a function of freshly generated nonce r_2, so it can not link this message to any particular tag. Moreover, it can not distinguish one tag's response from another's. Thus, our protocol provides location privacy for the tag. Formally the attack works as follows:

1. **Learning:** $\mathcal{A}$ chooses a time interval I, sends any $Execute, Execute^*$ and $Send$ queries to a tag T_0 during the time interval I and gets $\Omega_I(T_0)$.
2. **Challenge:** A tag T_b is given to $\mathcal{A}$ from a set $\{T_0, T_1\}$. $\mathcal{A}$ sends a $Send(r_1)$ query to the tag T_b and obtains its response r_2, p_1
3. **Guessing:** $\mathcal{A}$ outputs $b^{'}$ as its guess of the value of b.

The probability of $(b^{'} = b)$ is $\frac{1}{2}$ since messages sent by the tag and the reader give no useful information to $\mathcal{A}$ to determine which of the given tags is T_0.

$$Adv_P^{UNT}(\mathcal{A}) = 2Pr(b^{'} = b) - 1 = 0 \tag{5.1}$$

5.3.2. Replay Attack

An adversary can not use messages from previous sessions because in every authentication session, all messages on communication rounds are created using nonces r_1 and r_2. This condition is sufficient to protect our protocol against replay attacks.

5.3.3. Denial of Service Attack

Our protocol needs synchronization between the server and the tag. The tag updates its secrets after taking a confirmation message from the server. An adversary can prevent the reader or the tag from receiving a message. If the adversary performs this attack on the third flow of our protocol, it can prevent the tag from taking the confirmation message. This breaks the synchronization between the tag and the server because the server updates secrets of the tag but the tag does not. However, in our protocol, the server makes itself synchronize with the tag in such a situation because it stores old and new values of secrets of the tag.

Table 5.5. RFID protocols and ACA: security and privacy features

Security and Privacy Threat	[21]	[24]	[23]	[42]	[12]	[7]	[10]	ACA
Information Privacy	○	○	○	○	○	○	○	○
Location Privacy	−	○	−	○	−	○	○	○
Tag Impersonation	−	○	−	○	○	−	○	○
Replay Attack	−	○	−	○	○	−	○	○
DoS Attack	○	○	−	○	○	○	○	○
Backward Traceability	−	−	○	−	○	○	○	○
Forward Traceability	−	−	−	−	△	+	+	△
Server Impersonation	−	−	−	−	○	−	△	○

○ : provided

△ : provided under the assumption in [12, 7]

\+ : partially provided under the assumption in [12, 7]

− : not provided

5.3.4. Cloning Attack

In cloning attack, an adversary queries a tag to obtain secrets of the tag. It then places obtained secrets into a fake tag. In our protocol, the tag responds to the reader query using freshly generated nonces r_1 and r_2. It is hard to search correct secrets using r_1, r_2 and p_1 for the adversary. Thus, our protocol has resistance to cloning attack and tag impersonation attack.

5.3.5. Backward Traceability

A strong adversary knowing the present internal state of a tag tries to identify the past interactions of it. Our protocol prevents the strong adversary from identifying past interactions of the tag by updating secrets of it after each successful authentication

session. Thus, our protocol provides backward untraceability. Formally the game works as follows:

1. **Learning:** $\mathcal{A}$ chooses a time interval I, sends any $Execute, Execute^*, Send$ and $Corrupt$ queries to a tag T_0 during the time interval I and gets $\Omega_I(T_0)$.
2. **Challenge:** A tag T_b is given to $\mathcal{A}$ from a set $\{T_0, T_1\}$. $\mathcal{A}$ chooses a time interval I_b such that $I_b < I$, sends any $Execute, Execute^*, Send$ and $Corrupt$ queries to the tag T_b and gets $\Omega_{I_b}(T_b)$.
3. **Guessing:** $\mathcal{A}$ outputs $b^{'}$ as its guess of the value of b.

The probability of $(b^{'} = b)$ is $\frac{1}{2}$. During the time interval of I, $\mathcal{A}$ gets the current secrets of the tag T_0. It can not identify the past interactions of the tag T_0 because secrets of the tag are updated using one way hash function after each successful session.

$$Adv_P^{UNT}(\mathcal{A}) = 2Pr(b^{'} = b) - 1 = 0 \tag{5.2}$$

5.3.6. Forward Traceability

A strong adversary knowing the present internal state of a tag can track the future identifications of it, if it can compute updated values of secrets of the tag. Hence, it must access all messages needed to update secrets of the tag. Our protocol provides forward untraceability under an assumption that a strong adversary misses at least one of the values of r_1, r_2 and p_3 just once in a single successful authentication session after compromising the tag. In our protocol, the adversary must catch the last message p_3 from the reader to the tag for each session after compromising the tag to track future transactions, since it does not know and can not compute the value of $seed_i$. The adversary must monitor all communication flows to update secrets of the tag. Formally the game works as follows:

1. **Learning:** $\mathcal{A}$ chooses a time interval I, sends any $Execute, Execute^*, Send$ and $Corrupt$ queries to a tag T_0 during the time interval I and gets $\Omega_I(T_0)$.
2. **Challenge:** A tag T_b is given to $\mathcal{A}$ from a set $\{T_0, T_1\}$. $\mathcal{A}$ chooses a time interval I_b such that $I_b > I$, sends any $Execute, Execute^*, Send$ and $Corrupt$ queries to the tag T_b and gets $\Omega_{I_b}(T_b)$.
3. **Guessing:** $\mathcal{A}$ outputs $b^{'}$ as its guess of the value of b.

If there is at least a single session S_i between I and I_b where $\mathcal{A}$ misses at least one of messages r_1, r_2 and p_3, the probability of $(b^{'} = b)$ is $\frac{1}{2}$. During the time interval of I, $\mathcal{A}$ gets the current secrets of the tag T_0. It can not compute the current values of secrets of the tag after the session S_i because it misses at least a single value required to update secrets of the tag. Under such an assumption, the advantage of $\mathcal{A}$ is that:

$$Adv_P^{UNT}(\mathcal{A}) = 2Pr(b^{'} = b) - 1 = 0 \tag{5.3}$$

5.3.7. Server Impersonation Attack

The server impersonation is a novel attack introduced in [7]. The S-M protocol provides resistance to server impersonation under an assumption. Our protocol uses server validator val_i to resist server impersonation. Hence, the tag can validate the server. In previous protocols, a strong adversary knowing the current internal state of a tag T_i can impersonate the valid server, if it can access all messages needed to compute secrets of the tag from communication flows because the identifier serves as the server validator. In our protocol, the server validator val_i is determined by the valid reader randomly and independent from the identifier id_i. The seed of the server validator $seed_i$ is stored on the valid server. The strong adversary can access the current value of the $seed_i$ from message p_3, but it can not compute or predict the new value of $seed_i$ to impersonate the valid server. Thus, our protocol provides resistance to server

impersonation without any assumption.

Formally, if we apply the same attack given in Section 5.1.5, $\mathcal{A}$ can compute the current values of the identifier id_i and the server validator val_i in Step 3. It can not impersonate the valid server and can not pass check by the tag in Step 4, because it does not know the current value of $seed_i$. The current value of the $seed_i$ is determined by the valid reader and stored on the valid server.

We compare security and privacy properties of our protocol with those of previously proposed protocols in Table 5.5. It is clear that our protocol provides the greatest number of security and privacy properties. Furthermore, our protocol provides forward untraceability under the assumption in [12, 7] and resists server impersonation without any assumption.

5.4. Performance Analysis of ACA

The proposed protocol is efficient; the computational cost of the tag is computation of two hash functions and a pseudorandom function.

The comparison of storage requirements is given in Table 5.6. The S-M protocol has the least storage requirements. In the server side, our protocol needs the same amount of space with the S-M protocol. In the tag side, our protocol needs two times more storage than the S-M protocol. This difference is acceptable. For example, if we choose the length of the secret as 128 bits, we need 256 bits storage capacity in tag side. This amount of storage capacity is already available in low cost RFID tags.

The comparison of computational requirements is given in Table 5.7. The S-M protocol has the least computational requirements. In the server and the tag side, our protocol does the same computations with the S-M protocol under an assumption

Table 5.6. RFID protocols and ACA: storage requirements

Storage	[42]	[12]	[7]	ACA
Server	$(l + 2l_1 + 2l_2) \cdot N$	$2(l + 3l_3 + ml_4) \cdot N$	$4l \cdot N$	$4l \cdot N$
Tag	$l + l_1 + l_2$	$l + l_3$	l	$2l$

m : The maximum number of authentication failures in [12] (e.g., $m = 64$ [12])

l : The bit length of a tag identifier (e.g., $l = 64, 96$ or 128 [42, 12])

l_1 : The bit length of a PIN in [42] (e.g., $l_1 = 32$ [42])

l_2 : The bit length of a session key in [42] (e.g., $l_2 = 16$ [42])

l_3 : The bit length of a server validator in [12] (e.g., $l_3 = 32$ [12])

l_4 : The bit length of a tag secret transmitted in [12] (e.g., $l_4 = 32$ [12])

Table 5.7. RFID protocols and ACA: computational requirements

Computation		[42]	[12]	[7]	ACA
	2nd flow	kF	$(k_1 + 1)F$	kF	kF
Server	3rd flow	$1F$	$1F$		
	Updating	$2F$	$(k_1 + k_2 + m)F$	$1F$	$1F$
	Total	$(k + 3)F$	$(2k_1 + k_2 + m + 2)F$	$(k + 1)F$	$(k + 1)F$
Reader	1st flow				$1F$
	2nd flow	$1F$	$1F$	$1F$	$1F$
Tag	3rd flow	$1F$	$2F$	$1F$	$1F$
	Updating	$2F$	$1F$	$1F$	$1F$
	Total	$4F$	$4F$	$3F$	$3F$

F : A computationally complex function (such as a CRC, PRF or hash function)

n : The length of the backward key chain in [12] (e.g., $n = 2^{20}$ [12])

k : An integer satisfying $1 \leq k \leq 2N$ (look-up property)

k_1 : An integer satisfying $1 \leq k_1 \leq m - 1$

k_2 : An integer satisfying $1 \leq k_2 \leq n - 2$

that the compared protocols use the same cryptographic algorithms. Furthermore, our protocol has practical performance advantages over the protocol in [12]. In [12], computing the new value of server validator requires too many hash operations. For example, the length of the backward hash chain is 2^{20} so the server must compute $(2^{20} - m)$ hash operations in the successful session m. Our protocol needs only a single hash computation to compute the new value of the server validator. Our protocol needs a single hash computation in the reader to provide enhanced security. Because of the high computational capabilities of the RFID reader, this difference is acceptable.

As a result, we can say that our protocol is the first one that provides the greatest number of security features with minimum storage and computational requirements.

6. ACAT: A NEW RFID AUTHENTICATION PROTOCOL BASED ON TREE STRUCTURE

6.1. Detailed Description of Former Proposal (SAPA)

In [8], protocol based on sparse tree structure is proposed due to the tight relationships between the adjoining tags in the tree of [1]. In this protocol, each tag stores three secrets: a root key K_H that is shared by all tags, a path key K_m that is unique to each tag and a leaf key K_m that is unique to each tag. These secrets are updated after each successful authentication. For example, as shown in Figure 6.1, there are three tags T_1, T_2 and T_3. The reader stores their keys in a sparse tree.

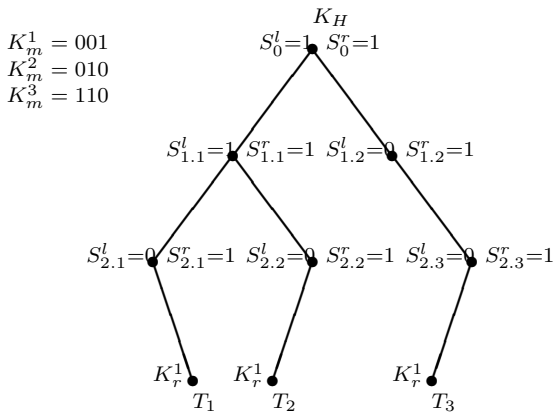


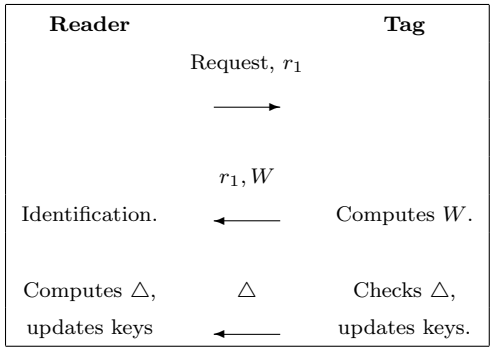
Figure 6.1. The sparse tree of SAPA

In identification phase shown in Table 6.1, a reader queries a tag with a random nonce r_1. After receiving r_1, the tag picks a random nonce r_2, computes a hash chain

$$
\begin{aligned}
W = \quad & h(K_H, r_1, r_2), h(h(K_H, r_1, r_2), K_m^i[1]), \\
& h(h(h(K_H, r_1, r_2), K_m^i[1]), K_m^i[2]), \\
& h(h(h(...h(K_H, r_1, r_2), K_m^i[1]), K_m^i[2]), K_r^i)
\end{aligned}
$$

and sends r_2, W to the reader. The reader searches the tag in the key tree by using the hash chain W. This algorithm is efficient, because the complexity of tree-based key search is logarithmic. If the reader identifies T_1, it removes a branch corresponding to K_m^i, updates K_m^i and K_r^i, inserts the branch corresponding to K_m^i, and then sends a synchronization message $\triangle = h(K_m^i, K_r^i, r_2, r_1, 3)$ to the tag. After checking $\triangle$, the tag updates K_m^i and K_r^i.

Table 6.1. The identification of SAPA



6.1.1. Weaknesses of SAPA

SAPA provides backward untraceability and low storage cost for the tags with high authentication efficiency. However, it is vulnerable to location tracking. SAPA needs synchronization between the tag and the server. However, the adversary can break the synchronization by preventing the tag from taking the synchronization message. On the other hand, in SAPA the communication and computation complexity for the tag is high because the tag computes a long hash-chain and sends it to the reader. For example, The bit length of K_m is 20 and each key is 64 bit, as a result

each tag must perform $(20 + 2)$ hash operations and sends $(22 + 2 + 1) \times 64 = 1476$ bits information to the reader.

6.1.1.1. Location Tracking. A passive adversary can trace the location of a tag between two successful authentication sessions. It can trace the location by only querying the tag. In SAPA, each tag has a unique path key and the adversary can easily extract the tag path key from the hash-chain and identify the tag. For example, the adversary sends a challenge r_1 to T_1 and then receives a response r_2 and

$$\begin{aligned} W = \quad & h(K_H, r_1, r_2), h(h(K_H, r_1, r_2), K_m^1[0]), \\ & h(h(h(K_H, r_1, r_2), K_m^1[0]), K_m^1[1]), \\ & h(h(h(h(K_H, r_1, r_2), K_m^1[0]),\ K_m^1[1]), K_m^1[2]). \\ & h(h(h(h(h(K_H, r_1, r_2), K_m^1[0]),\ K_m^1[1]), K_m^1[2]), K_r^1) \end{aligned}$$

He extracts the $K_m^1[0]$ from the second hash value by using the first hash value. It is easy because $K_m^1[0]$ provides one bit information. It extracts all bits of K_m^1 by using successive hash values. If the branching factor α is greater than 2, the adversary must compute $\alpha - 1$ hash value to extract $\log(\alpha)$ bits of the path. However, the branching factor should not be too high not to increase the search complexity. The branching factor $\alpha = 16$ is suggested in [8]. In above example, we show that a passive adversary that can not corrupt tags can easily trace a specific tag.

6.1.1.2. Denial of Service Attack. This protocol needs synchronization of keys between the tag and the server. However, SAPA is vulnerable to denial of service attack. The passive adversary can break the synchronization of keys by preventing the tag from taking the synchronization message $\triangle$ or invalidating it.

6.1.1.3. Server Impersonation Attack. A strong adversary compromising the tag secret can impersonate the valid server. Firstly, the adversary compromises a specific tag and then it sends a request to this tag. After receiving a response from the tag, the adversary can create a valid synchronization message, because it knows secrets of the tag. This attack can result in de-synchronized updating of keys between the tag and the server. Since the adversary can impersonate the valid server, the tag updates its keys and becomes unreachable from any valid readers. The adversary takes the ownership of the tag.

6.1.1.4. Forward Traceability. The only way to avoid forward tracing is that all communication round of the protocol must contain a necessary information to update or refresh secrets of the tag. In SAPA, a strong adversary compromising a tag can trace the future interactions of it. The adversary only needs to eavesdrop r_1 from the first round and r_2 from the second round. It does not need to monitor the third round of the protocol. This is the weak point of SAPA.

6.2. Detailed Description of ACAT

The goal of the proposed protocol is to meet all security requirements, improve the efficiency of the tag and the server side operations, and reduce the communication complexity.

In this protocol, we use a balanced binary tree with predefined depth like [1, 24]. Each node except the root is represented with 0 if it is the left child of its parent or 1 if it is the right child of its parent. Different from former proposed protocols, in addition to balanced tree, we use sparse tree structure on the leaf nodes of the balanced tree. For example in Figure 6.2, the first leaf node of the balanced tree has three children and the second leaf node of the balanced tree has a child. Each path from the root to

the leaf of the balanced tree is shared by the corresponding tags. Secrets of tags are stored on the leaf nodes of whole tree. After each successful authentication session, the path of the tag is updated by using a cryptographic hash function and a new leaf assigned to the tag is inserted to the updated path. Moreover, the leaf node that stores the most recent state of the tag is kept to prevent DoS attacks.

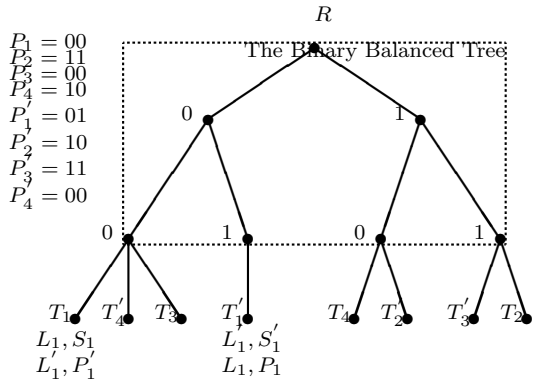


Figure 6.2. The tree structure of ACAT

Our protocol provides forward and backward untraceability by updating secrets of the tag. In an authentication session, a tag creates a temporary secret by using shared random numbers and its identifier. After the server identifies the tag, it accesses the seed of server validator and then sends the seed of the server validator combined with the temporary secret to the tag. After a successful authentication, the tag and the server use shared random numbers and the seed of server validator to update the identifier of the tag. Further, the tag stores a random number sent by reader as its new server validator. If a strong adversary compromising a tag and knowing the identifier and the server validator must gain access to all exchanged messages in each communication round to trace the tag in the future because it does not know the value of the seed of the server validator.

The important point is that the server validator of each tag is determined by the server randomly. Unlike the protocol in [7], there is no computational association

among old values of the server validator and its new value. Moreover, knowing the current value of the identifier and old values of the identifier does not help to compute the new value of the server validator. As a result, an active adversary compromising a tag and gaining access all messages in each communication round cannot impersonate the valid server.

6.2.1. Specifications

RFID systems consist of three elements: a tag, a reader and a server. The communication channel between the tag and the reader is insecure and vulnerable to eavesdropping. The communication channel between the reader and the server is secure and authentic. The server communicates only with the non-compromised readers.

The tags are resource constraint, however they store secrets R, P_i, L_i and the server validator V_i. They are vulnerable to cloning and physical attacks. They can perform hash functions and generate random numbers.

The server has secure system to store secrets of the tag. The reader can perform a hash function and generate random numbers.

6.2.2. Initialization

The server first builds a root for the tree and assigns randomly chosen R as key to it. According to the number of tags N, the server determine the depth of the binary balanced tree d and insert 2^d branches to the tree. For example, $N = 2^{20}$ the server choose $d = 15$. This means that on the average, one leaf node of the balanced tree has $2 \times (2^{20}/2^{15}) = 2^6$ children. For each tag T_i, backend server makes the following steps:

Table 6.2. Notations in ACAT

N	The number of tags.
T_i	The i-th Tag ($1 \leq i \leq N$).
T_i'	The most recent state of i-th Tag ($1 \leq i \leq N$).
R	The root secret.
P	The path secret.
L	The leaf secret.
S	The seed value.
P_i	The path of i-th tag (the first d bits is used as path).
L_i	The secret of i-th tag.
V_i	The server validator of i-th tag.
S_i	The seed value of V_i.
l	The bit-length of R, P_i, L_i, V_i, S_i.
x_{other}	The updated or most recent value of x.

Table 6.3. Functions in ACAT

h	A hash function $\{0,1\} \rightarrow \{0,1\}^l$.
g	A hash function $\{0,1\} \rightarrow \{0,1\}^{2l}$.
$\oplus$	XOR operator.
$\gg$	Right shift operator.
$\in$	The random choice operator.

1. The server chooses random secret R as a root key.
2. The server chooses random secret L_i and P_i for the tag T_i.
3. The server chooses random S_i for the tag T_i and computes $V_i = h(S_i)$.
4. The tag T_i stores tag secrets R, P_i, L_i and V_i.
5. The server creates entry $[L_i, NULL, S_i, NULL]$ for the tag T_i and assigns it to the leaf node inserted to the node of balanced tree pointed by P_i.

6.2.3. Authentication

The protocol is summarized in Table 6.4.

1. Reader: Generates random bit string $S_{r_1} \in \{0,1\}^l$ as the seed of the server validator and computes $r_1 = h(S_{r_1})$ sends r_1 to the tag T_i.
2. Tag: T_i generates random bit string $r_2 \in \{0,1\}^l$, computes $M_1 = h(r_1, r_2, R, 1)$, $tmp = h(r_2, r_1, R, 2)$, $M_2 = tmp \oplus P_i$, and $M_3, M_4 = g(r_1, r_2, P_i, L_i, 1)$ and sends (r_2, M_1, M_2, M_3) to the reader.
3. Reader: Sends $(r_1, r_2, M_1, M_2, M_3, S_{r_1})$ to the server.
4. Server: Computes $M_1^{'} = h(r_1, r_2, R, 1)$, if $M_1^{'} = M_1$, identifies the root key. If not stops the session.
5. Server: Computes $tmp^{'} = h(r_2, r_1, R, 2)$, $P_i^{'} = tmp^{'} \oplus M_2$.
6. Server: Goes the node pointed by $P_i^{'}$, searches leaves of the node by computing $M_3^{'}, M_4^{'} = g(r_1, r_2, P_i^{'}, L_i^{'}, 1)$, if $M_3^{'} = M_3$, identifies the tag. If no match found, then stops the session.
7. Server: Computes $M_5 = S_i \oplus M_4^{'}$, then sends M_5 to the reader.
8. Server: If $(L_i)_{other} \neq$ NULL, erases the leaf corresponding to $(L_i)_{other}$ from the node corresponding to $(P_i)_{other}$.
9. Server: Computes new keys for T_i. $L_{other} = L_i$, $P_{other} = P_i$, $L = h(L_i, S_i, r_1, r_2, 3)$, $P = h(S_i, L_i, r_1, r_2, 2)$, $S = S_{r_1}$.
10. Server: Makes changes on the current leaf that is assigned to T_i. Sets $(L_i)_{other} = L$ and $(P_i)_{other} = P$.
11. Server: Inserts a new leaf storing $[L, L_{other}, S, P_{other}]$ to the node corresponding to P for T_i.
12. Reader: Forwards M_5 to the tag T_i.
13. Tag: Computes $S_i^{'} = M_5 \oplus M_4$, if $h(S_i^{'}) = V_i$, authenticates the server, then updates P_i as $P_i = h(S_i^{'}, L_i, r_1, r_2, 2)$ and L_i as $L_i = h(L_i, S_i^{'}, r_1, r_2, 3)$ and sets $V_i = r_1$.

Table 6.4. ACAT authentication protocol

Server	**Reader**	**Tag** T_i
on each leaf		$[R, P_i, L_i, V_i]$
$[L_i, (L_i)_{other}, S_i, (P_i)_{other}]$		
	$S_{r_1} \in \{0,1\}^l$	
	$r_1 = h(S_{r_1})$	
	r_1 ⟶	
		$r_2 \in \{0,1\}^l$
		$M_1 = h(r_1, r_2, R, 1)$
		$tmp = h(r_2, r_1, R, 2)$
		$M_2 = tmp \oplus P_i$
		$M_3, M_4 = g(r_1, r_2, P_i, L_i, 1)$
	⟵ $r_1, r_2, M_1, M_2, M_3, S_{r_1}$	⟵ r_2, M_1, M_2, M_3
if $M_1 = h(r_1, r_2, R, 1)$		
$tmp^{'} = h(r_2, r_1, R, 2)$		
$P_i^{'} = M_2 \oplus tmp^{'}$		
goes the node pointed by the path $P_i^{'}$		
tries each leaf for $L_i^{'}$		
if $M_3, M_4^{'} = g(r_1, r_2, P_i^{'}, L_i^{'}, 1)$		
$M_5 = S_i^{'} \oplus M_4^{'}$		
	M_5 ⟶	M_5 ⟶
erases the leaf $(L_i)_{other}$		$S_i^{'} = M_5 \oplus M_4$
from the node $(P_i)_{other}$		if $h(S_i^{'}) = V_i$
makes computation for a new leaf		$P_i = h(S_i^{'}, L_i, r_1, r_2, 2)$
$P_{other} = P_i$		$L_i = h(L_i, S_i^{'}, r_1, r_2, 3)$
$P = h(S_i, L_i, r_1, r_2, 2)$		$V_i = r_1$
$L_{other} = L_i$		
$L = h(L_i, S_i, r_1, r_2, 3)$		
$S = S_{r_1}$		
makes changes on the current leaf		
$(L_i)_{other} = L$, $(P_i)_{other} = P$		
inserts a leaf storing		
$[L, L_{other}, S, P_{other}]$ to the node P		

6.3. Security Analysis of ACAT

In this section, we present how our protocol resists security and privacy attacks.

6.3.1. Information Privacy

In privacy attacks, a passive adversary wants to learn the contents of a tag T_i. It then queries the tag. In each session, the tag uses hash function h to create message M_1, M_2, M_3 and responds the reader with (r_2, M_1, M_2, M_3). We know that the output of the hash function can be seen as a random bit string. Only valid server can access the information associated with the tag so only valid server can extract the correct R, P_i, L_i from the message (r_2, M_1, M_2, M_3). An adversary knowing r_1, r_2, M_1, M_2 and M_3 can not obtain R, P_i, L_i. Thus, our protocol provides information privacy for the tag.

6.3.2. Location Privacy

In another form of the privacy attacks, a passive adversary tries to distinguish a tag T_i from other RFID tags over time to track T_i. In our protocol, the tag creates anonymous M_1, M_2, M_3 values for each challenge. The adversary tries to track the tag by using same r_1 but M_1, M_2 and M_3 are functions of freshly generated nonce r_2 so it can not link these messages to any particular tag. Moreover, it can not distinguish one tag's response from another's. Thus, our protocol provides location privacy for the tag.

6.3.3. Location Privacy After Compromising One Tag

A strong adversary compromising a tag can extract the path key P_i of all tags because it knows the root key R shared by all tags. This does not reduce the security

level of our protocol and the adversary can not trace uncompromised tags because we update the key $P_i = h(S_i, L_i, r_1, r_2, 2)$ and the leaf key $L_i = h(L_i, S_i, r_1, r_2, 3)$ after each successful authentication. The adversary can not compute the updated path key P_i since it does not know the leaf key L_i. Furthermore, the adversary can not trace a specific tag T_i between two successful authentications because the key P_i is not unique to the tag T_i and shared with other tags. It can not determine whether it queries the tag T_j or not. The only way to trace uncompromised tags is to extract the leaf key L_i that is unique to each tag. It is infeasible to extract the leaf key L_i from the value M_3 because of the one-way property of hash functions. As a result, the adversary can not distinguish one tag's response without knowing the leaf key L_i.

6.3.4. Replay Attack

An adversary can not use messages from previous sessions because in every authentication session, all messages on communication rounds are created using nonce r_1 and r_2. This condition is sufficient to protect our protocol against replay attack.

6.3.5. Denial of Service Attack

Our protocol needs synchronization between the server and the tag. The tag updates its secrets after taking confirmation from the server. An adversary can prevent the reader or the tag from receiving a message. If the adversary perform this attack on the third flow of our protocol, it can prevents the tag T_i from taking confirmation message M_5. This breaks the synchronization between the tag and the server because the server updates secrets of the tag but the tag does not. However, in our protocol, the server makes itself synchronize with the tag by keeping both the new and the most recent leaf assigned to the tag.

Table 6.5. RFID protocols and ACAT: security and privacy features

Security and Privacy Threat	[1]	[8]	Our Protocol
Information Privacy	○	–	○
Location Privacy	○	–	○
Tag Impersonation	○	○	○
Replay Attack	○	○	○
DoS Attack	–	–	○
Backward Traceability	–	○	○
Forward Traceability	–	–	△
Server Impersonation	–	–	○

○ : provided

△ : provided under the assumption

– : not provided

6.3.6. Cloning Attack

In cloning attack, an adversary queries a tag T_i to obtain secrets of the tag. It then places obtained secrets into the fake tag. In our protocol, the tag responds to the reader query using freshly generated nonce r_1 and r_2. It is hard to search correct keys using r_1, r_2, M_1, M_2 and M_3 for the adversary. Thus, our protocol has resistance to cloning attack and tag impersonation attack.

6.3.7. Backward Traceability

A strong adversary knowing the present internal state of a tag tries to identify the past interactions of it. Our protocol prevents the strong adversary from identifying the past interactions of the tag by updating secrets of the tag after each successful authen-

tication session. Thus, our protocol provides backward untraceability. Compromising secrets of the tag does not lead to compromise of secrets of other tag because our tree is not static. The path and leaf secret of each tag are updated after each successful authentication. Moreover, the strong adversary does not predict the depth of the tree from exchanged messages and compromised secrets.

6.3.8. Forward Traceability

A strong adversary knowing the present internal state of a tag can track the future identifications of it if it can compute updated value of secrets of the tag. Hence, it must access all messages needed to update secrets of the tag. Our protocol provides forward untraceability under the assumption that if a strong adversary misses at least one of the values of r_1, r_2 and M_5 just once in a single successful authentication session after compromising the tag. In our protocol, the adversary must catch the last message M_5 from the reader to the tag for every session after compromising the tag to track future transactions, since it does not know and can not compute the value of S_i. The adversary must monitor all communication flows to update secrets of the tag.

6.3.9. Server Impersonation Attack

The server impersonation is a novel type attack introduced in [7]. The protocol in [7] provides resistance to server impersonation under an assumption. Our protocol uses server validator V_i to resist server impersonation. Hence, the tag can validate the server. In previous schemes, a strong adversary knowing the current internal state of the tag T_i can impersonate the valid server if it can access all messages needed to compute secrets of the tag from communication flows because the tag identifier serves as the server validator. In our scheme, the server validator V_i is determined by the valid reader randomly and independent from secrets of the tag. The seed of the server validator S_i is stored on the valid server. The strong adversary can access the current

value of the S_i from message M_5, but it can not compute or predict new value of S_i to impersonate the valid server. Thus, our protocol provides resistance to server impersonation without any assumption.

We compare security and privacy properties of our protocol with those of previously proposed protocols in Table 6.5. It is clear that our protocol provides the greatest number of security and privacy properties.

6.4. Performance Analysis of ACAT

The proposed protocol is efficient. Computational cost of the tag is six hash functions.

The comparison of storage requirements is given in Table 6.6. The previously proposed protocol [8] has the least storage requirements. In the tag side, our protocol needs nearly same amount of space with the protocol in [8]. In the server side our protocol needs eight times more storage than [8]. This difference is ignorable. We know that the server and the reader in RFID systems are powerful devices. The server can easily manage this amount of storage.

Table 6.6. RFID protocols and ACAT: storage requirements

Storage	[1]	[8]	Our Protocol
Server	$(2N-1)\cdot l_1$	$(N+1)\cdot l_1$	$(8N+1)\cdot l_1$
Tag	$d_1 \cdot l_1$	$2l_1 + l_2$	$4l_1$

N : The number of tags

l_1 : The bit length of a key in [1, 8] (e.g., $l_1 = 64$)

l_2 : The bit length of a K_m path key in [8] (e.g., $l_1 = 20$ [8])

d_1 : The depth of the tree in [1] (e.g., $d_1 = 20, N = 2^{20}$ [1])

The comparison of computational requirements is given in Table 6.7. Our protocol has the least computational requirements on the tag side under an assumption that if the compared protocols use same cryptographic algorithms. Our computational cost on the tag side is fixed to six hash computation. In previously proposed protocols, computational cost is changing according to the depth of the tree. Computational and communication complexities of the server are nearly same as the scheme of [8]. Our protocol makes one hash computation in the reader to provide enhanced security. Because of the high computational capabilities of the RFID reader, this difference is acceptable.

Table 6.7. RFID protocols and ACAT: computational requirements

Computation		[1]	[8]	Our Protocol
Server	2nd flow	$(d_1+1)F$	$(l_2+2)F$	$(d_4+2)F$
	3rd flow	$1F$	$1F$	
	Updating	d_2F	$2F$	$2F$
	Total	$(d_1+d_2+2)F$	$(l_2+5)F$	$(d_4+4)F$
Reader	1st flow			$1F$
Tag	2nd flow	$(d_1+1)F$	$(l_2+2)F$	$3F$
	3rd flow	$1F$	$1F$	$1F$
	Updating	d_2F	$2F$	$2F$
	Total	$(d_1+d_2+2)F$	$(l_2+5)F$	$6F$

F : A computationally complex function (CRC, PRF or hash function)

N : The number of tags

l_2 : The bit length of a K_m in [8] (e.g., $l_2 = 20$ [8])

d_1 : The depth of the tree in [1] (e.g., $d_1 = 20, N = 2^{20}$ [1])

d_2 : An integer satisfying $1 \leq d_2 \leq d_1$ in [1]

d_3 : The depth of the binary balanced tree in our protocol

d_4 : An integer $\approx 2N \div 2^{d_3}$

The comparison of communication requirements is given in Table 6.8. It is clear that our protocol has the least communication requirements on the tag side.

Table 6.8. RFID protocols and ACAT: communication requirements

Communication	[1]	[8]	Our Protocol
Server	l_1	l_1	l_1
Tag	$(d_1 + 1) \cdot l_1$	$(l_2 + 2) \cdot l_1$	$4l_1$

l_1 : The bit length of a key in [1, 8] (e.g., $l_1 = 64$)

l_2 : The bit length of a K_m in [8] (e.g., $l_2 = 20$ [8])

d_1 : The depth of the tree in [1] (e.g., $d_1 = 20, N = 2^{20}$ [1])

As a result we can say that our protocol is the first one that provides the greatest number of security features with minimum storage, computation and communication requirements on the tag side.

7. CONCLUSIONS

In this thesis, security and privacy aspects of Radio Frequency Identification (RFID) is studied. In particular, we focus on traceability problems and server impersonation attacks. We give a brief overview of proposed authentication protocols and defined security and privacy problems in RFID.

In Chapter 4, a new RFID privacy model (ACAP) that defines abilities of the adversary under some restrictions is proposed. We put some restrictions to the adversary in terms of the number of communication rounds of valid sessions that can be read by the adversary.

In Chapter 5, we analyze security and privacy of a former protocol in [7]. The weak point of this protocol is that if an adversary knows secrets of a tag, it can compute the new value of secrets by monitoring the channel between the tag and the reader. Therefore, it can impersonate the legitimate reader to the tag or the tag to the legitimate reader. Moreover, it is claimed that proposed protocol is secure against forward traceability from the moment that the adversary misses any reader-to-tag communication flow. We show that even under such an assumption, the former proposed protocol is not secure. Further, a new authentication protocol (ACA) for RFID systems is presented. ACA assures the following security features: privacy, resistance to tag and server impersonation, backward untraceability, forward untraceability (under an assumption) and resistance to replay and DoS attacks. Our proposed protocol is secure against forward traceability from the moment the adversary misses any reader-to-tag communication flows. Our protocol provides forward untraceability by making all communication flows necessary to update secrets of the tag and provides complete resistance to server impersonation. Besides, our protocol has efficient performance on the tag and the server side and is the first one to provide all these features at once.

In Chapter 6, we analyze a tree based protocol (SAPA)[8] that is designed to provide perfect privacy, low search complexity for the server and low storage cost for the tag. We discover that this protocol is vulnerable to some attacks including tracing, server impersonation attack and denial of service attack. We also present a new authentication protocol (ACAT) for RFID systems. Our protocol assures the following security features: privacy, resistance to tag and server impersonation, backward untraceability, forward untraceability (under an assumption) and resistance to replay and DoS attacks. Our protocol enhances the security of the existing RFID protocols with high efficiency. The search complexity of the server is greater than the one in the scheme [1], but nearly logarithmic. We know that RFID server is a powerful device so our protocol provides the greatest number of security features by increasing search complexity of the server within the bounds of possibility of it. The computation and communication complexities of the tag is smaller than the ones of the schemes [1, 8]. Our protocol provides forward untraceability by making all communication flows necessary to update the tag secrets and provides complete resistance to server impersonation. Our protocol is the first one to provide all these features at once.

As future work, we will reduce the search complexity of ACA which is still linear to the number of tags in the system. Furthermore, we will improve our RFID privacy model with some realistic limitations to the adversary.

REFERENCES

1. Lu, L., J. Han, L. Hu, Y. Liu, and L. M. Ni, "Dynamic Key-Updating: Privacy-Preserving Authentication for RFID Systems", *Pervasive Computing and Communications, 2007. PerCom '07. Fifth Annual IEEE International Conference on*, pp. 13–22, March 2007.

2. Thornton, F., B. Hanies, A. M. Das, H. Bhargava, A. Campbell, and J. Kleinschmidt, *RFID Security*, Syngress, 2006.

3. Ahson, S. and M. Ilyas, *RFID Handbook: Applications, Technology, Security, and Privacy*, CRC Press, 2008.

4. Garfinkel, S. and B. Rosenberg, *RFID: Applications, Security, and Privacy*, Addison-Wesley, 2005.

5. Landt, J., "Shrouds of Time: The history of RFID", 2001, `www.aimglobal.org/technologies/rfid/resources/shrouds_of_time.pdf`.

6. Cole, P. H. and D. C. Ranasinghe, *Networked RFID Systems and Lightweight Cryptography*, Springer-Verlag, 2008.

7. Song, B. and C. J. Mitchell, "RFID authentication protocol for low-cost tags", *WiSec '08: Proceedings of the first ACM conference on Wireless network security*, pp. 140–147, ACM, New York, NY, USA, 2008.

8. Wang, W., Y. Li, L. Hu, and L. Lu, "Storage-Awareness: RFID Private Authentication based on Sparse Tree", *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2007. SECPerU 2007. Third International Workshop on*,

pp. 61–66, July 2007.

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

10. Cai, S., Y. Li, T. Li, and R. H. Deng, "Attacks and Improvements to an RFID Mutual Authentication Protocol and its Extensions", *WiSec '09: Proceedings of the second ACM conference on Wireless network security*, ACM, New York, NY, USA, 2009.

11. Ohkubo, M., K. Suzuki, and S. Kinoshita, "Efficient hash-chain based RFID privacy protection scheme", *In International Conference on Ubiquitous Computing Ubicomp,Workshop Privacy: Current Status and Future Directions*, Nottingham, England, September 2004.

12. Lim, C. H. and T. Kwon, "Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer", Ning, P., S. Qing, and N. Li (editors), *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*, Vol. 4307 of *Lecture Notes in Computer Science*, Springer, 2006.

13. Hjorth, T., *Supporting Privacy in RFID Systems*, Master's thesis, IMM, DTU, Lyngby, Denmark, December 2004.

14. Weis, S., *Security and privacy in radio-frequency identification devices*, Master's thesis, Massachusetts Institute of Technology (MIT), Massachusetts, USA, May 2003.

15. Feldhofer, M., S. Dominikus, and J. Wolkerstorfer, "Strong Authentication for

RFID Systems using the AES Algorithm", Joye, M. and J.-J. Quisquater (editors), *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2004*, Vol. 3156 of *Lecture Notes in Computer Science*, pp. 357–370, IACR, Springer-Verlag, Boston, Massachusetts, USA, August 2004.

16. Stallings, W., *Cryptography and Network Security*, Printice Hall, 2003.

17. Oorschot, P. and M. Wiener, "Parallel Collision Search with Application to Hash Functions and Discrete Logarithms", *CCS '94: Proceedings of the Second ACM conference on Computer and Communications security*, ACM, New York, NY, USA, 1994.

18. Weis, S., S. Sarma, R. Rivest, and D. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems", Hutter, D., G. Müller, W. Stephan, and M. Ullmann (editors), *International Conference on Security in Pervasive Computing – SPC 2003*, Vol. 2802 of *Lecture Notes in Computer Science*, pp. 454–469, Springer-Verlag, Boppard, Germany, March 2003.

19. Juels, A., "RFID security and privacy: a research survey", *Selected Areas in Communications, IEEE Journal on*, Vol. 24, No. 2, pp. 381–394, Feb. 2006.

20. Ohkubo, M., K. Suzuki, and S. Kinoshita, "Cryptographic approach to "privacy-friendly" tags", *In RFID Privacy Workshop*, MIT, Massachusetts, USA, November 2003.

21. Henrici, D. and P. Müller, "Hash-based Enhancement of Location Privacy for Radio-Frequency Identification Devices using Varying Identifiers", *PERCOMW '04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, p. 149, IEEE Computer Society, Washington, DC, USA, 2004.

22. Avoine, G., *Cryptography in Radio Frequency Identification and Fair Exchange Protocols*, Ph.D. thesis, EPFL, Lausanne, Switzerland, December 2005.

23. Dimitriou, T., "A Lightweight RFID Protocol to protect against Traceability and Cloning attacks", *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm*, IEEE, Athens, Greece, September 2005.

24. Molnar, D. and D. Wagner, "Privacy and security in library RFID: issues, practices, and architectures", *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pp. 210–219, ACM, New York, NY, USA, 2004.

25. Molnar, D., *Security and Privacy in Two RFID Deployments, With New Methods For Private Authentication and RFID Pseudonyms*, Master thesis, University of California Berkeley, Berkeley, California, USA, 2006.

26. Avoine, G. and P. Oechslin, "A Scalable and Provably Secure Hash-Based RFID Protocol", *PERCOMW '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 110–114, IEEE Computer Society, Washington, DC, USA, 2005.

27. Avoine, G., E. Dysli, and P. Oechslin, "Reducing Time Complexity in RFID Systems", Preneel, B. and S. Tavares (editors), *Selected Areas in Cryptography – SAC 2005*, Vol. 3897 of *Lecture Notes in Computer Science*, pp. 291–306, Springer-Verlag, Kingston, Canada, August 2005.

28. Hellman, M., "A cryptanalytic time-memory trade-off", *Information Theory, IEEE Transactions on*, Vol. 26, No. 4, pp. 401–406, Jul 1980.

29. Hopper, N. J. and M. Blum, "Secure Human Identification Protocols", *ASIACRYPT*, pp. 52–66, 2001.

30. Juels, A. and S. Weis, "Authenticating Pervasive Devices with Human Protocols", Shoup, V. (editor), *Advances in Cryptology – CRYPTO'05*, Vol. 3126 of *Lecture Notes in Computer Science*, pp. 293–308, IACR, Springer-Verlag, Santa Barbara, California, USA, August 2005.

31. Gilbert, H., M. Robshaw, and H. Sibert, "An Active Attack Against HB^+ – A provably Secure Lightweight Authentication Protocol", Manuscript, July 2005.

32. Bringer, J., H. Chabanne, and D. Emmanuelle, "HB^{++}: a Lightweight Authentication Protocol Secure against Some Attacks", *IEEE International Conference on Pervasive Services, Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing – SecPerU 2006*, IEEE, IEEE Computer Society Press, Lyon, France, June 2006.

33. Piramuthu, S., "HB and Related Lightweight Authentication Protocols for Secure RFID Tag/Reader Authentication", *Collaborative Electronic Commerce Technology and Research – CollECTeR 2006*, Basel, Switzerland, June 2006.

34. Nguyen Duc, D. and K. Kim, "Securing HB^+ against GRS Man-in-the-Middle Attack", *Proceedings of the Symposium on Cryptography and Information Security (SCIS2007)*, 2007.

35. Piramuthu, S. and Y. Tu, "Modified HB Authentication Protocol", *Western European Workshop on Research in Cryptology*, Germany, July 2007.

36. Munilla, J. and A. Peinado, "HP-MP: A Further Step in the HB-family of Lightweight authentication protocols", *Computer Networks*, Vol. 51, pp. 2262–2267, July 2007.

37. Gilbert, H., M. J. B. Robshaw, and Y. Seurin, "$HB^{\#}$: Increasing the Security and

Efficiency of HB$^+$", *EUROCRYPT*, pp. 361–378, 2008.

38. Ouafi, K., R. Overbeck, and S. Vaudenay, "On the Security of HB# against a Man-in-the-Middle Attack", *Advances in Cryptology - Asiacrypt 2008*, Vol. 5350 of *Lecture Notes in Computer Science*, pp. 108–124, Springer, Melbourne, Australia, December 2008.

39. Hammouri, G. and B. Sunar, "PUF-HB: A Tamper-Resilient HB Based Authentication Protocol", *ACNS*, pp. 346–365, 2008.

40. Leng, X., K. Mayes, and K. Markantonakis, "HB-MP+ Protocol: An Improvement on the HB-MP Protocol", *IEEE International Conference on RFID*, pp. 118–124, April 2008.

41. Karthikeyan, S. and M. Nesterenko, "RFID Security without Extensive Cryptography", *Workshop on Security of Ad Hoc and Sensor Networks – SASN'05*, pp. 63–67, ACM, ACM Press, Alexandria, Virginia, USA, November 2005.

42. Chien, H.-Y. and C.-H. Chen, "Mutual Authentication Protocol for RFID Conforming to EPC Class 1 Generation 2 standards", *Computer Standarts & Interfaces, Elsevier Science Publishers*, Vol. 29, No. 2, pp. 254–259, February 2007.

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

44. 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 — ACNS 2008*, Vol. 5037 of *Lecture Notes in Computer Science*, pp. 479–489, Springer, New York, USA, June 2008.

45. Ouafi, K. and R. C.-W. Phan, "Privacy of Recent RFID Authentication Protocols", *4th International Conference on Information Security Practice and Experience – ISPEC 2008*, Vol. 4991 of *Lecture Notes in Computer Science*, pp. 263–277, Springer, Sydney, Australia, April 2008.

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

47. Juels, A. and S. Weis, "Defining Strong Privacy for RFID", *International Conference on Pervasive Computing and Communications – PerCom 2007*, pp. 342–347, IEEE, IEEE Computer Society Press, New York, USA, March 2007.

48. Vaudenay, S., "On Privacy Models for RFID", *Advances in Cryptology - Asiacrypt 2007*, Vol. 4833 of *Lecture Notes in Computer Science*, pp. 68–87, Springer-Verlag, Kuching, Malaysia, December 2007.