RFID Authentication Protocols using Symmetric Cryptography

Boyeon Song

Technical Report
RHUL–MA–2009–24
16 December 2009

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England

http://www.rhul.ac.uk/mathematics/techreports

RFID Authentication Protocols using Symmetric Cryptography

Boyeon Song

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
Department of Mathematics
Royal Holloway, University of London

2009

Declaration

These doctoral studies were conducted under the supervision of Prof. Chris J. Mitchell.

The work presented in this thesis is the result of original research carried out by myself, in collaboration with others, whilst enrolled in the Information Security Group of Royal Holloway, University of London as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Boyeon Song
December 2009

Acknowledgements

I dedicate my thesis to my parents. Their love, trust and support have given me great comforts and encouragements through my PhD student life. I can never thank them enough.

I would like to express my heartfelt gratitude to my supervisor, Chris J. Mitchell. His sharp and excellent comments and guidance have refined and polished my PhD work.

I am thankful to my colleges and staffs in the Information Security Group for their helps in various ways.

Christ Church Virginia Water was a wonderful blessing for me. Their cares, prayers and services have sustained and brighten my life in the UK.

Most of all, thanks to God for what He has done for me; He has been with me wherever I go and has strengthened me. I will praise the Lord all my life.

Abstract

Radio Frequency IDentification (RFID) is emerging in a variety of applications as an important technology for identifying and tracking goods and assets. The spread of RFID technology, however, also gives rise to significant user privacy and security issues. One possible solution to these challenges is the use of a privacy-enhancing cryptographic protocol to protect RFID communications.

This thesis considers RFID authentication protocols that make use of symmetric cryptography. We first identify the privacy, security and performance requirements for RFID systems. We then review recent related work, and assess the capabilities of previously proposed protocols with respect to the identified privacy, security and performance properties.

The thesis makes four main contributions. First, we introduce server impersonation attacks as a novel security threat to RFID protocols. RFID tag memory is generally not tamper-proof, since tag costs must be kept low, and thus it is vulnerable to compromise by physical attacks. We show that such attacks can give rise to desynchronisation between server and tag in a number of existing RFID authentication protocols. We also describe possible countermeasures to this novel class of attacks.

Second, we propose a new authentication protocol for RFID systems that provides most of the identified privacy and security features. The new protocol resists tag information leakage, tag location tracking, replay attacks, denial of service attacks and backward traceability. It is also more resistant to forward traceability and server impersonation attacks than previously proposed schemes. The scheme requires less tag-side storage than existing protocols and requires only a moderate level of tag-side computation.

Next, we survey the security requirements for RFID tag ownership transfer. In some applications, the bearer of an RFID tag might change, with corresponding changes required for the RFID system infrastructure. We propose novel authentication protocols for tag ownership and authorisation transfer. The proposed protocols satisfy the requirements presented, and have desirable performance characteristics.

Finally, we address the issue of scalability in anonymous RFID authentication protocols. Many previously proposed protocols suffer from scalability issues because they require a linear search to identify or authenticate a tag. Some RFID protocols, however, only require constant time for tag identification; unfortunately, all previously proposed schemes of this type have serious shortcomings. We propose a novel RFID pseudonym protocol that takes constant time to authenticate a tag, and meets the identified privacy, security and performance requirements. The proposed scheme also supports tag delegation and ownership transfer in an efficient way.

Contents

List of Figures

List of Tables

Abbreviations

BMM	Burmester-de Medeiros-Motta
CC	Chien-Chen
D	Dimitriou
DB	Database
DoS	Denial-of-Service
DPLK	Duc-Park-Lee-Kim
EPC	Electronic Product Code
FA	Fouladgar-Afifi
HAC	Hash-based Access Control
HM	Henrici-Müller
LK	Lim-Kim
MAC	Message Authentication Code
MitM	Man-in-the-Middle
MSW	Molnar-Soppera-Wagner
MW	Molnar-Wagner
OSK	Ohkubo-Suzki-Kinoshita
OTYT	Osaka-Takagi-Yamazaki-Takahashi
PHER	Peris-Lopez-Hernandez-Castro-Estevez-Tapiador-Ribagorda
PRBG	Pseudo-Random Bit Generator
PRN	Pseudo-Random Number
PRF	Pseudo-Random Function
RAC	Randomised Access Control
RF	Radio Frequency
RFID	Radio Frequency IDentification
SA	Strong Attacker
SIS	Saito-Imamoto-Sakurai
T	Tsudik
TC	Trusted Centre
WA	Weak Attacker
XOR	eXclusive OR

Notation

We use the following notation throughout the thesis, unless otherwise stated.

n	The number of tags that a server manages
r	A random string
S	A server
T	A tag
T_i	The i-th tag ($1 \leq i \leq n$)
$\hat{x}$	The most recent value of x (for any value x)
$\overline{x}$	The updated value of x (for any value x)
$x \gg a$	Right circular shift operator, which rotates all bits of a bit-string x to the right by a bits, as if the right and left ends of x were joined.
$x \ll a$	Left circular shift operator, which rotates all bits of a bit-string x to the left by a bits, as if the left and right ends of x were joined.
$\oplus$	XOR operator
$\|$	Concatenation operator
$\leftarrow$	Substitution operator
$\stackrel{?}{=}$	An operator that checks whether the right side value equals the left side value

Chapter 1

Introduction

Contents

This chapter gives an overview of the thesis as a whole. In section 1.1 we present the motivation for the research, and in the following section we describe the main contributions of the thesis. Section 1.3 provides an overview of the overall structure of the thesis. Finally, we present a list of publications related to the thesis.

1.1 Motivation

Radio Frequency IDentification (RFID) is an automatic identification technology that uses radio waves to identify objects such as products, animals or persons. The use of RFID has become widespread, including in point of sale applications [29], product tracking in a supply chain [34, 38], transport payments [22, 42], entry access control [29], animal supervision [22, 36], automated vehicle identification [22, 29], library book administration [29, 46], patient identification in hospitals [29], and electronic passports [30, 39].

The main benefits of RFID systems are that they can provide automated contactless identification of a range of physical entities, and can be used to track valuable objects. However, the use of such RFID tags gives rise to serious privacy and security concerns, including the possibility of eavesdropping, snooping, cloning, and tracking of end users [19, 23, 24, 29, 71]. These concerns arise from the ways in which RFID tags operate.

An RFID reader and an RFID tag communicate via a wireless radio communications channel. Thus, interactions between a reader and a tag are susceptible to eavesdropping and/or manipulation. Also, each RFID tag has a unique value that is used to identify it. If a tag emits its fixed value to every reader that queries it, then the location of the tag can be tracked, and thus the privacy of the tag holder could be invaded. Moreover, an RFID tag is typically designed to be inexpensive for mass distribution. Such a low-cost tag has limited memory capacity and processing ability, and its memory is typically not tamper-resistant. Thus, information stored in an RFID tag, including stored identifers and keys, is vulnerable to compromise, e.g. by side-channel attacks.

In recent years, a considerable volume of papers have been published providing solutions to these RFID security and privacy challenges. One approach to addressing such privacy and security threats is to use a tag authentication scheme in which a tag is both identified and verified in a manner that does not reveal the tag identity to an eavesdropper. Many RFID authentication protocols use cryptographic techniques to protect messages exchanged over a radio frequency interface from eavesdropping. However, RFID tags have limited processing power and storage because of tight tag cost requirements. Thus, in current RFID tags it is infeasible to use computationally intensive cryptographic algorithms, such as public key cryptographic techniques. Instead, symmetric cryptographic schemes, such as hash functions and symmetric encryption algorithms, are commonly used. As a result, authentication protocols for RFID systems should not only be designed to address privacy and security threats, but should also take into account the limited capabilities of RFID tags.

A wide variety of protocols for RFID protocols using symmetric cryptosystems have been proposed. However, as we show in chapter 5, they all have privacy, security, and/or performance drawbacks.

For these reasons, this thesis focuses on the design of RFID authentication protocols using symmetric cryptographic techniques. The thesis begins by identifying the privacy, security and performance requirements for such protocols, and analyses the strengths and weaknesses of existing protocols. We aim to propose novel RFID authentication protocols that meet the identified requirements.

1.2 Main Contributions

As discussed above, in this thesis we consider RFID identification/authentication protocols that use symmetric cryptography, such as hash functions and MACs. The main contributions of the thesis are as follows:

1. We identify the privacy, security and performance requirements for RFID protocols.
2. We assess the prior art against the identified privacy, security and performance requirements.
3. We introduce a novel security threat, namely server impersonation attacks, which we argue poses a genuine risk to RFID protocols.
4. We propose a new RFID authentication protocol that provides most of the identified privacy, security and performance requirements.
5. We present requirements for secure RFID tag ownership transfer, and propose RFID authentication protocols for tag ownership transfer that satisfy these requirements and that have desirable performance characteristics.
6. We propose an RFID pseudonym protocol with desirable scalability properties that takes constant time to authenticate a tag and that meets the identified privacy, security and performance features.

1.3 Organisation

The remainder of the thesis is organised as follows:

- Chapter 2 introduces the security goals and cryptographic primitives that are used in this thesis.
- Chapter 3 provides an overview of RFID systems. We outline the history of RFID technology, and describe RFID components, protocols and applications.
- In chapter 4 we investigate the possible threats to RFID systems, and identify associated privacy, security and performance requirements. We also discuss

two additional functional requirements, namely tag delegation and tag ownership transfer.

- In chapter 5 we assess recently proposed RFID protocols that use symmetric cryptography against the identified privacy, security and performance requirements.

- In chapter 6 we introduce server impersonation attacks, a practical security threat to RFID security protocols that has not previously been described. We show how such attacks can give rise to desynchronisation between server and tag in a number of existing RFID protocols.

- In chapter 7 we propose an RFID authentication protocol that provides the identified privacy and security features and that has modest tag resource requirements.

- In chapter 8 we survey the security requirements for RFID tag ownership transfer and identify desirable features. We also propose novel authentication protocols for tag ownership and authorisation transfer that satisfy the identified requirements.

- Chapter 9 considers the issue of scalability in anonymous RFID authentication protocols. We propose a novel RFID authentication protocol with desirable scalability properties, and that possesses the identified desirable privacy, security and performance properties.

- In chapter 10 we summarise the contributions of the thesis, and identify directions for future research.

1.4 Publications

This thesis contains material that has been published or submitted for publication in [63, 64, 65, 66, 67].

The contents of [64] form the basis for chapter 6, and the contents of [65] form the basis for chapter 7. The contents of [63] have been updated since publication, and an updated version can be found in [67]. The contents of [63] form the basis for chapter 8, and the contents of [66] form the basis for chapter 9.

Chapter 2

Cryptographic Techniques and Protocols

Contents

This chapter introduces general communications security goals and cryptographic primitives. In section 2.1 we identify communications security goals that are relevant to this thesis, and in section 2.2 we present cryptographic primitives that will be used to meet these goals in the schemes we consider here. Section 2.3 defines authentication protocols, a major class of security techniques.

2.1 Communications Security Goals

To effectively assess security needs, and evaluate/choose the most effective solution for a particular application, a definition of the security goals or requirements for that application is needed [27]. In this section, we present general communications security goals that are relevant to this thesis.

The main communication security goals, also known as security services, can be defined as follows [27, 43]:

- **Confidentiality** is a service that denies read access to information to all but those authorised to have it. There are numerous approaches to providing confidentiality, ranging from physical protection to mathematical algorithms which render data unintelligible. Encryption can be used to meet this goal.

- **Data integrity** is a service which addresses the unauthorised alteration of data. To provide data integrity for data communicated across an unreliable channel, it must be possible to detect data manipulation by unauthorised parties. Data manipulation includes such things as insertion, deletion and substitution. Note that this contrasts with data integrity provision for data stored within a computer system, where it is possible (at least in principle) to prevent modification to data. That is, in a computer system the objective of an access control system is typically to control access to data thereby preventing unauthorised modifications, whereas in communications system it is not possible to prevent changes, only to detect them.

- **Authentication** is a term used with two distinct, albeit related, meanings. **Data origin authentication** enables the recipient of transmitted data to verify its origin. **Entity authentication** enables a party in a communication session to verify, at an instant in time, the identity of the other party in the session; mutual authentication refers to the provision of entity authentication for both parties.

- **Non-repudiation** is a service which prevents an entity from denying having made a commitment or performed an action. The provision of the service involves the generation of evidence about the commitment or action. In a communications context, non-repudiation of origin involves the provision of evidence about the transmission of a message, that has value even if the message originator subsequently denies having sent the message.

- **Access control** provides protection against unauthorised use of resources, e.g. the use of a communications resource; the reading, writing, or deletion of an information resource; or the execution of a processing resource.

Besides these security goals, certain general usability requirements also play an important role in developing security solutions [27]:

- **Scalability**: a network protocol is said to be scalable if the number of nodes can be significantly increased without imposing an unacceptable workload on any entity in the network. The interpretation of unacceptable will vary depending on the context (and the size of the network). For example, a load that is quadratic function of the number of network nodes may be unacceptable if the number of network nodes becomes large, whereas a load that is linear (or less, e.g. logarithmic) in the network size is more likely to be acceptable, even for very large networks. Any security scheme deployed in a network should not significantly affect its scalability. In the context of secure RFID systems, we would typically require that the workload on the server to complete a single transaction should not be a linear function of the number of deployed RFID tags.
- **Performance**: security features must have minimal impact on network performance. This is especially important for real-time communications, where meeting the security requirements must not prevent the provision of the required quality of service. Performance also goes hand in hand with the resource usage of the medium; the security solutions must not, for example, cause a decrease in the overall capacity of the network.

2.2 Cryptographic Primitives

Cryptography is the study of mathematical techniques to hide information [43]. The fundamental objective of cryptography is to enable two parties to communicate over an insecure channel in such a way that an adversary cannot understand and/or manipulate what is being said [69]. This channel could be a telephone line, computer network, or wireless interface [69].

Cryptographic techniques can be divided into two main classes, symmetric and asymmetric techniques, depending on the nature of the keys used [43, 44, 69]. In symmetric cryptography (also called secret key cryptography), the sender and receiver share a common secret key. In asymmetric cryptography (also called public key cryptography), every participating entity has its own key pair, made up of a private key, which is kept secret by its owner, and a public key, which can be disseminated freely.

2.2.1 Symmetric Techniques

When using a symmetric cryptographic algorithm to protect a transmitted message, the sender and receiver must share a secret key. The precise use of the key will depend on the nature of the protection provided by the algorithm being used (e.g. to protect the confidentiality or the integrity of the message).

The main classes of symmetric techniques are: encryption algorithms, message authentication code algorithms, hash functions and pseudo-random bit generators, each of which are discussed in greater detail below.

Symmetric Encryption

There are a variety of different types of symmetric encryption techniques, also known as secret key encryption algorithms. The most widely discussed class of symmetric cipher is the block cipher. In a block cipher, data are processed in blocks, for example, of 64 or 128 bits [44]. A block cipher algorithm is made up of encryption and decryption functions [43, 44, 69]. Encryption takes as input a block of plaintext and a secret key, and outputs a block of ciphertext [43, 44, 69]. Decryption, when given the same secret key, always maps a ciphertext block back to the correct plaintext block [43, 44, 69].

The principle function of encryption is to provide the confidentiality service for transmitted or stored data [43, 44, 69]. However, it is possible to provide other services if encryption is used in appropriate ways [44].

Message Authentication Codes

A Message Authentication Code (MAC) algorithm is a cryptographic function that takes as input a message and a secret key, and outputs a short, fixed length, block of bits known as the MAC [43, 44, 69]. This MAC is then sent or stored with the message, and acts to protect its integrity and guarantee its origin [43, 44, 69]. If the recipient of a MAC is equipped with the correct secret key, then the key can be used with the received message to re-compute the MAC value [43, 44, 69]. If this re-computed value agrees with the MAC value sent or stored with the message, then the recipient knows that the message has not been changed and that it must have been sent by someone who knows the secret key (presumably the legitimate originator) [43, 44, 69].

A MAC algorithm is a family of functions f parameterised by a secret key k, with the following properties [43]:

- Easy of computation: given a value k and an input x, the MAC $f_k(x)$ is easy to compute.
- Compression: f maps an input x of arbitrary finite bit-length to an output $f_k(x)$ of fixed bit-length l (e.g. $l = 64$ or 128).
- Forgery-resistance: given a sequence of text-MAC pairs $(x', f_k(x'))$ for a fixed key k, it is computationally infeasible to compute a text-MAC pair $(x, f_k(x))$ for any $x \neq x'$.

Hash Functions

Hash functions are somewhat different to the functions we have considered so far in that they do not use keys. A hash function takes an input an arbitrary data string and gives as output a short, fixed-length value that is a function of the entire input; this output is known as a hash code or hash value [43, 44, 69]. Hash functions must have the one-way property, that is, they must be designed so that they are simple and efficient to compute, but also so that given an arbitrary output, it is computationally infeasible to find an input that gives the chosen output [43, 44, 69].

That is, a hash function is an efficiently computable function which maps an arbitrary length input to a fixed length output; i.e. $h : \{0,1\}^* \to \{0,1\}^l$. The basic requirements for a cryptographic hash function are as follows [43]:

- Preimage resistance: for any output y, it is computationally infeasible to find an input x such that $h(x) = y$, given no corresponding input is known.
- 2nd-preimage resistance: given x, it is computationally infeasible to find $x' \neq x$ such that $h(x) = h(x')$.

- Collision resistance: it is computationally infeasible to find any pair of distinct inputs x and x' such that $h(x) = h(x')$.

Pseudo-Random Bit Generators

A pseudo-random bit generator (PRBG) is a deterministic algorithm which, given a truly random binary sequence of length m, outputs a binary sequence of length $l > m$ which appears to be random. The input to the PRBG is called the seed, while the output of the PRBG is called a pseudo-random bit sequence [43, 68].

The security strength of a PRBG depends on a variety of factors, including the period and probability distribution of the output sequence [14].

2.2.2 Asymmetric Techniques

We now discuss two of the main classes of asymmetric algorithms, namely asymmetric encryption algorithms and digital signature schemes.

Unlike symmetric cryptosystems, which make use of a single key known to sender and receiver, asymmetric cryptosystems employ two keys, a public key and a private key. These public and private keys are related mathematically, and an entity's private key cannot be derived from its public key [43, 44, 68, 69].

Asymmetric Encryption

Asymmetric encryption, also known as public key encryption, involves an encryption operation that transforms blocks of plaintext into ciphertext blocks, and a decryption operation that reverses this process [43, 44, 68, 69]. The main difference from symmetric encryption is the way in which keys are used [43, 44, 68, 69]. The public key of the intended recipient of a message is used for encryption and the recipient's private key is used for decryption [43, 44, 68, 69]. A user's public key is made available to anyone who wants to encrypt a message intended for that user; the recipient's private key is used to decrypt received encrypted messages [78].

Implementing such an algorithm requires the computation of complex mathematical functions, e.g. involving multi-precision integer or finite field arithmetic [44]. As a result, public key encryption schemes tend to be more computationally intensive, and hence slower to compute, than secret key encryption algorithms [44]. Because of this, simple wireless devices such as RFID tags are likely to lack the computational power necessary to handle asymmetric encryption algorithms [44].

Digital Signatures

A digital signature is computed as a function of the message to be signed using the signer's private key, and can then be verified by anyone equipped with the signer's public key [43, 44, 68, 69]. When computing a signature it is almost always the case that a hash function is applied to the message being signed. The most common form of a signature gives a value that, much like a MAC, is sent or stored with the message it is protecting [43, 44, 68, 69].

One key difference from a MAC is the way in which signatures are verified [44]. Verifying a MAC essentially involves re-computing it. However, verifying a digital signature uses a special verification function that takes as input the signature, the message and the public verification key, and gives as output an indication as to whether the signature is valid or not [44]. Thus, just because an entity can verify the correctness of a signature, does not mean that it is possible to forge a signature [44]. Thus, as well as being able to provide data integrity and data origin authentication functions, a digital signature can also provide non-repudiation services. The disadvantage is that digital signature functions are generally significantly more complex to compute than MAC functions [44].

2.3 Authentication Protocols

We also briefly introduce authentication protocols, a major class of security techniques that incorporate the use of cryptographic algorithms [44].

An authentication protocol is a defined exchange of messages between two (or possibly more) parties, with the objective of providing one or both parties with an entity authentication service [43, 44, 60]. That is, the objective is for one or both of the parties to verify the identity of who it is they are exchanging messages with, and that the other party is actively involved in the protocol, that is, that the messages are not replayed versions of ‘old’ messages [43, 44, 60].

Authentication protocols make use of cryptographic techniques to protect the origin and integrity of individual messages [43, 44, 60]. One common approach is to employ MACs for this purpose. As an alternative to the use of MACs to protect the protocol messages, it is also possible to use digital signatures [43, 44].

Chapter 3

Overview of RFID

Contents

This chapter provides an overview of RFID systems. We outline the history of RFID technology in section 3.1, and describe the components of an RFID system in section 3.2. Sections 3.3 and 3.4 introduce RFID protocols and applications.

3.1 The History of RFID Technology

RFID is a wireless automatic identification and data capture technology that uses radio communications. RFID technology can be used for automatically identifying objects such as products, animals or persons, collecting data about them, and entering that data directly into computer systems. We now briefly describe below how RFID technology started and evolved.

The roots of RFID technology can be traced back to the Second World War, where radar technology was employed to detect incoming aircraft by sending out pulses of radio energy and detecting the echoes that came back [57]. The Germans, Japanese, Americans and British all used radar to warn of approaching planes while they were still miles away [57]. However, there was no way to identify which planes belonged to whom [36, 57]. The Germans attempted to solve the identification problem by simultaneously rolling their aircraft in response to a signal from the ground radar station [36, 57]. This would change the radar reflection's polarisation, creating a distinctive blip on the radars [57]. This system was the first demonstration of active RFID [57].

The British responded by creating the Identify Friend or Foe (IFF) system [57]. This involved placing a transponder on each plane which, when it received signals from a radar station on the ground, broadcasts a signal back that identified the aircraft as friendly [36, 57]. RFID uses the same basic concept [36, 57]. When a signal is sent to a transponder, the transponder wakes up and either reflects back a signal (passive system) or broadcasts a signal (active system) [36, 57].

An early paper exploring the idea behind RFID was published in 1948 by Stockman [70], which contains the first public description of RFID technology [57]. Advances in radar and RF communications systems continued through the 1950s and 1960s [36]. Several technologies related to RFID were developed, such as the long-range transponder systems of IFF for aircraft [36].

The 1960s were the prelude to the explosion of interest in RFID systems of the 1970s; the theory of RFID was developed, and application field trials started [36]. Harrington studied electromagnetic theory related to RFID [36, 57]. The first RFID companies, Sensormatic and Checkpoint, were founded in the late 1960s. They developed a commercial application, Electronic Article Surveillance (EAS), designed to counter theft, jointly with other companies such as Knogo [36].

In the 1970s, there was a major advance in RFID technology [36]. A wide range of bodies, including companies, academic institutions, and government laboratories actively worked on RFID, and notable advances were realised [36]. For example, the Los Alamos National Laboratory developed a system for tracking nuclear materials [36]. The concept of putting an RFID tag in a truck and using RFID readers at the gates of secure facilities to identify them was also developed [36].

The 1980s saw the practical implementation of RFID technology, to varying degrees in different parts of the world [36]. Applications emerged in transport, industry, personnel access and animal supervision [36].

The 1990s saw the widespread adoption of electronic toll collection throughout the United States [36]. In the early 1990s, IBM engineers developed and patented an ultra-high frequency (UHF) RFID system. UHF offered longer read range and faster data transfer [36, 57].

UHF RFID was further developed in 1999, when the Uniform Code Council, European Article Number (EAN) international, Procter & Gamble and Gillette funded the establishment of the Auto-ID Centre at the Massachusetts Institute of Technology (MIT) [36, 57]. At MIT, Brock and Sarma investigated the possibility of attaching low-cost RFID tags to products, in order to track them through the supply chain [36, 57]. Their proposal involved putting only a serial number on the tag to keep the price down; a simple microchip that stores very little information is less expensive to produce than a more complex chip with more memory [36, 57]. Data associated with the serial number on the tag would be stored in a database that would be accessible over the Internet [36, 57].

Between 1999 and 2003, the Auto-ID Centre gained the support of more than 100 large end-user companies, as well as the U. S. Department of Defence and many key RFID vendors [36, 57]. It established research laboratories in Australia, the United Kingdom, Switzerland, Japan and China [36, 57]. It also developed two air interface protocols (Class 0 and Class 1), the Electronic Product Code (EPC) numbering scheme, and a network architecture designed to support retrieval of data associated with an RFID tag on the Internet [36, 57]. According to the Generation 1 (or Version 1) specifications [15], EPCglobal Class 0 tags are read-only devices, whereas Class 1 tags are one-time programmable.

The technology was licensed to the Uniform Code Council in 2003, which created EPCglobal, a joint venture with EAN International, to commercialise EPC technology [36, 57]. The Auto-ID Centre shut down in October 2003, and its research responsibilities were passed to Auto-ID Labs and EPCglobal, which was responsible for managing the new EPC Network [36, 57].

Some of the largest retailers in the world, including Albertsons, Metro, Target, Tesco, Wal-Mart, as well as the U. S. Department of Defence, have described their plans to integrate EPC technology into goods tracking for their supply chains [36, 57]. The pharmaceuticals, tyre, defence and other industries are also moving to adopt the technology [36, 57]. EPCglobal ratified a second-generation standard in December 2004, paving the way for broad adoption [36, 57].

RFID has demonstrated its importance in a wide range of markets, including livestock identification and automated vehicle identification systems. This broad range of applications arises from its capability for tracking moving objects [36, 57].

3.2 RFID Systems

An RFID system consists of three key components: RFID tags, RFID readers and a back-end server. Each of these components is now explored in greater detail.

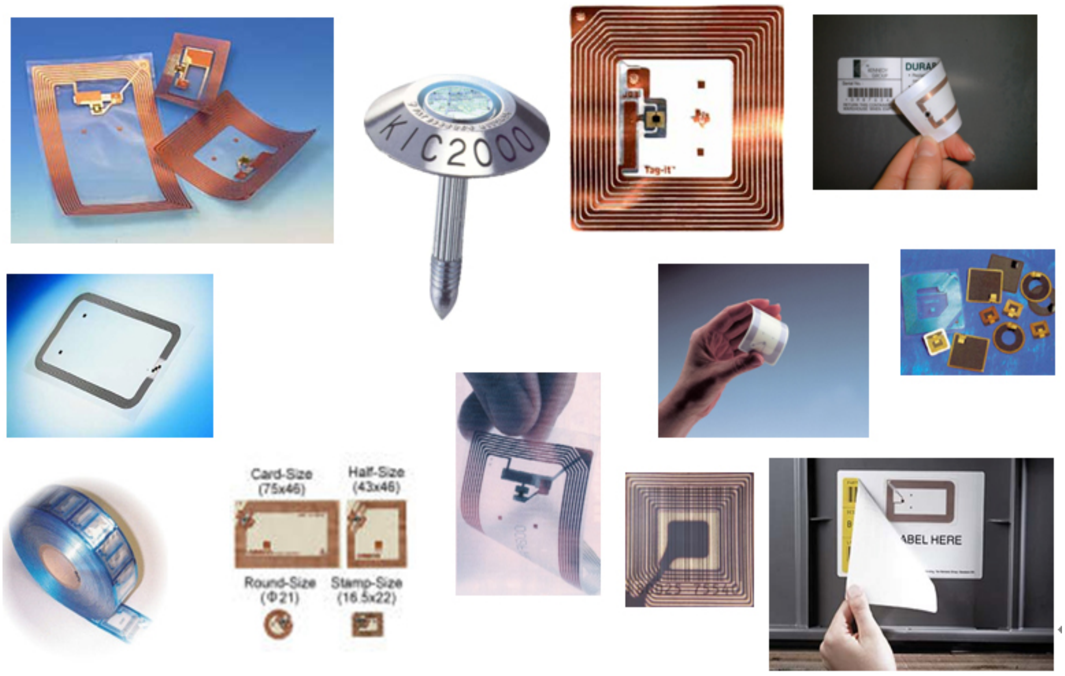
3.2.1 RFID Tags

An RFID tag is an identification device which has a unique identifier and which transmits data over the air using radio frequency (RF) in response to interrogation by an RFID reader [23, 29]. It is also known as a transponder. We use here the term 'tag' for its simplicity.

A tag is designed to receive a specific radio signal and automatically transmit a reply [71]. A tag consists of an integrated circuit and an antenna. The integrated circuit is for processing data; it modulates and demodulates radio signals, and stores and processes information. The antenna is used for communicating via an RF signal with RFID readers. The memory on tags may be read-only, write-once read-many, or fully rewritable [79]. Figure 3.1 shows some examples of tags.

Processing capacity

RFID devices can be divided into two broad classes, 'dumb' and 'smart'. A dumb tag has no significant processing power, its unique identifier will normally be a fixed length value, typically 10 or 16 hexadecimal digits long, and its memory capacity is likely to be fairly small — of the order of a few hundred bytes to a maximum



Source: *http://www.anbitarabia.com/rfid.htm*

Figure 3.1: RFID tags

of around 2kBytes [37]. In its simplest implementation, a tag listens for a radio signal, and sends a signal of its own as a reply [71]. More complicated systems may transmit a single letter or digit back to the source, or send multiple strings of letters and numbers [71].

A smart tag, by contrast, has on-board processors that are typically capable of performing cryptographic operations [37]. It will often have a much larger memory capacity of 32kBytes or more, and be capable of requiring authentication before allowing access to stored data [37]. Such a tag may also be able to encrypt communications using session keys to avoid snooping or data injection attacks [37].

Power supply

Tags can be categorised into three classes according to the nature of the power supply (if present): active, passive and semi-passive tags [71].

Passive tags do not contain a battery or other power source. Instead they contain a resonant circuit capable of absorbing power from an RFID reader's antenna [71].

Table 3.1: RFID operation frequencies

Name	Range	Read range	Typical applications
LF	$30 - 300$ kHz	50 cm	Pet identification
HF	$3 - 30$ MHz	3 m	Building access control
UHF	300 MHz $-$ 3 GHz	9 m	Box and pallet tracking
Microwave	$>$ 3 GHz	$>$ 10 m	Vehicle identification

Source: [23], pages 59–60

Thus, they must wait for a signal from an RFID reader in close proximity [71].

Active tags have their own power source, usually an internal battery. Since they contain a battery to power the radio circuitry, they can actively transmit and receive without having to be powered by an RFID reader's antenna, and can therefore interact with an RFID reader at significantly greater distances [71].

A semi-passive tag has a battery to power its memory circuitry, but relies on signals from an RFID reader's antenna to power its radio circuits when receiving and sending data [71].

Operating frequency

The operating frequency is the electromagnetic frequency used by a tag to communicate and/or obtain power. The electromagnetic spectrum within which RFID systems typically operate is commonly divided into low frequency (LF), high frequency (HF), ultra-high frequency (UHF), and microwave [23].

Different frequencies have different properties. Lower frequency signals are better able to travel through water, while higher frequencies can carry information at higher rates [23]. Higher frequency signals are typically also easier to read at a distance. Table 3.1 [23] shows the read ranges for the four main frequency ranges, and how they have been used in applications.

Stored data

The volume of data carried by a tag can range anywhere from a few bytes up to several megabytes, depending on the application and the individual tag [71]. The data carried in a tag can be in a variety of formats. In the absence of ratified standards, many tag data formats are proprietary in nature, although standards are now emerging [71].

The Electronic Product Code (EPC) is the RFID-based replacement for the Universal Product Code (UPC) bar code. The latter allows for 10^{10} products, and its numbers are quickly being used up [71]. The EPC uses the EPCglobal organisation's General Identifier (GID-96) format. A GID-96 identifier contains 96 bits (12 bytes) of data. Under the GID-96 standard, every EPC consists of three separate fields: the 28-bit General Manager Number that identifies the company or organisation; the 24-bit Object Class that divides products into groups; and the 36-bit serial number that is unique to the individual object. A fourth field consists of an 8-bit header that is used to guarantee the uniqueness of the EPC code [71]. This allows for a total of 3×10^{25} unique items under the EPC system [71].

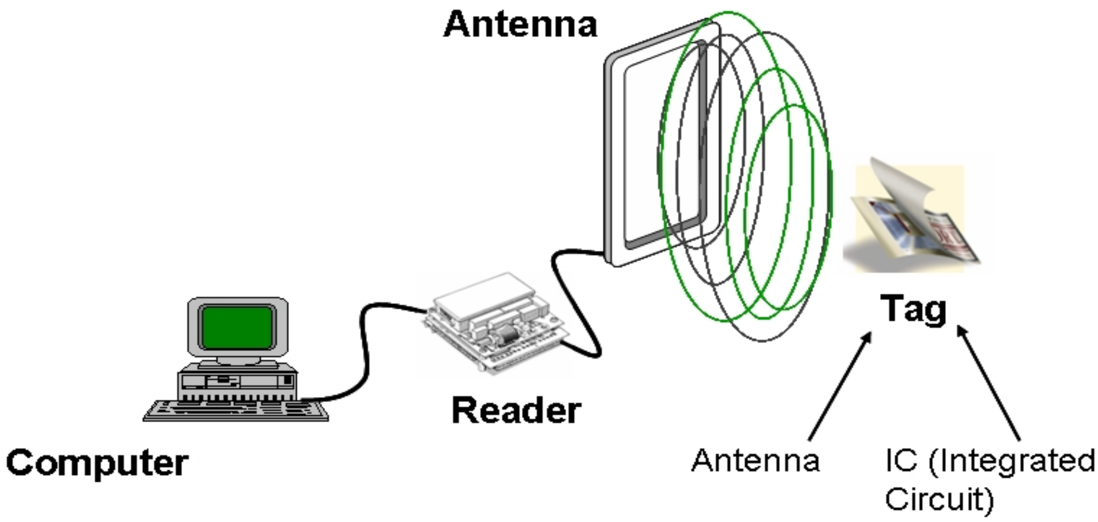
3.2.2 RFID Readers

An RFID reader is a device that can recognise the presence of RFID tags and read the information supplied by them [23]. It is also known as a transceiver or interrogator. Here, we simply use the term 'reader'. A reader uses its antenna to query and receive data from tags by broadcasting an RF signal. RFID readers are typically connected to a back-end server equipped with a database of tag information [71]. In such a case, the reader forwards tag responses to this back-end server for further processing.

3.2.3 Back-end Server

A back-end server manages a database containing information associated with the RFID tags which it manages, and can retrieve the detailed tag information (or the identity of the item attached to the tag) using the tag response as a key. Figure 3.2 shows how the components relate to one another.

In summary, when a back-end server wants to identify one or more tags, a reader emits an RF signal via its antenna. Any tag within range of the signal responds with certain stored data, such as a tag identifier. The reader then passes the received tag data to the back-end server. The server processes the tag information to identify



Source: *http://www.jesic-tech.com/RFID_tag.html*

Figure 3.2: RFID system

the tag, and can then retrieve any needed information.

3.3 RFID Protocols

RFID systems involve a reader transmitting a radio signal, and a tag receiving this signal answering with a response signal [71]. This response is then processed by a back-end server. Depending on the tag's computational power (if any), the tag may perform cryptographic operations in order to compute its response (and/or update its stored data). The sequence of messages exchanged by the reader, tag and back-end server (along with any computations performed by the parties) constitutes the RFID protocol.

Some tags are 'read-only', while other tags have data 'written' to them. In practice tag protocols are often proprietary, although EPCglobal and the International Organisation for Standardisation (ISO) have defined a number of protocols designed for use in RFID systems [71]. Further information can be found, for example, in [23, 71]

3.3.1 Focus of the Thesis

In this thesis we are concerned with a particular type of RFID tag, i.e. those tags capable of performing (light-weight) cryptographic operations. Therefore, in line with much of the existing research literature, we focus on RFID protocols with the following properties.

- The protocols involve two parties, namely a server and a tag.
- The term 'server' means a combination of a back-end server and its readers. That is, the channel between the back-end server and the readers is secure.
- A server maintains a secure database of information for the tags that it manages, and has a significantly greater processing capability than a tag.
- A tag can perform lightweight cryptographic operations; it can also generate pseudo-random numbers, compute hash functions, and perform symmetric encryption operations.
- A tag has a rewritable memory which may not be tamper-resistant, and thus is susceptible to compromise.
- The channel between the server and tag is an insecure radio-frequency interface, and thus tag-server communications are subject to eavesdropping and/or modification.

3.3.2 Cryptography for RFID

We make the following assumptions about the availability of cryptographic functions suitable for implementation on RFID tags.

- There are sufficiently secure hash functions which are suitable for a low-cost tag.
- There is a sufficiently secure pseudo-random number generator for a low-cost tag.

Implementing asymmetric cryptographic techniques such as RSA in most current RFID tags remains impractical because of the limited resources available to tags [2, 31]. Instead, symmetric cryptographic schemes are commonly applied for RFID systems. Standard symmetric cryptosystems such as SHA-1 or AES are also rather unsuited to today's low-cost tags [18]. However, a number of light-weight implementations of block ciphers and hash functions have recently been studied, including the schemes presented in [11, 17, 40, 55, 58, 61, 80].

The following two hash functions appear to be of particular value in this respect.

- The Whirlpool hash function has been standardised by ISO/IEC and evaluated by the New European Schemes for Signatures, Integrity and Encryption (NESSIE) project [56]. Pramstaller et al. [55] presented a compact hardware implementation of Whirlpool that uses an innovative state representation, and that makes it possible to significantly reduce the required hardware resources.
- Shamir [61] introduced a one-way function called SQUASH (short for SQUarehASH) that is based on modular squaring. SQUASH is ideally suited to challenge-response authentication in highly constrained devices, such as RFID tags, and is provably as secure as factoring [7, 61].

A one-way function such as a cryptographic hash function or a block cipher can be used to generate pseudo-random bit sequences [43]. In practice, an iterated keyed hash function which takes a cheap and weak pseudo-random source (for instance circuitry noise) and an internal key as its inputs could be used as a pseudo-random bit generator [54, 72].

3.4 RFID Applications

Landt [36] claims that RFID technology can increase revenue, improve efficiency and lower costs in business. RFID systems are being adopted in a wide variety of fields, including product management, animal supervision, transportation payments, library book administration, automated vehicle identification, patient identification in hospitals, entry access control, and electronic passports [24, 29, 37]. For example, RFID tags can help hospitals to improve patient care, pharmaceutical companies to reduce counterfeiting, logistics providers to improve the management of moveable assets and supply chains to track goods from the point of manufacture to the retail point of sale [26].

There are many different types of tags, as described in section 3.2.1. The following general guidelines can be used to help select the best type of tag for a particular application [23]:

- Use smart labels for automated application in a warehouse.
- Use passive tags for the lowest cost, and semi-passive or active tags only as necessary for additional capabilities or greater read range.
- Use LF/HF tags for individual items.
- Use UHF tags for shipping units such as pallets.
- Use microwave tags for vehicles and long-distance reading.
- Where possible, to reduce cost, store only an identifier on the tag and look up the rest of the information. Greater on-tag storage is more expensive.
- Follow standards where possible, and watch what the largest adopters are doing.

Chapter 4

Security, Privacy and Performance Requirements

Contents

In this chapter we investigate the possible threats to RFID systems, identify associated privacy, security and performance requirements, and present a list of requirements for RFID protocols, which has been constructed by taking the union of all previously given sets of requirements. Section 4.1 describes the privacy and security requirements for RFID systems, and section 4.2 identifies the performance requirements for such systems. In section 4.3 we also discuss two additional functional requirements, namely tag delegation and tag ownership transfer.

4.1 Privacy and Security

To provide security for a system, the possible threats and risks to that system need to be determined. These can then be used to set security requirements. Finally, countermeasures to the threats and residual risks can be selected and implemented [1].

In this section, we first investigate two main classes of threats to RFID systems, namely threats to privacy and security. The threats that we examine are taken from the existing literature. We also introduce a list of requirements to mitigate such threats.

In this thesis we consider the RFID systems that fit the model given in section 3.3.1.

4.1.1 Privacy

One of the main concerns of users of RFID systems is user privacy. Unprotected communications between a tag and a server over a wireless channel can disclose information about a tag, including its location (and, by implication, the location of its owner).

Threats

Two major privacy issues are as follows [2, 29, 47, 49, 79].

- **Tag Information Leakage** [49]: in a typical RFID system, when a server queries a tag, the tag responds with its identifier. If unauthorised entities can also obtain a tag identifier, then they may be able to request and obtain the private information related to the tag held in the server database. For example, if the information associated with a tag attached to a passport, ID-card or medical record could be obtained by any server, then the damage would be very serious.
- **Tag Tracking** [79]: if the responses of a tag are linkable to each other or distinguishable from those of other tags, then the location of a tag could be tracked by multiple collaborating unauthorised entities. For example, if the response of a tag to a server query is a static ID code, then the movements of the tag can be monitored, and the social interactions of an individual carrying a tag may be available to third parties without him or her knowing.

Privacy Requirements

RFID systems should meet the following privacy requirements in order to mitigate the two threats described above.

- **Tag Information Privacy**: RFID systems should be able to resist tag information leakage. To protect against such a threat, RFID systems need to be controlled so that only authorised entities are able to access the information associated with a tag.
- **Tag Location Privacy**: RFID systems should be able to resist tag tracking attacks. If messages from tags are anonymous, then the problem of tag tracking by unauthorised entities can be avoided.

4.1.2 Security

Communications between a server and a tag via an insecure wireless channel are susceptible to eavesdropping. Security threats feasible to RFID systems are discussed below.

Attack Model

We divide possible attackers into two groups, as follows.

- A weak attacker (WA) is a malicious entity that can observe and manipulate communications between a server and a target tag, but cannot compromise the tag.
- A strong attacker (SA) is a malicious entity that has compromised a target tag, in addition to having the capabilities of a weak attacker.

Threats by an SA as well as a WA should be considered in RFID protocol design, because the internal data in a tag memory are liable to exposure as a result of side-channel attacks [4, 41]. Such attacks are attacks that are based on side channel information that can be retrieved from a device performing cryptographic computations [5]. Side channels provide information about internal computations through measurement, e.g. by monitoring variations in power consumption, external radiation, or the time taken to perform calculations [5]. Side channel attacks make use of such information to recover the key the device is using [5].

Security threats to RFID systems can be classified into weak and strong attacks in line with the attacker types defined above.

Weak attacks

The following attacks are feasible for a WA [2, 29, 79].

- **Tag Impersonation** [79]: a WA could impersonate a target tag to a server without knowing the tag's internal secrets. It could communicate with a server instead of the tag and be authenticated as the tag.
- **Replay attack** [13]: a WA could replay messages exchanged between a server and a tag without being detected, thereby performing a successful authentication between a tag and a server.
- **Man-in-the-Middle (MitM) attack** [28]: a WA could interfere with messages sent between a server and a tag (e.g. by insertion, modification or deletion).
- **Denial-of-Service (DoS) attack** [79]: a WA could block messages transmitted between a server and a tag. Such an attack could cause the server and the tag to lose synchronisation. For example, the server might update its shared secrets, while the tag does not; as a result, they would no longer be able to authenticate each other.

Strong attacks

An SA may be able to perform the following attacks, as well as the weak attacks described above [2, 41, 49].

- **Backward Traceability** [49]: an SA might be able to trace past transactions between a server and a compromised tag using knowledge of the internal state of the tag. That is, given all the internal state of a target tag at time t, the attacker is able to identify target tag interactions that occurred at time $t' < t$. The past transcripts of a tag may allow tracking of the tag owner's past behaviour.

- **Forward Traceability** [41]: an SA might be able to trace future transactions between a server and a compromised tag using knowledge of the internal state of the tag. That is, knowledge of a tag's internal state at time t can help to identify tag interactions that occur at time $t' > t$. The only way of maintaining future security once the current tag secrets have been revealed is to detect key compromise as soon as possible, and to replace the compromised secrets as soon as possible.
- **Server Impersonation** [64]: an SA might be able to impersonate a legitimate server to a compromised tag using knowledge of the internal state of the tag. This attack does not appear to have been discussed previously, despite its potential importance. The SA could, for example, ask the tag to update its internal state, with the effect that the legal server will no longer be able to communicate successfully with the tag, although the SA will. We introduce this novel strong attack in detail in chapter 6.

Resistance to backward traceability is sometimes also referred to as *forward security* [10, 14, 49, 76]. In the thesis, we use the terms backward traceability and forward traceability (defined as in [41]) in order to clearly distinguish between threats to past and future anonymity.

Security requirements

We identify security requirements for RFID systems designed to mitigate the threats of the weak and strong attacks described above, as follows.

- **Resistance to Tag Impersonation**: an adversary should not be able to impersonate a tag without compromising a tag.
- **Resistance to Replay attack**: an adversary should not be able to reuse messages exchanged between a server and a tag, thereby performing a successful session between the tag and the server.

- **Resistance to MitM attack**: an adversary should not be able to manipulate messages sent between a server and a tag without compromising a tag.
- **Resistance to DoS attack**: blocking of messages transmitted between a server and a tag should not mean that the server and the tag can no longer communicate successfully.
- **Backward Untraceability**: an adversary should not be able to to trace past transactions between a server and a tag, even if it compromises the tag.
- **Forward Untraceability**: an adversary should not be able to to trace future transactions between a server and a tag, even if it compromises the tag.
- **Resistance to Server Impersonation**: an adversary should not be able to impersonate a server to a tag, even if it compromises that tag.

4.2 Performance Requirements

RFID schemes cannot use computationally intensive cryptographic algorithms to provide privacy and security because tight tag cost requirements put severe limits on tag-side resources (such as processing power and storage).

RFID schemes should address the following performance issues [2, 4, 33, 47, 74, 79].

- **Storage Capacity** [79]: the volume of data stored in a tag should be minimised because of tight tag cost requirements.
- **Computation** [79]: the complexity of tag computations should be minimised because of the very limited power available to a tag.
- **Communication** [79]: the number and size of messages exchanged between a tag and a reader should be minimised.
- **Scalability** [4]: the server should be able to handle a large tag population. It should be able to identify multiple tags using the same radio channel. Performing an exhaustive search to identify individual tags is difficult when the number of tags is large.

The issue of scalability raises questions about the complexity of searching stored data. As background to subsequent discussions, we briefly review some of the main approaches to searching arrays of data.

The need to search an array for a value is a common problem [48]. For example, in an RFID system, a server maintains a database containing information for tags that it manages. When a server queries a tag and receives a response in return, it examines its database to find an entry whose 'key' value matches the appropriate entry in the tag response. It can now identify the tag and access other information related to it.

Three fundamentally important searching methods applicable when records are stored in arrays are linear search, binary search and hash table search.

- Linear search is the simplest method. It involves examining each element in a list in sequence until a match is found [35, 48]. Its running time is of the order of n, written $O(n)$, where n is the number of elements in the list [35]. Searching relatively small lists sequentially does not require much computer time [48]. However, when the list get longer (as in, for example, telephone directories or lists of credit card customers), linear searches are very inefficient [48]. Since large amounts of computer time could mean considerable costs and/or delays when large amounts of data must be frequently searched, more efficient means of searching are needed [48].

- Binary search is a more sophisticated search algorithm. It involves examining the middle entry in an (ordered) array to see which half of the array contains the desired value [48]. The middle value of the appropriate half is then examined to see which half of this half contains the value in question [48]. This halving process is continued until the value is located or it is determined that the value is not in the list [48]. Clearly, a binary search requires that the list must be sorted before searching [35, 48]. The sorting process has its own costs which should be evaluated [48]. Binary search runs in $O(\log n)$ time [35]. This is significantly better than linear search for large lists of data [35].

  When n is large, binary search takes much less time, in the worst case, than linear search. This is because it makes $\log n$ rather than n comparisons [35]. However, the steps involved in a binary search are typically more complex and more time-consuming than the steps performed in a linear search [35]. Hence, for smaller values of n, it is possible that a linear search will be faster [35].

- Hash tables can also be used for list searching [35]. A hash table is a data structure for the storage of records, that is intended to provide rapid access to a record with a given key value [35]. The search time is a constant, i.e. $O(1)$, in the average case. This search technique requires the ability to assign key values so that all keys are within an ordered range of indices, ideally not much larger than the number of entries in the array; this is achieved using hashing. It also requires sufficient storage to be available to accommodate all the records [35].

  To make a hash table, a hash function is used to map a key into the index (the hash) of an array element where the corresponding value is to be sought [35]. The hash function here is not necessarily a cryptographic hash function. To search for an entry in the hash table, the key is hashed, and the hashed value is then looked up in the table. Of course, there is the possibility of two keys hashing to the same value — this can be resolved, for example, by putting a colliding entry into the next available slot in the table (which is why the table needs to be a little larger than the number of entries in the array).

4.3 Additional Functional Requirements

In this section, we introduce two functional requirements, tag delegation and ownership transfer, that are likely to be required in some RFID systems.

4.3.1 Tag Delegation

In RFID systems, a centralised back-end server is often in charge of a large number of tags. In some systems in which tag responses are anonymous, tag identification requires the back-end server to compute every possible tag output in turn until it finds a match [81]. This can seriously damage scalability [81].

A related issue is that many protocols require a reader to interact with the centralised back-end server in order to identify a tag [81]. In some applications, this reading latency can be an unacceptable overhead [81]. In addition, if the database becomes unavailable for some reason, such as network connectivity failure, all interactions with tags relying on that back-end server will be stopped [81].

Delegation is one possible solution to these performance issues [81]. Delegation enables a back-end server to delegate the right to identify and authenticate a tag to a specified entity, such as a reader [41, 45, 81].

Delegation may be permanent or temporarily [81]. In the first case, a reader is given permanent means to interact with a tag in its read range, and the back-end server is contacted only when a new tag arrives or an old tag leaves the system [81]. In the second case, the back-end server temporarily transfers the right to interact with a set of tags for a limited number of queries, and updates or revokes the delegation according to a delegation policy [81].

4.3.2 Tag Ownership Transfer

Another possible functional requirement for RFID systems is tag ownership transfer. In some applications, an RFID tag may change its owner a number of times during its lifetime. For example, suppose that a manufacturer produces a tag and attaches it to an object, and that a retailer purchases the tagged object from the manufacturer, and then sells it to a customer. The customer may then resell the object at some later time. In such RFID systems, changes of tag owner could occur frequently, and thus a secure and privacy-preserving means of tag ownership transfer is needed.

Tag ownership means having authorisation to identity a tag and control all the related information [41, 45]. Tag ownership transfer implies a shift of such capabilities to a new owner [41, 45], who must therefore be given the necessary private information to securely interact with and identify the tag. Thus all information associated with the tag will need to be passed from the old to the new owner. However, at the moment of tag ownership transfer, both the old and new owners have the information necessary to authenticate a tag, and this fact may cause an infringement of tag owner privacy. More specifically, if the previous owner is malicious, it may still be able to read the tag using retained tag information after transfer, and/or trace the new owner's transactions with the tag. That is, the privacy of the new owner might be compromised by the previous owner. Conversely, if the new owner is malicious, then it might be able to trace the previous owner's past transactions with the tag.

That is, the privacy of the previous owner might be compromised by the new owner. In chapter 8, we identify requirements for secure tag ownership transfer.

Chapter 5

Related Work

Contents

In this chapter we consider previously proposed RFID identification and authentication protocols based on the use of symmetric cryptographic techniques. In each case (in sections 5.2-5.16) we assess them against the privacy, security and performance requirements identified in chapter 4. Comparisons of the properties of these protocols are provided in section 5.17.

5.1 Preliminaries

A variety of protocols for use in RFID systems have been proposed. Many of these protocols use cryptographic functions to protect messages exchanged between an RFID reader and tag and to provide authentication. As discussed in section 3.3.2, implementing asymmetric cryptographic techniques in most current RFID tags remains impractical; instead, symmetric cryptographic schemes are commonly applied. Thus, this thesis focuses on RFID protocols using symmetric cryptographic schemes.

The following notation is used in the protocol descriptions in this chapter.

S	A server
T_i	The i-th tag $(1 \leq i \leq n)$
n	The number of tags that S manages
$\hat{x}$	The most recent value of x (for any value x)
$\overline{x}$	The updated value of x (for any value x)
$\oplus$	XOR operator
$\|\|$	Concatenation operator
$\leftarrow$	Substitution operator
$\stackrel{?}{=}$	An operator that checks whether the right side value equals the left side value

The protocols introduced in this chapter work under the assumptions described in section 3.3.1. Tables 5.1 and 5.2 provide comparisons of the schemes presented in this chapter with respect to the security, privacy and performance properties identified in chapter 4.

In the following sections 5.2-5.16, we review previously proposed RFID protocols presented in the order in which they were published.

5.2 The Weis-Sarma-Rivest-Engels Protocols

Weis et al. [79] proposed two protocols known as Hash-based Access Control and Randomised Access Control (referred to here as the HAC and RAC protocols, respectively) in 2003. The schemes control access to a tag by locking or unlocking the tag using a one-way hash function h.

In the HAC scheme, a server S stores values ID_i, k_i and HID_i for each tag T_i, where ID_i is an identifier for T_i, k_i is a secret key and $HID_i = h(k_i)$. Each tag T_i stores ID_i and HID_i. When a tag T_i is locked, it responds with HID_i to all queries. To unlock a tag, the server sends it k_i. If, for a received key k_i, $HID_i = h(k_i)$, the tag unlocks itself and replies with ID_i. Figure 5.1 summarises the protocol, where time flows from the top to the bottom, i.e. the top-most message is sent first and the bottom-most message is sent last. The other figures in this thesis use the same convention.

However, the scheme allows a tag to be tracked by an eavesdropper, because the static value HID_i is used repeatedly [79]. In addition, k_i and ID_i are transferred in clear text, and hence they can be obtained by an eavesdropper.

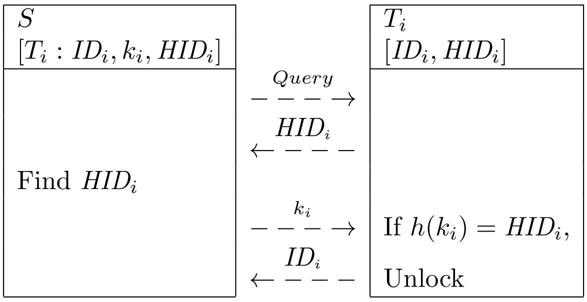


Figure 5.1: The HAC protocol

The second scheme, the RAC protocol, employs a random number generator as well as a hash function to prevent the tracking attack possible for the HAC protocol. A server stores ID_i for each tag T_i, and a tag T_i stores ID_i. A locked tag T_i generates a different response in every session by responding with the pair $(r, M_1 = h(ID_i \| r))$, for a randomly tag-generated r, whenever it is queried. A server then identifies the tag by performing an exhaustive search of all stored tag records; the server computes $h(ID_j \| r)$ for each stored ID_j in turn, until it finds a match to the value received from the tag. To unlock a tag, the server sends it the value ID_i. Figure 5.2 summarises the RAC protocol.

However, a tag response M_1 can be intercepted, and knowledge of M_1 allows replay and tag impersonation attacks. In an attempt to avoid such problems, Weis et al. [79] suggested that $ID_i \oplus f_{k_i}(r)$ or $(ID_i \| h(ID_i)) \oplus f_{k_i}(r)$ could be used instead of $h(ID_i \| r)$, using a secret key k_i shared between a server and a tag T_i. In addition, the use of a fixed identifier ID_i can give rise to traceability threats.

The properties of these protocols are summarised in Tables 5.1 and 5.2.

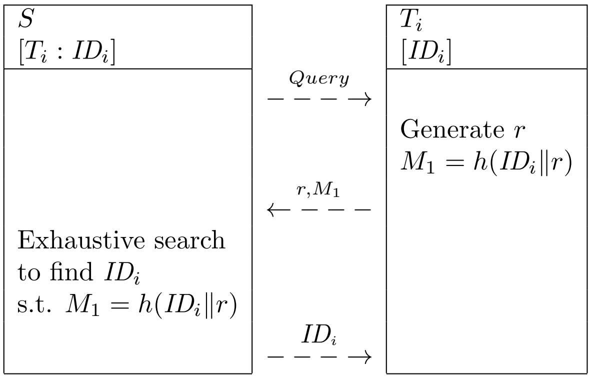


Figure 5.2: The RAC protocol

5.3 The Ohkubo-Suzki-Kinoshita Protocol

In 2003, Ohkubo, Suzki and Kinoshita [49] proposed an RFID privacy protection scheme (referred to here as the OSK protocol) designed to protect against tag tracking and backward traceability. It involves updating the tag identifier whenever a tag is queried, using a low-cost hash-chain mechanism.

The server stores a secret s_i^0 and an identifier ID_i for each tag, and a tag initially stores the secret information s_i as s_i^0. In a session, the tag sends $M_1 = g(s_i)$ to the server, and renews its secret s_i to $h(s_i)$, where g and h are hash functions. The server then identifies the tag via an exhaustive search, computing $g(h^j(s_i^0))$ for each tag until it finds a match with the received value M_1, where h^j denotes j iterations of the function h. Figure 5.3 summarises the OSK protocol.

This protocol is subject to replay attacks, and hence an eavesdropper can impersonate a target tag without knowing the tag secrets. In addition, the scheme is not scalable, because a server needs to perform $O(n)$ work to identity a tag from amongst a population of n tags (see section 4.2). The properties of this protocol are summarised in Tables 5.1 and 5.2.

Avoine, Dysli and Oechslin [3] proposed a modification to this scheme to prevent

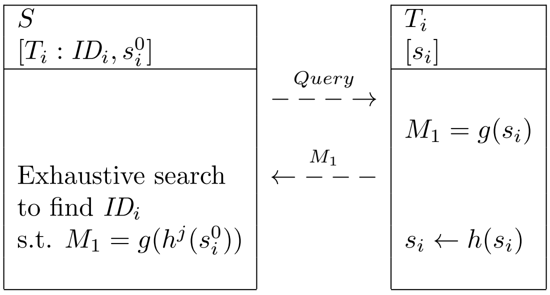


Figure 5.3: The OSK protocol

replay attacks. In the revised version, a server sends a random challenge r to a tag, and the queried tag replies with $M_1 = g(s_i \oplus r)$. As a result, an adversary cannot reuse a tag reply M_1 to respond to a different challenge r.

Avoine and Oechslin [4] further modified the protocol to introduce a time-memory trade-off designed to reduce the server's tag identification workload, thereby improving its scalability. The main drawback of this modified scheme is the need for large server storage, because precomputation and storage of look-up tables are required [3, 4].

5.4 The Henrici-Müller Protocol

Henrici and Müller [25] proposed an RFID authentication scheme (referred to here as the HM protocol) in 2004, designed to enhance location privacy. It uses a one-way hash function h and a binary operation $\circ$ on bit strings (a simple exclusive-or function is adequate for this purpose [25]).

A server must store a table containing the following entries for each tag T_i: the hash of the current tag identifier HID_i that acts as the primary index for the table, the current tag identifier ID_i, a transaction number TID_i, the number of the last successful transaction LST_i, and the stored entry for the associated tag AE_i. A tag T_i stores the values of ID_i, TID_i and LST_i. When queried, a tag responds with $M_1 = h(ID_i)$, $M_2 = h(ID_i \circ TID_i)$, and $\delta = TID_i - LST_i$. The server then finds an ID_i that matches the received values of M_1, M_2 and δ. If the server identifies the tag T_i, it generates a random number r and sends $M_3 = h(r \circ TID_i' \circ ID_i)$ back to the tag, where $TID_i' = LST_i + \delta$ and $+$ is regular integer addition. If the authentication process completes correctly, the tag and the server will update their stored copies of ID_i to the value $ID_i \circ r$. Figure 5.4 summarises the HM protocol.

This scheme still allows a degree of tag tracking, because a tag always replies with the same hashed ID_i before the next successful authentication [10]. Also, it does not provide backward untraceability, because an attacker that compromises a tag could easily compute the identifiers used in previous sessions. The properties of the protocol are summarised in Tables 5.1 and 5.2.

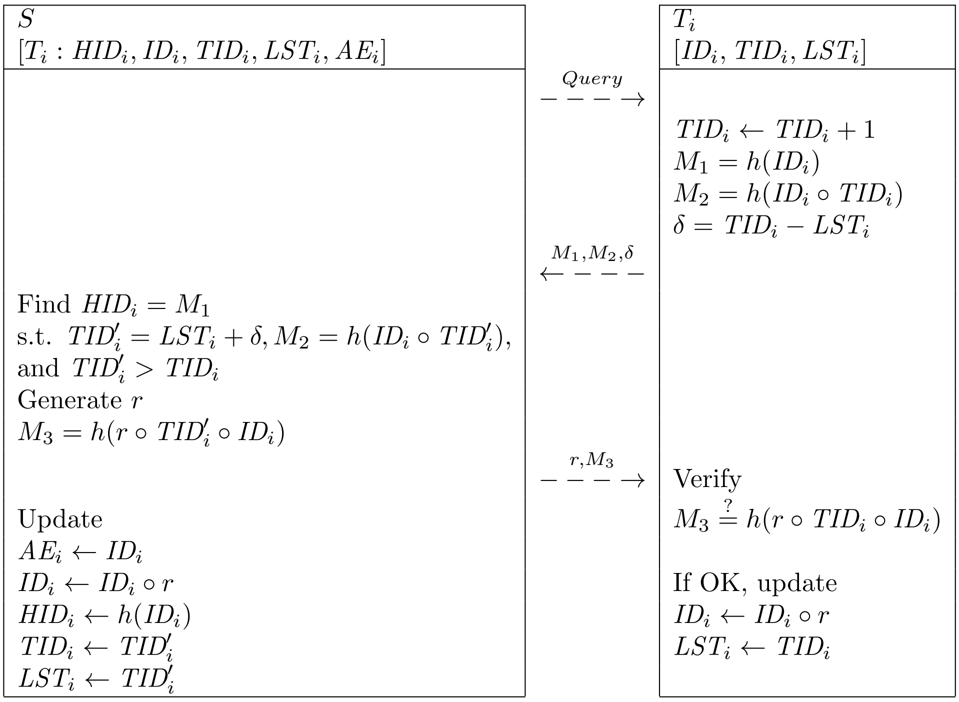

S
$[T_i : HID_i, ID_i, TID_i, LST_i, AE_i]$
T_i
$[ID_i, TID_i, LST_i]$
$\xrightarrow{Query}$
$TID_i \leftarrow TID_i + 1$
$M_1 = h(ID_i)$
$M_2 = h(ID_i \circ TID_i)$
$\delta = TID_i - LST_i$
$\xleftarrow{M_1, M_2, \delta}$
Find $HID_i = M_1$
s.t. $TID'_i = LST_i + \delta, M_2 = h(ID_i \circ TID'_i)$,
and $TID'_i > TID_i$
Generate r
$M_3 = h(r \circ TID'_i \circ ID_i)$
$\xrightarrow{r, M_3}$
Verify
$M_3 \overset{?}{=} h(r \circ TID_i \circ ID_i)$
Update
$AE_i \leftarrow ID_i$
$ID_i \leftarrow ID_i \circ r$
$HID_i \leftarrow h(ID_i)$
$TID_i \leftarrow TID'_i$
$LST_i \leftarrow TID'_i$
If OK, update
$ID_i \leftarrow ID_i \circ r$
$LST_i \leftarrow TID_i$

Figure 5.4: The HM protocol

5.5 The Molnar-Wagner Protocol

Molnar and Wagner [46] proposed a mutual authentication scheme (referred to here as the MW protocol) to provide privacy for library RFID systems. The scheme uses a pseudo-random number function to protect the messages communicated between tag and server.

In their basic authentication protocol, a server S and a tag T_i share a shared secret k_i, that is used as a key for a pseudo-random number function f. The server queries a tag by sending it a random number r_1. The tag generates a random number r_2, computes $M_1 = ID_i \oplus f_{k_i}(0\|r_1\|r_2)$, and sends them both to the server. The server finds the value ID_i for the tag using the received values of r_2 and M_1, and sends $M_2 = ID_i \oplus f_{k_i}(1\|r_1\|r_2)$ back to the tag to complete server authentication. Figure 5.5 summarises the protocol.

One feature of this scheme is its use of a tree-based technique that reduces tag identification complexity from $O(n)$ to $O(\log n)$ (see section 4.2). The n tags are considered as leaves in a balanced binary tree, and each edge in the tree is associated with a secret. A server knows all the secrets, and each tag stores the $\log n$ secrets corresponding to the path from the root to the tag. Thus, the work for the server to identify a tag has magnitude $O(\log n)$.

However, this scheme could degrade privacy if an adversary tampers with a tag, because in such a case the adversary is able to trace other tags in a probabilistic way, as described in [3]. Also, this scheme uses a fixed key k_i for each tag T_i, and hence it cannot resist backward traceability; once a tag is compromised, the attacker can trace past communications related to the tag. The properties of the protocol are summarised in Tables 5.1 and 5.2.

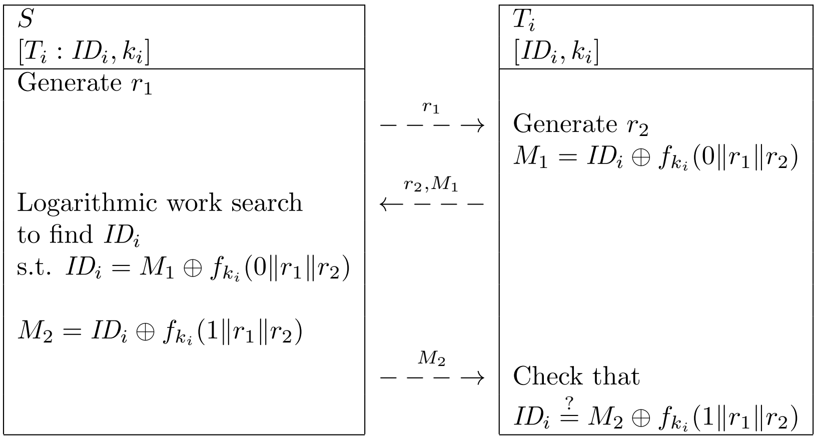
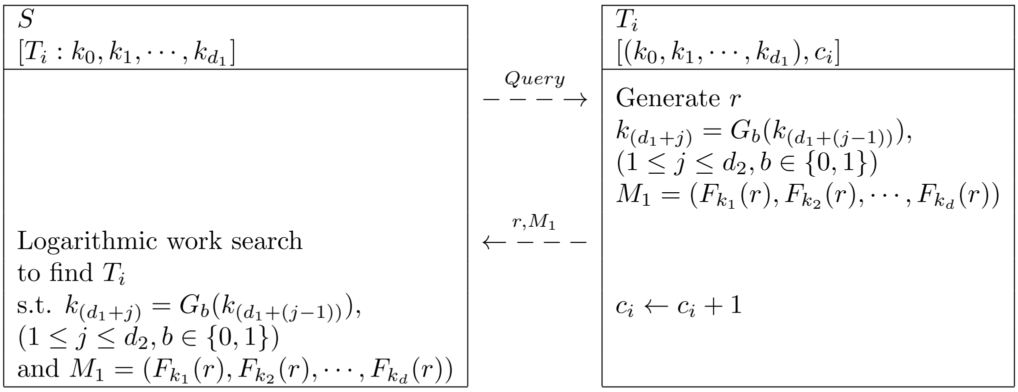

S
$[T_i : ID_i, k_i]$
Generate r_1
r_1
Logarithmic work search to find ID_i s.t. $ID_i = M_1 \oplus f_{k_i}(0\|r_1\|r_2)$
$M_2 = ID_i \oplus f_{k_i}(1\|r_1\|r_2)$
M_2
T_i
$[ID_i, k_i]$
Generate r_2
$M_1 = ID_i \oplus f_{k_i}(0\|r_1\|r_2)$
r_2, M_1
Check that
$ID_i \overset{?}{=} M_2 \oplus f_{k_i}(1\|r_1\|r_2)$

Figure 5.5: The MW protocol

5.6 The Molnar-Soppera-Wagner Protocol

In 2005, Molnar, Soppera and Wagner [45] proposed an RFID pseudonym protocol (referred as here the MSW protocol) that employs pseudo-random number functions. The scheme uses a tree of secrets of depth $d = d_1 + d_2$, like the MW scheme. Each node in the tree has an associated l-bit secret key. The first d_1 levels of the tree contain node secrets that are chosen uniformly and independently at random by the trusted centre during system initialisation using a function H, where $H(s)$ denotes the key associated with node s in the tree. Each node at depth d_1 corresponds to a unique tag. When a tag is enrolled into the system, it receives all keys on the path from its node to the root. Thus, each tag needs the capacity to store d_1 secrets. The next d_2 levels of the tree contain secrets that are derived using a pseudo-random generator G.

When a tag is queried, it generates a random number r, computes the key values $k_{(d_1+j)} = G_b(k_{(d_1+(j-1))})$, where $j = 1, 2, \cdots, d_2$ and $b \in \{0, 1\}$, and responds with a pseudonym $M_1 = (F_{k_1}(r), F_{k_2}(r), \cdots, F_{k_d}(r))$, where F is a pseudo-random number function and k_j $(j = 1, 2, \cdots, d)$ represent the secrets along the path in the tree of secrets from the root to the tag's current leaf. The tag then increments the counter c_i. Figure 5.6 summarises the MSW protocol.



S
$[T_i : k_0, k_1, \cdots, k_{d_1}]$

T_i
$[(k_0, k_1, \cdots, k_{d_1}), c_i]$

$\xrightarrow{Query}$

Generate r
$k_{(d_1+j)} = G_b(k_{(d_1+(j-1))})$,
$(1 \leq j \leq d_2, b \in \{0, 1\})$
$M_1 = (F_{k_1}(r), F_{k_2}(r), \cdots, F_{k_d}(r))$

$\xleftarrow{r, M_1}$

Logarithmic work search
to find T_i
s.t. $k_{(d_1+j)} = G_b(k_{(d_1+(j-1))})$,
$(1 \leq j \leq d_2, b \in \{0, 1\})$
and $M_1 = (F_{k_1}(r), F_{k_2}(r), \cdots, F_{k_d}(r))$

$c_i \leftarrow c_i + 1$

Figure 5.6: The MSW protocol

The scheme takes $O(\log n)$ work to identify a tag because of its use of a tree of secrets (see section 4.2). It further allows for time-limited delegation. However, the ownership transfer procedure of this scheme is rather restrictive, in the sense that the old and new owners must trust the same Trusted Centre (TC), and the

TC's database controls all the secret tag information. A reader that has received partial information from the TC can read the tag only a limited number of times without on-line connectivity to the TC. Thus, the scheme more closely resembles a time-limited access delegation scheme than a system for ownership transfer [41].

The scheme only provides tag identification, not mutual authentication. It also allows replay attacks. The tree-based secrets approach requires that each tag stores the $\lceil \log n \rceil$ secrets corresponding to the path from the root to the tag and performs a number of pseudo-random number function computations to generate its pseudonym. In addition, if a tag secret is compromised, then the attacker can compute the secrets for every descendent in the subtree rooted at that tag node. That is, the more tags an adversary tampers with, the greater the privacy breaches, as for the MW scheme [3, 54]. The properties of the protocol are summarised in Tables 5.1 and 5.2.

5.7 The Saito-Imamoto-Sakurai Protocols

Saito et al. [59] proposed two approaches for reassigning an RFID tag's key for ownership transfer. The schemes protect the privacy of the new owner from the old owner by updating the tag key using a symmetric cryptosystem. As these schemes are only a process for key change, they need to be combined with an RFID authentication scheme to form a complete RFID system. The first scheme (referred to here as the SIS1 scheme) employs a three-party model using a Trusted Third Party (TTP). The current owner server (S_{old}) first passes the secret key k_i used to encrypt the tag identifier ID_i to the new owner server (S_{new}). The new owner server then generates a new encryption key k_i', and asks the TTP to send both keys k_i and k_i' to the tag, after encrypting them, so that the tag can update its identifier using the new key k_i'. Figure 5.7 summarises the protocol.

However, this scheme has usability and security issues, because the new owner must communicate with a TTP [59], and, if a tag is compromised, the secret key shared between the TTP and the tags will be exposed.

The second scheme (referred to here as the SIS2 scheme) uses a two-party model. It is based on the premise that the backward channel (i.e. the communication channel from a tag to a reader) is more secure against eavesdropping than the forward channel

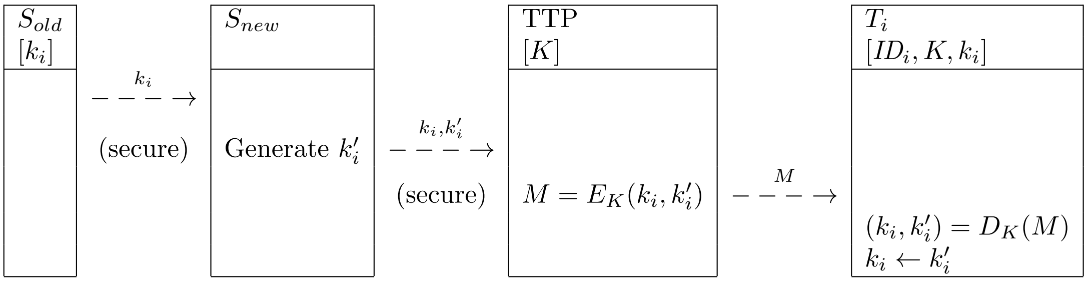


Figure 5.7: The SIS1 protocol

(i.e. the communication channel from a reader to a tag), because the signal strength in the backward channel is weaker than one in the forward channel. In this scheme, the new owner server (S_{new}) of a tag encrypts both the old encryption key k_i provided by the current owner server (S_{old}) and the new key k'_i which the new owner server has created, using a nonce n received from the tag over the backward channel. It sends $M = E_n(k_i, k'_i)$ back to the tag, where $E_x(y, z)$ denotes the encrypted version of a plaintext (y, z) computed using a symmetric encryption scheme, keyed by x. The tag obtains k'_i by decrypting M, and updates k_i to k'_i. Figure 5.8 summarises the protocol.

However, the assumption that intercepting the nonce sent from the tag is difficult is questionable, because, as pointed out in [59], an attacker could be located close to the tag and thereby successfully eavesdrop on the backward channel, despite its short range. Also, a tag always stores a fixed identifier, and hence the tag's past interactions could be traced if the tag is compromised.

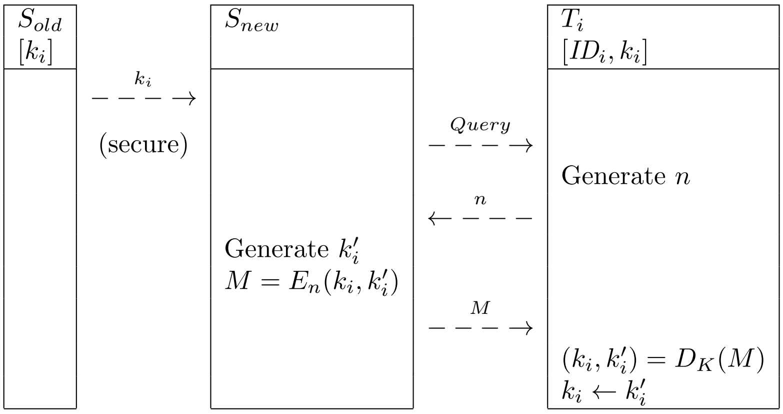


Figure 5.8: The SIS2 protocol

5.8 The Dimitriou Protocol

Dimitriou [13] proposed an RFID authentication protocol (referred to here as the D protocol) in 2005, designed to enforce user privacy and protect against tag cloning. A tag T_i stores its identifier ID_i, and a server S stores the identifier ID_i and a hash of the identifier HID_i for each tag T_i, where HID_i serves as the primary key used to identify information related to the tag. This scheme makes use of a challenge-response approach and employs a hash function h and a keyed hash function f. A server queries a tag by sending it a random number r_1, and a tag responds with a random number r_2, a hash of its identifier $M_1 = h(ID_i)$, and a keyed hash of the random numbers $M_2 = f_{ID_i}(r_2 \| r_1)$.

The scheme maintains scalability in the sense that the server can find the value HID_i corresponding to the received value of M_1, without an exhaustive search. If the server finds a matching value HID_i, it checks that the received value of M_2 equals $f_{ID_i}(r_2 \| r_1)$. If the validation is successful, the server authenticates the tag and updates its identifier ID_i to $g(ID_i)$, where g is a one-way function. The server then sends a message $M_3 = f_{ID_i}(r_2 \| r_1)$ using the updated identifier to the tag. The tag authenticates the server by checking the received value of M_3. If the check is successful, the copy of the identifier ID_i held by the tag is also updated. The protocol is summarised in Figure 5.9.

However, the tag identifier remains the same between valid sessions, thereby making the scheme vulnerable to tracking. Additionally, the scheme is prone to DoS attacks [13]; if the message M_3 does not reach the tag in a session, the server will update the tag identifier but the tag will not. The properties of the protocol are summarised in Tables 5.1 and 5.2.

5.9 The Duc-Park-Lee-Kim Protocol

Duc et al. [14] proposed an authentication protocol (referred to here as the DPLK protocol) for EPCglobal Class-1 Gen-2 RFID tags [16] in 2006. It uses simple cryptographic primitives such as a pseudo-random number generator and a cyclic redundancy code.

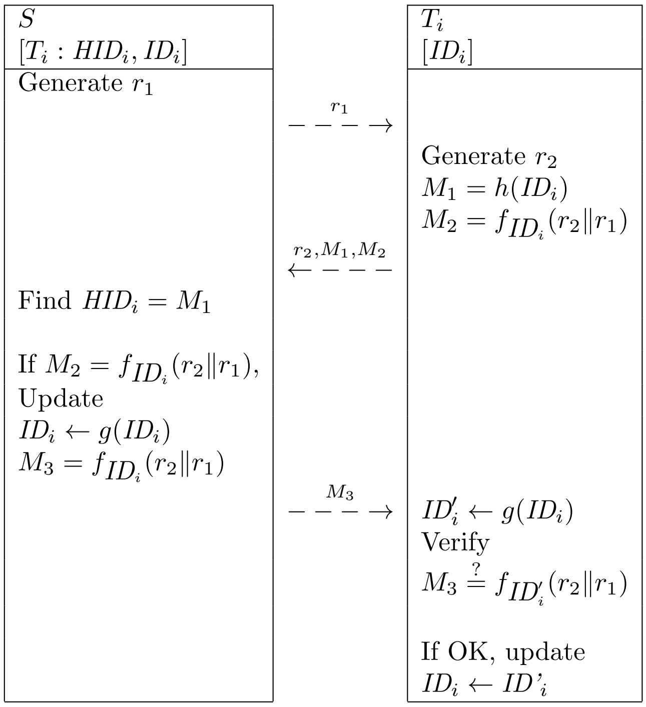

S
$[T_i : HID_i, ID_i]$
Generate r_1
r_1
Find $HID_i = M_1$
If $M_2 = f_{ID_i}(r_2\|r_1)$,
Update
$ID_i \leftarrow g(ID_i)$
$M_3 = f_{ID_i}(r_2\|r_1)$
M_3
T_i
$[ID_i]$
Generate r_2
$M_1 = h(ID_i)$
$M_2 = f_{ID_i}(r_2\|r_1)$
r_2, M_1, M_2
$ID'_i \leftarrow g(ID_i)$
Verify
$M_3 \overset{?}{=} f_{ID'_i}(r_2\|r_1)$
If OK, update
$ID_i \leftarrow ID'_i$

Figure 5.9: The D protocol

A tag T_i is set up by assigning it an Electronic Product Code EPC_i, a session key k_i and a long-term secret p_i. A server stores the values of EPC_i, k_i and p_i for each tag. We use f and h respectively to denote the pseudo-random number generator and cyclic redundancy code used by the scheme. When queried, a tag generates a random number r and responds with $M_1 = h(EPC_i\|r) \oplus k_i$ and $M_2 = h(M_1 \oplus r)$. The server then performs an exhaustive search to find a stored EPC_i for which $M_1 \oplus k_i = h(EPC_i\|r)$. If the server finds a matching value, it sends $M_3 = h(EPC_i\|p_i\|r) \oplus k_i$ to the tag. If a session ends successfully, the copies of the session key k_i held by the tag and server are updated using f. The protocol is summarised in Figure 5.10. The security of the scheme can be improved by employing cryptographic hash functions instead of h and f.

The scheme cannot prevent replay attacks before the next successful authentication, because M_1 and M_2 can be reused by an attacker to impersonate the tag. Most seriously, a DoS attack could permanently desynchronise a server and a tag [10]. The scheme also does not provide backward untraceability because EPC_i and p_i are fixed [10]. The properties of the protocol are summarised in Tables 5.1 and 5.2.

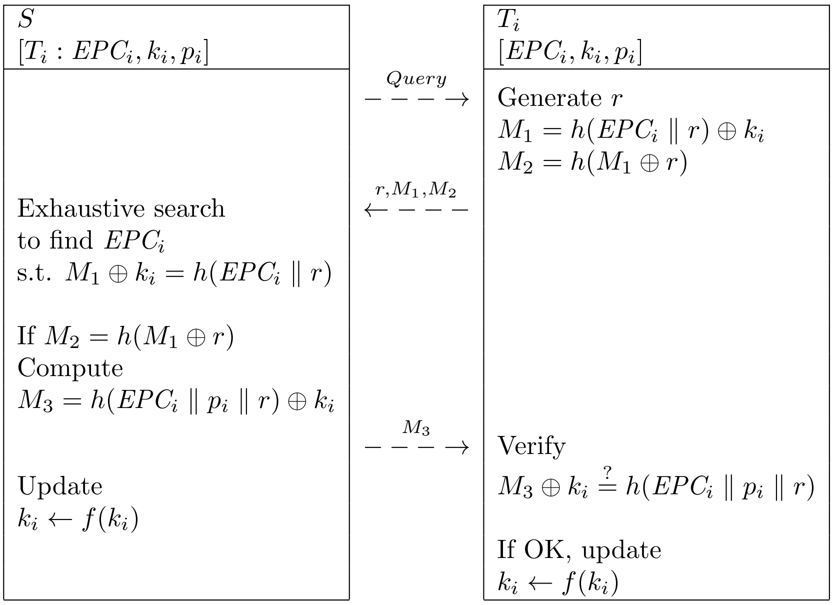

S
$[T_i : EPC_i, k_i, p_i]$
T_i
$[EPC_i, k_i, p_i]$
$Query$
Generate r
$M_1 = h(EPC_i \parallel r) \oplus k_i$
$M_2 = h(M_1 \oplus r)$
r, M_1, M_2
Exhaustive search
to find EPC_i
s.t. $M_1 \oplus k_i = h(EPC_i \parallel r)$
If $M_2 = h(M_1 \oplus r)$
Compute
$M_3 = h(EPC_i \parallel p_i \parallel r) \oplus k_i$
M_3
Verify
$M_3 \oplus k_i \overset{?}{=} h(EPC_i \parallel p_i \parallel r)$
Update
$k_i \leftarrow f(k_i)$
If OK, update
$k_i \leftarrow f(k_i)$

Figure 5.10: The DPLK protocol

5.10 The Lim-Kwon Protocol

The RFID authentication protocol (referred to here as the LK protocol) proposed by Lim and Kim [41] is a challenge-response protocol employing pseudo-random functions. The LK scheme uses a forward key chain for tag secret evolution and a backward key chain (used in reverse order) for server validation.

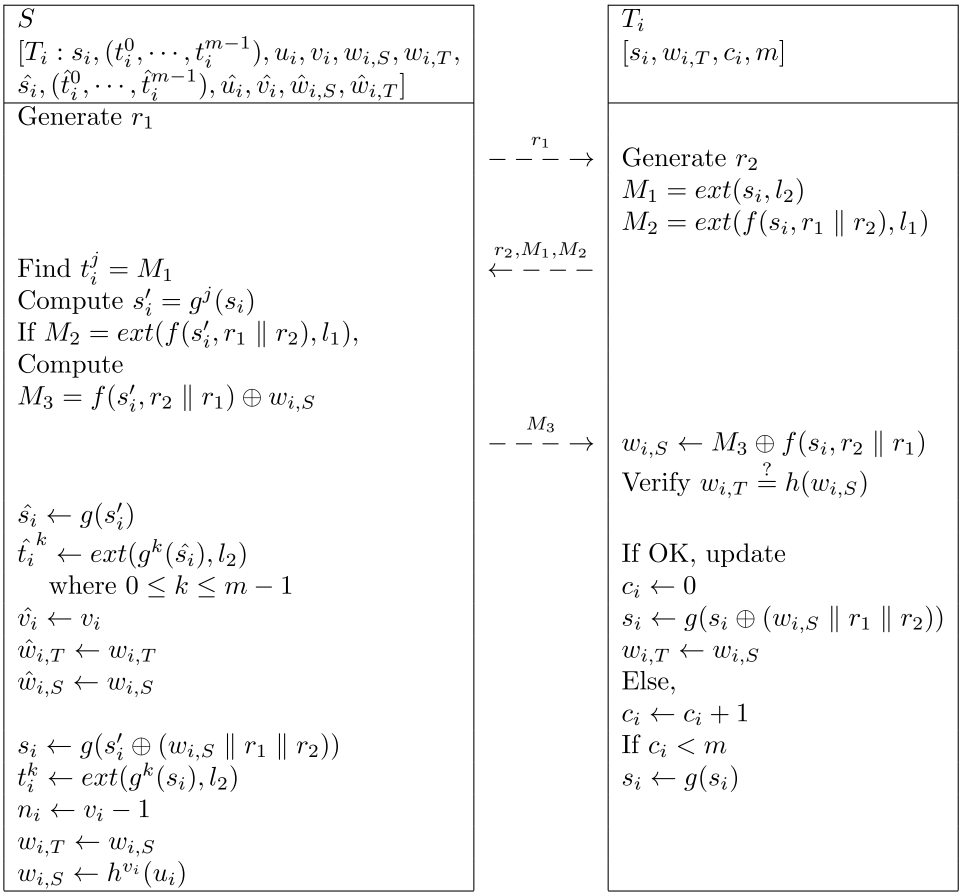
This scheme makes use of three pseudo-random functions, $f : \{0,1\}^l \times \{0,1\}^{2l_1} \rightarrow \{0,1\}^{2l_1}$, $g : \{0,1\}^l \rightarrow \{0,1\}^l$, and $h : \{0,1\}^{2l_1} \rightarrow \{0,1\}^{2l_1}$ where l is the bit-length of a tag secret, and l_1 is the bit-length of random challenges and responses. A server stores two sets of data for each tag T_i, namely the current data set and the most recent old data set (initially empty). The current data set contains the following tag identification data: a random secret s_i, m identifiers ${t_i}^j = ext(g^j(s_i), l_2)$ for $0 \leq j \leq m-1$, a random number u_i for a backward key chain, the length v_i of the backward key chain, and two secrets for server validation $w_{i,S} = h^{v_i - 1}(u_i)$ and $w_{i,T} = h(w_{i,S})$, where m is the maximum number of allowable authentication failures between two valid sessions, $ext(x, l)$ denotes a simple extract function returning l bits out of x, g^j denotes j iterations of the function g, and l_2 is the bit-length of a tag secret sent by the tag to help the back-end server to identify it. The tag stores the tag secret s_i, the server validator $w_{i,T}$, a failure counter c_i and the maximum number of the counter m, where c_i is initialised to 0.

The LK protocol refreshes the tag secret s_i within both the tag and the server whenever the authentication process completes successfully, using exchanged random numbers r_1 and r_2 and a secret for server validation $w_{i,S}$, i.e. $s_i \leftarrow g(s_i \oplus (w_{i,S} \| r_1 \| r_2))$. If an authentication session fails, only the tag updates its stored tag secret s_i to $g(s_i)$. The protocol is summarised in Figure 5.11.

For tag ownership transfer, the server of the new owner of a tag securely communicates with the server of the current owner, and receives all the relevant information for the tag. The new owner's server then communicates with the tag outside the reading range of the previous owner's server. As a result, the tag refreshes its secrets using random values shared only with the new server, and so no other party can communicate with the tag from this point onwards.

The protocol provides forward untraceability from the moment that an adversary misses just one successful authentication session after it has compromised the tag secret. The use of the backward hash key chain makes it difficult to impersonate a server to tags. Storing the most recent old data set protects against the desynchronisation problem arising from DoS attacks. The server can identify a tag in $O(1)$ work (see section 4.2).

However, the server must perform $O(m)$ operations for tag authentication, and must perform a significant number of pseudo-random function computations to update the tag secrets after a successful tag authentication. In addition, the server must store two key chains for each tag [41]. Also, an attacker can perform rapid-fire interrogation of a tag to increment its counter c_i to the maximum possible value. As a result, the tag secret s_i will remain static and will give the same response M_1 to every query until a valid session is performed, thereby allowing tag tracking [41]. Moreover, an attack introduced in [51] allows an adversary to trace a tag without compromising the tag. The properties of the protocol are summarised in Tables 5.1 and 5.2.



S

$[T_i : s_i, (t_i^0, \cdots, t_i^{m-1}), u_i, v_i, w_{i,S}, w_{i,T},$
$\hat{s_i}, (\hat{t}_i^0, \cdots, \hat{t}_i^{m-1}), \hat{u_i}, \hat{v_i}, \hat{w}_{i,S}, \hat{w}_{i,T}]$

Generate r_1

$\xrightarrow{r_1}$

T_i

$[s_i, w_{i,T}, c_i, m]$

Generate r_2
$M_1 = ext(s_i, l_2)$
$M_2 = ext(f(s_i, r_1 \parallel r_2), l_1)$

$\xleftarrow{r_2, M_1, M_2}$

Find $t_i^j = M_1$
Compute $s_i' = g^j(s_i)$
If $M_2 = ext(f(s_i', r_1 \parallel r_2), l_1)$,
Compute
$M_3 = f(s_i', r_2 \parallel r_1) \oplus w_{i,S}$

$\xrightarrow{M_3}$

$w_{i,S} \leftarrow M_3 \oplus f(s_i, r_2 \parallel r_1)$
Verify $w_{i,T} \overset{?}{=} h(w_{i,S})$

$\hat{s_i} \leftarrow g(s_i')$
$\hat{t_i}^k \leftarrow ext(g^k(\hat{s_i}), l_2)$
where $0 \leq k \leq m-1$
$\hat{v_i} \leftarrow v_i$
$\hat{w}_{i,T} \leftarrow w_{i,T}$
$\hat{w}_{i,S} \leftarrow w_{i,S}$

If OK, update
$c_i \leftarrow 0$
$s_i \leftarrow g(s_i \oplus (w_{i,S} \parallel r_1 \parallel r_2))$
$w_{i,T} \leftarrow w_{i,S}$
Else,
$c_i \leftarrow c_i + 1$
If $c_i < m$
$s_i \leftarrow g(s_i)$

$s_i \leftarrow g(s_i' \oplus (w_{i,S} \parallel r_1 \parallel r_2))$
$t_i^k \leftarrow ext(g^k(s_i), l_2)$
$n_i \leftarrow v_i - 1$
$w_{i,T} \leftarrow w_{i,S}$
$w_{i,S} \leftarrow h^{v_i}(u_i)$

Figure 5.11: The LK protocol

5.11 The Osaka-Takagi-Yamazaki-Takahashi Protocol

Osaka et al. [50] presented an RFID security method (referred to here as the OTYT protocol) that supports ownership transfer with high efficiency. The scheme uses a hash function h and a symmetric encryption scheme E. Each tag stores $e_i = E_{k_i}(ID_i)$ as its identifier, where k_i is a secret key shared by the tag and server and ID_i is a long term tag identifier (known only to the server). The server stores ID_i, k_i, and e_i for each tag T_i.

A server queries a tag by sending it a random number r, and the tag responds with $M_1 = h(e_i \oplus r)$. The server performs an exhaustive search to find a pair (e_i, k_i) for which $h(e_i \oplus r) = M_1$, and authenticates the tag by decrypting e_i using k_i.

To transfer ownership of a tag, the current owner generates a new key k', updates its identifier e to $e' = E_{k'}(ID_i)$ in order to protect its privacy from the new owner, and then sends k' to the new owner via a secure channel. The new owner then creates a new key k'', computes a new tag identifier $e'' = E_{k''}(ID_i)$ in order to protect its privacy from the previous owner, and sends $M_2 = e' \oplus e''$ to the tag so that the tag can obtain the new tag identifier e'' from the received value of M_2. Figure 5.12 summarises the protocol.

The scheme requires a relatively small amount of tag computation; however, it has a number of security vulnerabilities. The value of $e' \oplus e''$ is vulnerable to manipulation by an adversary, and the scheme does not prevent DoS attacks. In addition, if an attacker queries a tag twice using the same random number, then it receives the same response from the tag, enabling it to track the tag. Also, if a tag is compromised, an adversary is able to track past tag transactions, because an adversary that has M_2 for a previous session can compute a previous identifier by computing $M_2 \oplus e_i$. The properties of the protocol are summarised in Tables 5.1 and 5.2.

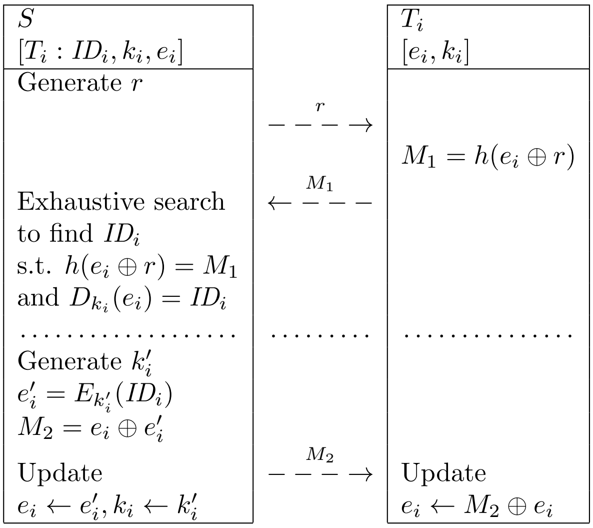


Figure 5.12: The OTYT protocol

5.12 The Chien-Chen Protocol

The RFID mutual authentication protocol proposed by Chien and Chen [10] in 2007 (referred to as the CC protocol) is based on the EPCglobal Class-1 Gen-2 RFID standard [16]. The protocol uses simple cryptographic primitives such as a pseudo-random number generator and a cyclic redundancy code.

The server stores five values for each tag T_i: an Electronic Product Code EPC_i, the new authentication key k_i, the new access key p_i, the most recent old authentication key $\hat{k_i}$, and the most recent old access key $\hat{p_i}$. A tag stores three values, namely EPC_i, k_i, and p_i. The values of k_i and p_i are updated after each successful authentication to give backward untraceability. We use f and h respectively to denote the pseudo-random number generator and cyclic redundancy code used by the scheme.

The server queries a tag by sending it a random number r_1. The tag generates a random number r_2, computes $M_1 = h(EPC_i \| r_1 \| r_2) \oplus k_i$ and sends r_1 and M_1 back to the server. The server performs an exhaustive search to find an EPC_i for which $M_1 \oplus k_i = h(EPC_i \| r_1 \| r_2)$. If such an EPC_i is found, the server computes $M_2 = h(EPC_i \| r_2) \oplus p_i$ and sends it to the tag. The server then updates k_i and p_i to $f(k_i)$ and $f(p_i)$, respectively. The tag verifies the received value of M_2. If the verification is successful, the tag also updates its keys k_i and p_i. Figure 5.13 summarises the protocol.

The scheme still permits a degree of backward traceability. Suppose that a strong attacker has compromised a tag and intercepted the values of r_1 and M_1 sent in the immediately previous session. The attacker can check that $f(M_1 \oplus h(EPC_i \| r_1 \| r_2))$ equals the compromised key k_i, and can thus determine the previous key $\hat{k_i}$, because EPC_i is fixed. In the same way, if a strong attacker has intercepted r_2 and M_2 in the immediately previous session, it can compute $\hat{p_i}$ by checking whether $f(M_2 \oplus h(EPC_i \| r_2))$ equals p_i. Moreover, Peris-Lopez et al. [52] show that the CC scheme permits location tracking, tag impersonation, server impersonation, and backward traceability, because of the linear properties of a cyclic redundancy code (CRC) that is used as a checksum algorithm. In addition, the use of a short length (16-bit) CRC allows desynchronisation by DoS attacks and does not guarantee unequivocal tag identification [52]. The properties of the protocol are summarised in Tables 5.1 and 5.2.

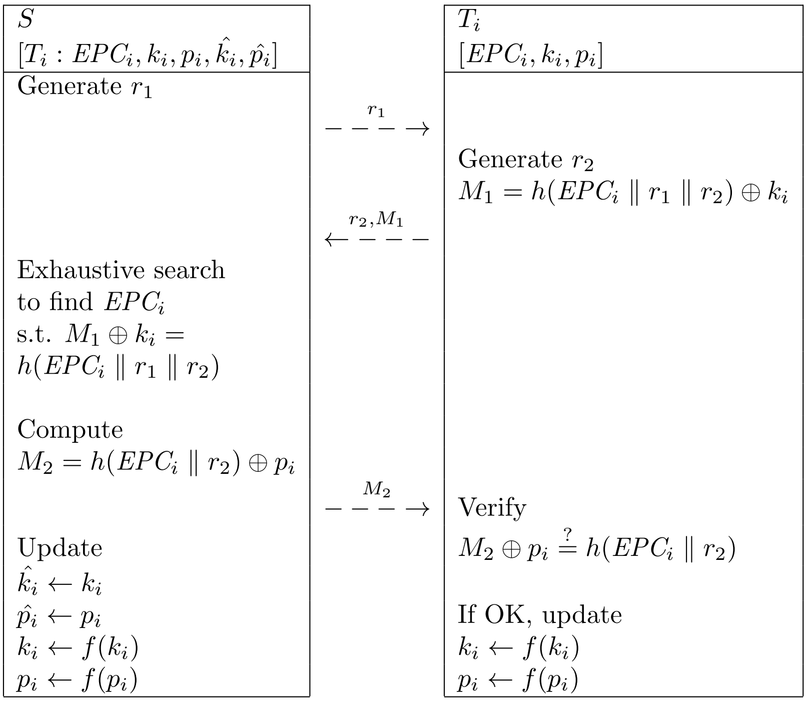

S
$[T_i : EPC_i, k_i, p_i, \hat{k_i}, \hat{p_i}]$
T_i
$[EPC_i, k_i, p_i]$
Generate r_1
$\xrightarrow{r_1}$
Generate r_2
$M_1 = h(EPC_i \parallel r_1 \parallel r_2) \oplus k_i$
$\xleftarrow{r_2, M_1}$
Exhaustive search to find EPC_i s.t. $M_1 \oplus k_i = h(EPC_i \parallel r_1 \parallel r_2)$
Compute $M_2 = h(EPC_i \parallel r_2) \oplus p_i$
$\xrightarrow{M_2}$
Verify $M_2 \oplus p_i \overset{?}{=} h(EPC_i \parallel r_2)$
Update
$\hat{k_i} \leftarrow k_i$
$\hat{p_i} \leftarrow p_i$
$k_i \leftarrow f(k_i)$
$p_i \leftarrow f(p_i)$
If OK, update
$k_i \leftarrow f(k_i)$
$p_i \leftarrow f(p_i)$

Figure 5.13: The CC protocol

5.13 The Peris-Lopez-Hernandez-Castro-Estevez-Tapiador-Ribagorda Protocol

Peris-Lopez et al. [53] proposed an RFID mutual authentication protocol (referred to here as PHER protocol) in 2007, obtained by modifying a scheme due to Shieh et al. [62]. The PHER scheme makes use of a secure one-way hash function h.

In the scheme, a server S stores the following entries for each tag T_i: the tag identifier ID_i, the tag password p_i, the server secret x_i for the tag, a random index-pseudonym t_i, and the most recent old value of t_i, denoted by $\hat{t}_i$, where t_i is used as the key to retrieve tag information. A tag T_i stores the values of t_i, a_i and p_i, where $a_i = h(ID_i \oplus x_i)$ is computed by the server in the tag registration phase.

For mutual authentication and index-pseudonym update, the scheme uses five exchanged messages. When a tag T_i is queried, it generates a random number r_1, computes $M_1 = h(r_1 \| t_i)$ and $M_2 = h(r_1 \| a_i)$, and sends r_1, M_1 and M_2 to the server. The server performs an exhaustive search until it finds a value t_i for which $h(r_1 \| t_i)$ equals M_1. If a match is found, the server computes $a_i = h(ID_i \oplus x_i)$ and checks that the received value of M_2 equals $h(r_1\|a_i)$. If the check is successful, the server generates a random number r_2, computes $M_3 = h(r_1\|r_2\|a_i)$ and sends r_2 and M_3 to the tag. The tag checks that the received value of $M_3 = h(r_1\|r_2\|a_i)$ and, if so, it sends $M_4 = h(r_2\|a_i + 1)$ to the server. If the received value of $M_4 = h(r_2\|a_i + 1)$, the server updates the tag index-pseudonym t_i to $h((r_1\|r_2) \oplus a_i \oplus t_i)$, and sends $M_5 = h((r_1 \oplus r_2)\|t_i)$ back to the tag to confirm the tag index-pseudonym update. When the tag receives M_5, it computes a new index-pseudonym $t'_i = h((r_1\|r_2) \oplus a_i \oplus t_i)$, and checks that the received value of $M_5 = h((r_1 \oplus r_2)\|t'_i)$. If the validation is successful, the tag also updates t_i to the new value t'_i. The protocol is summarised in Figure 5.14.

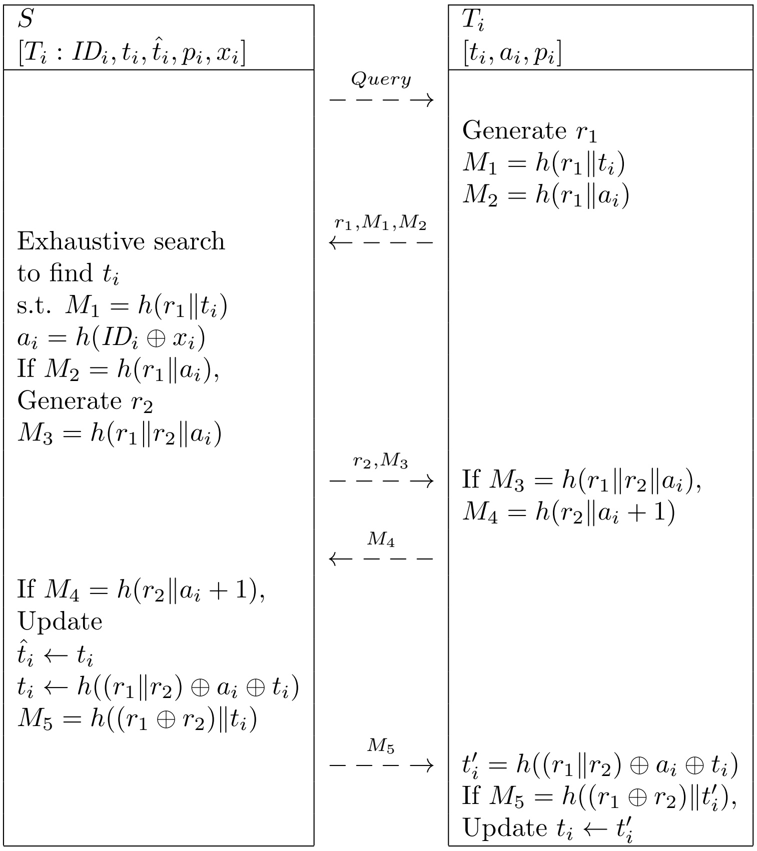

S
$[T_i : ID_i, t_i, \hat{t}_i, p_i, x_i]$
T_i
$[t_i, a_i, p_i]$
$Query$
Generate r_1
$M_1 = h(r_1\|t_i)$
$M_2 = h(r_1\|a_i)$
r_1, M_1, M_2
Exhaustive search
to find t_i
s.t. $M_1 = h(r_1\|t_i)$
$a_i = h(ID_i \oplus x_i)$
If $M_2 = h(r_1\|a_i)$,
Generate r_2
$M_3 = h(r_1\|r_2\|a_i)$
r_2, M_3
If $M_3 = h(r_1\|r_2\|a_i)$,
$M_4 = h(r_2\|a_i + 1)$
M_4
If $M_4 = h(r_2\|a_i + 1)$,
Update
$\hat{t}_i \leftarrow t_i$
$t_i \leftarrow h((r_1\|r_2) \oplus a_i \oplus t_i)$
$M_5 = h((r_1 \oplus r_2)\|t_i)$
M_5
$t'_i = h((r_1\|r_2) \oplus a_i \oplus t_i)$
If $M_5 = h((r_1 \oplus r_2)\|t'_i)$,
Update $t_i \leftarrow t'_i$

Figure 5.14: The PHER protocol

The scheme is rather complex in that it requires the transmission of five messages. As a result, a tag must perform six hash function computations in a session. In addition, the tag secret a_i is static, even if the tag updates t_i in every successful authentication session. Thus, backward traceability is possible. If a strong attacker intercepts all messages sent in a previous session, it can identify the values of M_2, M_3 and M_4 that relate to the compromised tag using knowledge of a_i. The properties of the protocol are summarised in Tables 5.1 and 5.2.

5.14 The Fouladgar-Afifi Protocols

Fouladgar and Afifi [20, 21] presented two protocols for RFID tag delegation and ownership transfer in 2007. In both schemes, a tag stores two secret keys k'_i and k_i and a counter c_i. The key k'_i is used to compute pseudonyms, and k_i is used to update both keys. The server stores the values of ID_i, k'_i and k_i for each tag. The first scheme [21] (referred to as the FA1 protocol) uses a cryptographic hash function h, and the second scheme (referred to as the FA2 protocol) uses a symmetric encryption function E.

In the FA1 scheme, a server queries a tag by sending it a random number r_1. If the tag's counter c_i is not the the maximum value c_{max}, it increments its counter and replies with a random number r_2 and $M_1 = h(r_2 \oplus k'_i)$. Otherwise, if $c_i = c_{max}$, the tag replies with r_2 and $M_1 = h(r_2 \oplus k_i)$. The server identifies the tag by exhaustively searching through its database for a key k'_i or k_i such that $M_1 = h(r_2 \oplus k'_i)$ or $M_1 = h(r_2 \oplus k_i)$. If $M_1 = h(r_2 \oplus k'_i)$, the session terminates. Otherwise, if $M_1 = h(r_2 \oplus k_i)$, the server then generates a random number δ, computes $M_2 = h(\delta) \| (\delta \oplus h(k_i))$ and sends M_2 to the tag. The server also updates k'_i and k_i to $h(k'_i \oplus \delta)$ and $h(k_i \oplus \delta)$, respectively. When the tag receives M_2, it obtains δ from M_2 and updates k'_i and k_i in the same way as the server. Figure 5.15 summarises the FA1 protocol.

The new owner queries a target tag using r_1, and receives a random number r_2 and a hash-based pseudonym $M_1 = h(r_2 \oplus k'_i)$ in return. The new owner then sends them to the current owner of the tag to request ownership transfer. The current owner then instructs the tag to set its counter to the maximum possible value. As a result, in the next session, the tag and the server of the new owner will update the copies of tag secrets k'_i and k_i, using a hash function and a random number generated by the tag.

Unfortunately, the scheme does not resist replay attacks, because a tag response is a function of only k'_i (or k_i) and r_2, and thus an eavesdropper can reuse the tag response to impersonate a tag. The tag keys k_i' and k_i are fixed until the tag counter c_i reaches c_{max}. This might give rise to traceability threats, if a tag is compromised. An attacker could also perform a rapid-fire interrogation DoS attack to increment c_i up to c_{max} [20].

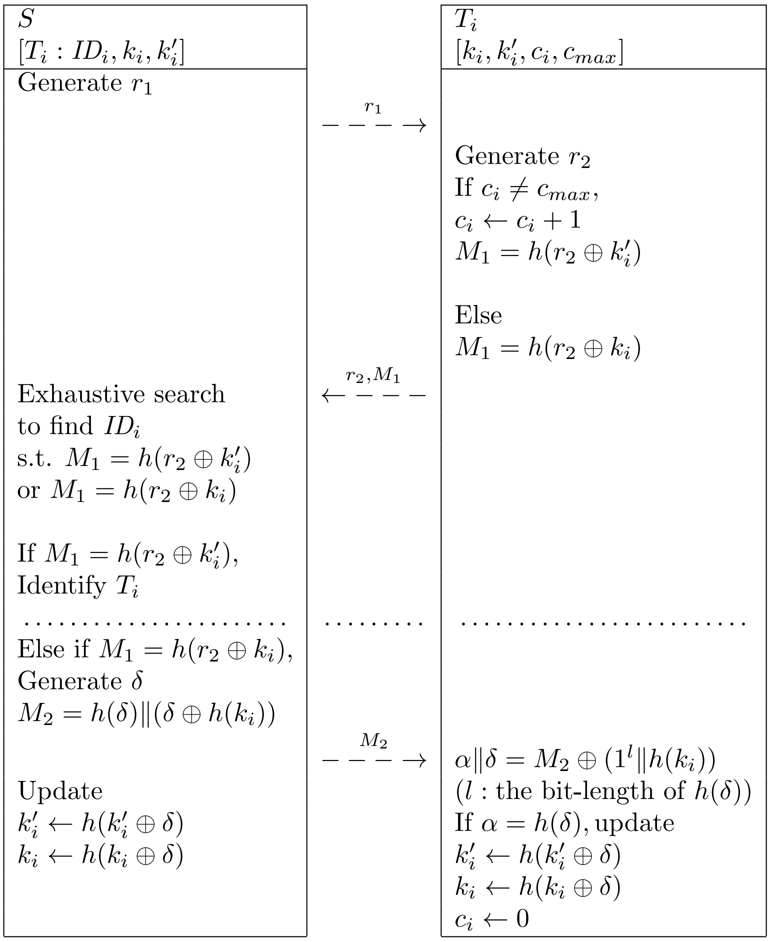

S
$[T_i : ID_i, k_i, k_i']$
Generate r_1
r_1
Exhaustive search
to find ID_i
s.t. $M_1 = h(r_2 \oplus k_i')$
or $M_1 = h(r_2 \oplus k_i)$
If $M_1 = h(r_2 \oplus k_i')$,
Identify T_i
Else if $M_1 = h(r_2 \oplus k_i)$,
Generate δ
$M_2 = h(\delta) \| (\delta \oplus h(k_i))$
Update
$k_i' \leftarrow h(k_i' \oplus \delta)$
$k_i \leftarrow h(k_i \oplus \delta)$
r_2, M_1
M_2
T_i
$[k_i, k_i', c_i, c_{max}]$
Generate r_2
If $c_i \neq c_{max}$,
$c_i \leftarrow c_i + 1$
$M_1 = h(r_2 \oplus k_i')$
Else
$M_1 = h(r_2 \oplus k_i)$
$\alpha \| \delta = M_2 \oplus (1^l \| h(k_i))$
$(l :$ the bit-length of $h(\delta))$
If $\alpha = h(\delta)$, update
$k_i' \leftarrow h(k_i' \oplus \delta)$
$k_i \leftarrow h(k_i \oplus \delta)$
$c_i \leftarrow 0$

Figure 5.15: The FA1 protocol

The FA2 scheme [20, 21] is similar to the FA1 scheme, except that it uses a symmetric encryption algorithm instead of a hash function. When a tag receives a query r_1, it replies with a random number r_2 and $M_1 = E_{k_i'}(r_1 \oplus r_2)$, if the tag counter c_i is not the maximum value c_{max}. Otherwise, if $c_i = c_{max}$, the tag replies with $M_1 = E_{k_i}(r_1 \oplus r_2)$. The server identifies the tag by exhaustively searching through its database for a key k_i' or k_i such that $M_1 = E_{k_i'}(r_1 \oplus r_2)$ or $M_1 = E_{k_i}(r_1 \oplus r_2)$. If $M_1 = E_{k_i'}(r_1 \oplus r_2)$, the session terminates. Otherwise, if $M_1 = E_{k_i}(r_1 \oplus r_2)$, the server then generates a random secret δ, computes $M_2 = E_{k_i}(r_2 \| \delta)$ and transfers

M_2 to the tag. The server then updates k_i' and k_i to $k_i' \oplus r_2 \oplus \delta$ and $k_i \oplus r_2 \oplus \delta$, respectively. When the tag receives M_2, it obtains δ by decrypting M_2, and then updates k_i' and k_i as $k_i' \leftarrow k_i' \oplus r_2 \oplus \delta$ and $k_i \leftarrow k_i \oplus r_2 \oplus \delta$. Figure 5.16 summarises the FA2 protocol.

For tag ownership transfer, the new owner queries a target tag using r_1, and receives a random number r_2 and a symmetric cryptography based pseudonym $M_1 = E_{k_i'}(r_1 \oplus r_2)$ from the tag, and then transmits them to the current owner asking for tag ownership to be transferred. Before transferring ownership, the server of the current owner generates a random number, and makes both its database and the tag update the tag secrets k_i' and k_i using random numbers generated by both the server and the tag, to protect its privacy from the new owner. It then passes the updated secrets to the new owner, along with other necessary information. To protect its privacy from the old owner of the tag, the new owner of the tag executes another session with the tag, thereby updating its keys. As a result, the old owner can no longer identify the tag. The new owner can also keep the key values transferred from the old owner if the tagged item has a warranty for after-sales service. However, the authors do not describe in detail how the tag can recover the old key kept by the old owner for after-sales service.

The FA2 scheme also allows replay attacks. Suppose an eavesdropper intercepts the values of r_1, r_2 and M_1 sent in a session. The eavesdropper can now impersonate the tag to the server in the following way. When the server sends r_1 to the eavesdropper (impersonating the tag), the eavesdropper can reply with $r_2' = r_1 \oplus r_2 \oplus r_1'$ and M_1. As in the FA1 scheme, if an adversary compromises a tag, then it may be able to trace the tag's past communications that used the same keys k_i' and k_i. Also, a counter exhaustion attack is possible, exactly as for the attack on FA1. The properties of the protocols are summarised in Tables 5.1 and 5.2.

5.15 The Tsudik Protocols

Tsudik [72] described an RFID identification protocol (referred to as the T1 protocol) that provides a basic level of tag identification using time-stamps. Tsudik [73] proposed two further schemes (referred to as the T2 and T3 schemes) that also provide tag authentication. The schemes use monotonically increasing time-stamps

S $[T_i : ID_i, k_i, k_i']$		T_i $[k_i, k_i', c_i, c_{max}]$
Generate r_1	$\overset{r_1}{- - - \rightarrow}$	
		Generate r_2 If $c_i \neq c_{max}$ $c_i \leftarrow c_i + 1$ $M_1 = E_{k_i'}(r_1 \oplus r_2)$ Else $M_1 = E_{k_i}(r_1 \oplus r_2)$
Exhaustive search to find ID_i s.t. $M_1 = E_{k_i'}(r_1 \oplus r_2)$ or $M_1 = E_{k_i}(r_1 \oplus r_2)$ If $M_1 = E_{k_i'}(r_1 \oplus r_2)$ Identify T_i	$\overset{r_2, M_1}{\leftarrow - - -}$	
....	....	....
Else if $M_1 = E_{k_i}(r_1 \oplus r_2)$ Generate δ $M_2 = E_{k_i}(r_2 \Vert \delta)$		
Update $k_i' \leftarrow k_i' \oplus r_2 \oplus \delta$ $k_i \leftarrow k_i \oplus r_2 \oplus \delta$	$\overset{M_2}{- - - \rightarrow}$	$r_2 \Vert \delta = D_{k_i}(M_2)$ Update $k_i' \leftarrow k_i' \oplus r_2 \oplus \delta$ $k_i \leftarrow k_i \oplus r_2 \oplus \delta$ $c_i \leftarrow 0$

Figure 5.16: The FA2 protocol

for tracking-resistant tag authentication, and employ a keyed hash function f.

In these three schemes, a tag stores k_i, t_i, and $t_{m,i}$, where k_i is a tag-specific secret, t_i is initially a time-stamp assigned to the tag at the time of manufacture, and $t_{m,i}$ is the highest possible time-stamp. The value k_i serves as both a tag identifier and a cryptographic key. Neither t_i nor $t_{m,i}$ need to be unique to a tag. A tag must also possess a uniquely seeded pseudo-random number generator.

In T1, a server maintains a periodically updated hash table in which each row contains ID_i, k_i, and $f_{k_i}(t_i')$ for a tag, where t_i' is the time-stamp for the server. The server first sends t_i' to a tag T_i. If $t_i' \leq t_i$ or $t_i' > t_{m,i}$, the tag generates and returns a pseudo-random number r. Otherwise, the tag updates its time stamp t_i to t_i', and replies with $M_1 = f_{k_i}(t_i)$. The server can identify the tag by finding M_1 in its look-up table. Figure 5.17 summarises the T1 protocol.

The T1 scheme only needs $O(1)$ operations to identify a tag, because a hash table is used for all look-ups (see section 4.2). However, the scheme merely identifies a tag and does not provide tag authentication. Additionally, the scheme is susceptible to a trivial DoS attack in which an attacker incapacitates a tag by sending it an inaccurate future time-stamp value [73]. Moreover, the scheme makes the assumption that a given tag is never identified (interrogated) more than once within the same time interval [73].

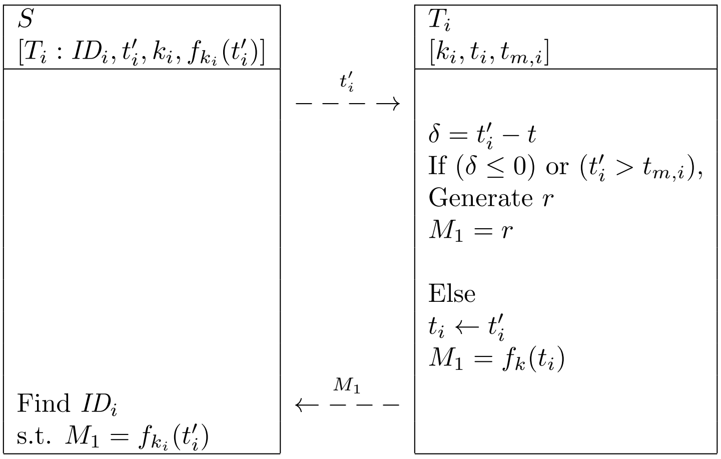


Figure 5.17: The T1 protocol

The T2 scheme adds tag authentication to the T1 scheme using a challenge-response method. When a tag T_i receives a query consisting of a time-stamp t'_i and a random number r_1, it checks that t'_i is a valid time-stamp. If the validation is successful, the tag updates t_i to t'_i, and computes $M_1 = f_{k_i}(t_i)$. Otherwise, the tag generates a pseudo-random number r_2 instead of computing M_1. The tag then generates a pseudo-random number r_3, computes $M_2 = f_{k_i}(r_3 \| r_1)$ and responds to the server with r_3, M_1 and M_2. The server identifies the tag by finding M_1 in its look-up table for the time-stamp t'_i, and authenticates the tag by checking that $M_2 = f_{k_i}(r_3 \| r_1)$. Figure 5.18 summarises the T2 protocol.

This scheme also takes constant time to identify and authenticate a tag because of its use of a look-up table. However, if a tag has been previously desynchronised by an attacker, it requires the server to perform $O(n)$ operations to authenticate the tag. The T2 scheme is also susceptible to DoS attacks, like the T1 scheme [73].

The T3 scheme mitigates the DoS vulnerability of the T1 and T2 schemes by using

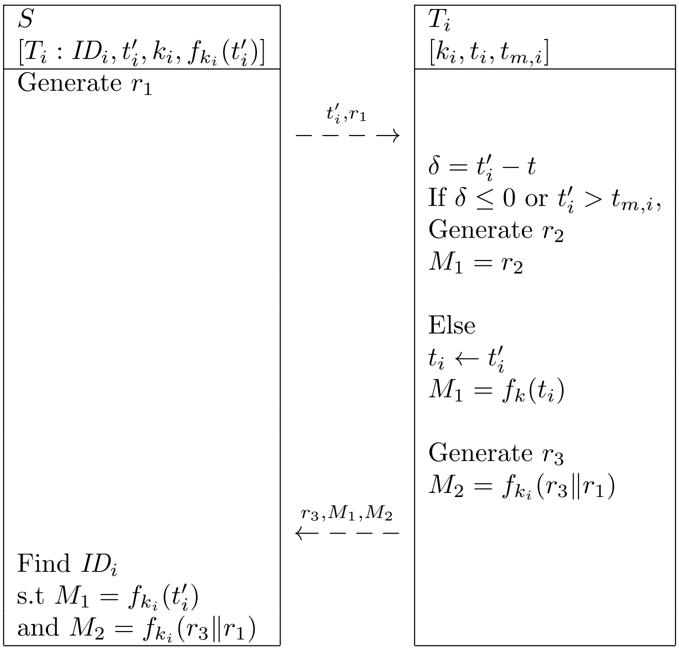


Figure 5.18: The T2 protocol

a hash-chain to generate a so-called epoch token, which allows a tag to ascertain that a time-stamp is not too far into the future.

A server generates a hash chain of length v by starting with an initial value (say, X) and repeatedly hashing it v times to produce a root $h^v(X)$, where $v = \lceil t_{m,i}/\omega \rceil$ and ω is the epoch duration, (e.g. one day). Each tag initially stores an epoch token z_i as a root of the hash chain, $h^v(X)$.

A server generates a random number r_1 and sends t'_i, r_1 and its epoch token z'_i to a tag. The tag checks the received values of t'_i and z'_i, by verifying that $t'_i \geq t_i$ and $t'_i < t_{m,i}$, and that $z'_i = h^{\gamma}(z'_i)$, where $\gamma = \lfloor t'_i/\omega \rfloor - \lfloor t/\omega \rfloor$. If the validations are successful, the tag updates t_i and z_i to t'_i and z'_i, respectively. The tag then computes $M_1 = f_{k_i}(t_i)$, generates r_2, computes $M_2 = f_{k_i}(r_2 \| r_1)$, and sends M_1, M_2, and r_2 as its reply. Otherwise, the tag generates pseudo-random numbers r_2, r_3 and r_4, and sends them instead. The server identifies the tag by finding M_1 in its look-up table for the time-stamp t'_i, and authenticates the tag by checking that $M_2 = f_{k_i}(r_2 \| r_1)$. Figure 5.19 summarises the T2 protocol.

The server only needs to perform $O(1)$ operations to identify and authenticate a tag, if the tag reply is valid. If not, the server takes $O(n)$ time to authenticate a tag. For T3, DoS attacks still remain a threat, because an adversary can incapacitate a tag for the epoch duration ω, if it queries the tag with the current epoch token and the maximum possible t'_i within the current epoch [73]. In addition, for both T2 and T3, the adversary can potentially distinguish between synchronised and desynchronised tags by timing the server responses, because a synchronised tag only requires a server to perform a quick table look-up, whereas a desynchronised tag requires it to perform an exhaustive search. Moreover, all three of the Tsudik schemes have backward traceability, because of their use of a fixed key k_i [73]. The properties of the protocols are summarised in Tables 5.1 and 5.2.

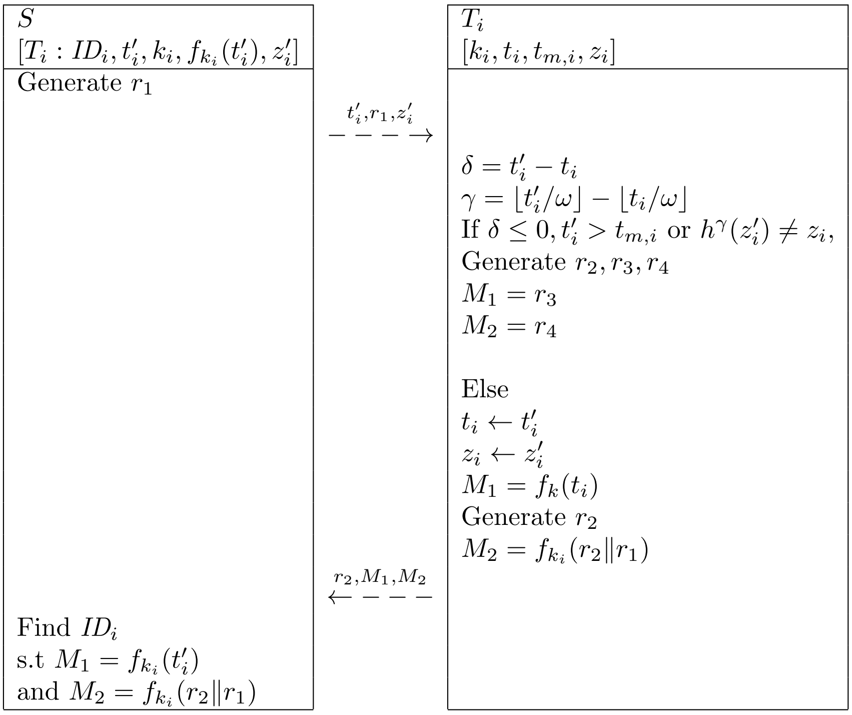

S
$[T_i : ID_i, t'_i, k_i, f_{k_i}(t'_i), z'_i]$
Generate r_1
t'_i, r_1, z'_i
Find ID_i
s.t $M_1 = f_{k_i}(t'_i)$
and $M_2 = f_{k_i}(r_2 \| r_1)$
T_i
$[k_i, t_i, t_{m,i}, z_i]$
$\delta = t'_i - t_i$
$\gamma = \lfloor t'_i/\omega \rfloor - \lfloor t_i/\omega \rfloor$
If $\delta \leq 0, t'_i > t_{m,i}$ or $h^{\gamma}(z'_i) \neq z_i$,
Generate r_2, r_3, r_4
$M_1 = r_3$
$M_2 = r_4$
Else
$t_i \leftarrow t'_i$
$z_i \leftarrow z'_i$
$M_1 = f_k(t_i)$
Generate r_2
$M_2 = f_{k_i}(r_2 \| r_1)$
r_2, M_1, M_2

Figure 5.19: The T3 protocol

5.16 The Burmester-de Medeiros-Motta Protocol

Burmester, de Medeiros and Motta [7] proposed an anonymous RFID authentication protocol (referred to here as the BMM protocol) in 2008. This protocol is based on a scheme originally proposed by van Le, Burmester and de Medeiros [76].

Each tag T_i is initially assigned a unique tuple (k_i, τ_i, q_i) of l-bit random values that is stored in non-volatile memory; k_i is its secret key, τ_i is a one-time pseudonym, and q_i is a seed for generating pseudonyms. The tag also has a boolean variable b_i and a cyclic counter c_i that takes m distinct values. The scheme uses an appropriate pseudo-random function g to generate values for pseudonyms, authenticators and confirmation. For each tag, the server S stores in a key-lookup database a tuple: $k_i, q_i, (\tau_i, q_i, q_i^1, \cdots q_i^m)$, where $q_i^j = g_{k_i}(q_i \| IV \| \text{ctr}(j))$, IV is an initial vector, and $\text{ctr}(j)$ is the jth value of the cyclic counter c_i described above. The server also stores the most recent old values of the tuple to maintain state coordination with the tag.

The state of tag T_i is controlled by the value of b_i; if the tag fails to receive a correct value of M_3 from the server, b_i is set to 1, otherwise $b_i = 0$. The server sends a random number r_1 to tag T_i. The tag replies with M_1 and M_2; if the stored value b_i of a tag is 0, the tag's pseudonym is $M_1 = \tau_i$, otherwise it computes $M_1 = g_{k_i}(q_i \| IV \| \text{ctr}(j))$. The tag also computes three values v_0, v_1 and v_2: $v_0 \| v_1 \| v_2 = g_{k_i}(M_1 \| r_1)$, where v_0 is kept for later use as a pseudonym update, v_1 is used for an authenticator, and v_2 is used for confirmation. The tag then replies to the server with M_1 and $M_2 = v_1$. The server uses its key-lookup database to identify the tag using the received value of M_1, and checks the validity of the received value of M_2 by computing $v_0' \| v_1' \| v_2' = g_{k_i}(M_1 \| r_1)$. If the validation is successful, the server sends $M_3 = v_2'$ to the tag, and updates the tuple for the tag. If the received value of $M_3 = v_2$ and the stored value of $b_i = 0$, then the tag updates its pseudonym τ_i to v_0. Otherwise, it updates its seed q_i to v_0. Figure 5.20 summarises the protocol.

A tag can be identified with constant-cost key lookup, thereby maintaining scalability for the back-end server (see section 4.2). However, the scheme does not provide backward untraceability and allows a degree of tag tracking [7]. The properties of the protocol are summarised in Tables 5.1 and 5.2.

5.17 Comparisons

We now compare the protocols introduced above against the privacy, security and performance requirements identified in chapter 4.

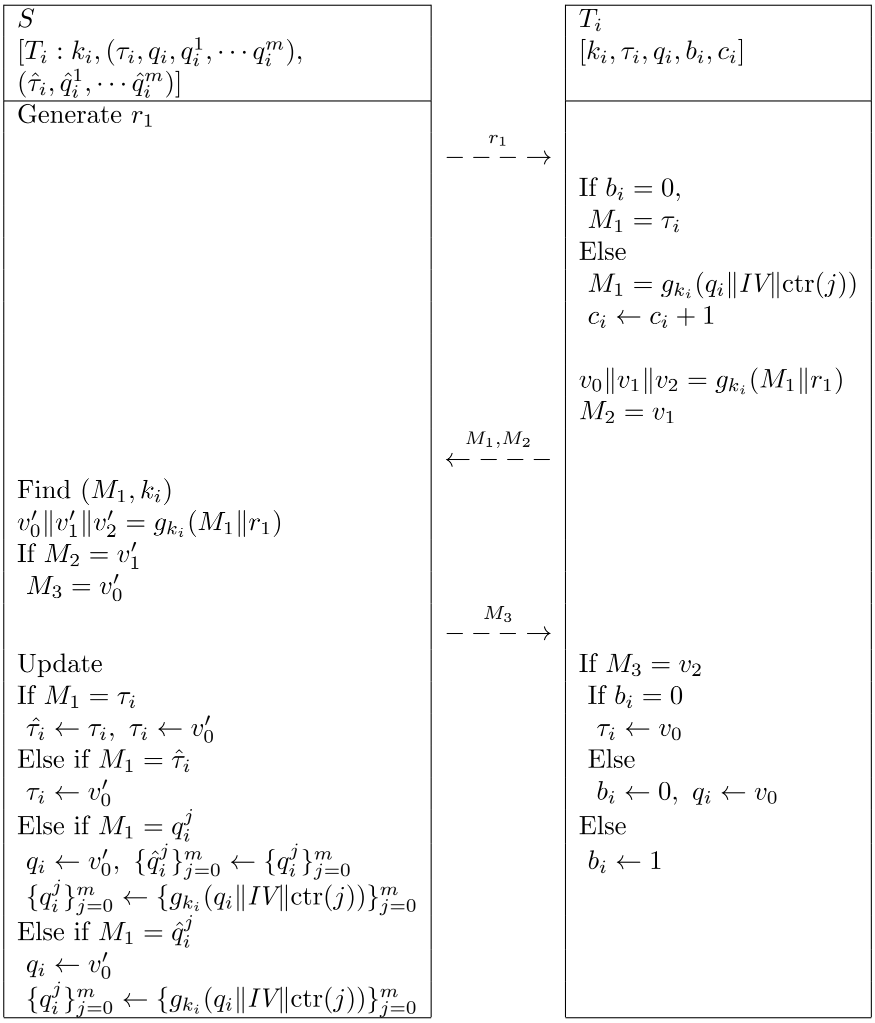

S
$[T_i : k_i, (\tau_i, q_i, q_i^1, \cdots q_i^m),$
$(\hat{\tau}_i, \hat{q}_i^1, \cdots \hat{q}_i^m)]$
T_i
$[k_i, \tau_i, q_i, b_i, c_i]$
Generate r_1
$\xrightarrow{r_1}$
If $b_i = 0$,
$M_1 = \tau_i$
Else
$M_1 = g_{k_i}(q_i \| IV \| \mathrm{ctr}(j))$
$c_i \leftarrow c_i + 1$
$v_0 \| v_1 \| v_2 = g_{k_i}(M_1 \| r_1)$
$M_2 = v_1$
$\xleftarrow{M_1, M_2}$
Find (M_1, k_i)
$v_0' \| v_1' \| v_2' = g_{k_i}(M_1 \| r_1)$
If $M_2 = v_1'$
$M_3 = v_0'$
$\xrightarrow{M_3}$
Update
If $M_1 = \tau_i$
$\hat{\tau}_i \leftarrow \tau_i, \ \tau_i \leftarrow v_0'$
Else if $M_1 = \hat{\tau}_i$
$\tau_i \leftarrow v_0'$
Else if $M_1 = q_i^j$
$q_i \leftarrow v_0', \ \{\hat{q}_i^j\}_{j=0}^m \leftarrow \{q_i^j\}_{j=0}^m$
$\{q_i^j\}_{j=0}^m \leftarrow \{g_{k_i}(q_i \| IV \| \mathrm{ctr}(j))\}_{j=0}^m$
Else if $M_1 = \hat{q}_i^j$
$q_i \leftarrow v_0'$
$\{q_i^j\}_{j=0}^m \leftarrow \{g_{k_i}(q_i \| IV \| \mathrm{ctr}(j))\}_{j=0}^m$
If $M_3 = v_2$
If $b_i = 0$
$\tau_i \leftarrow v_0$
Else
$b_i \leftarrow 0, \ q_i \leftarrow v_0$
Else
$b_i \leftarrow 1$

Figure 5.20: The BMM protocol

In Table 5.1, we give a comparison of the privacy and security properties of the protocols. The table indicates whether or not a protocol resists privacy violations (tag information leakage or tag location tracking), weak attacks (tag impersonation, replay attacks, man-in-the-middle attacks, or denial-of-service attacks), and strong attacks (backward traceability, forward traceability, or server impersonation). The table also indicates whether a protocol provides tag authentication and/or server authentication.

Table 5.2 gives the performance properties of the protocols. The protocols are compared with respect to the tag storage, tag computation, communication and scalability requirements identified in chapter 4. More specifically, the table indicates the type of secrets stored in a tag, the type and number of cryptographic function computations required in a tag (which are significantly more computationally complex computations than arithmetic and logical operations), the number of pseudo-random numbers that need to be generated in a tag, the number of exchanged messages, and the server complexity of identifying and authenticating a tag.

Tables 5.1 and 5.2 enable the strengths and weaknesses of the protocols to be readily assessed. For example, the LK and PHER schemes have many desirable privacy and security properties, but require significant cryptographic function computations in a tag and possess scalability issues; the MSW scheme requires $O(\log n)$ work for tag identification but needs the greatest amount of tag storage and the largest number of cryptographic function computations in a tag; the HAC, HM, D, T1, T2, T3 and BMM schemes need only $O(1)$ work for tag identification and authentication, but all have critical security shortcomings.

These comparisons are intended to be helpful for implementers considering the use of an RFID protocol, and who need to compare the privacy, security and performance properties of a range of schemes. For example, if the application requires the number of complex function computations in a tag to be minimised, then properties C_2 and C_3 in Table 5.2 will be key to their choice; alternatively, if scalability is a critical requirement, then low complexity in column C_5 is the key criterion; finally, if minimising tag memory is particularly important, then protocols which have a small entry in column C_1 are advantageous.

5.18 Summary

We have reviewed a number of recently proposed RFID identification and authentication protocols. We have also assessed their privacy, security and performance properties against the requirements identified in chapter 4.

In the following chapters we describe a series of novel schemes which have privacy, security and performance properties superior to those of the prior art summarised here.

Table 5.1: Privacy and security properties

	$P1$	$P2$	$W1$	$W2$	$W3$	$W4$	$S1$	$S2$	$S3$	$A1$	$A2$
HAC	√	·	·	·	·	√	·	·	·	×	×
RAC	√	√	·	·	·	√	·	·	·	×	×
OSK	√	√	·	·	·	√	√	·	·	×	×
HM	√	·	·	·	·	√	·	·	·	×	○
MW	√	√	√	√	√	√	·	·	·	○	○
MSW	√	√	·	·	·	√	·	·	·	×	×
D	√	·	√	√	√	·	√	·	·	○	○
DPLK	√	√	·	·	·	·	·	·	·	×	○
LK	√	·	√	√	√	√	√	∗	∗	○	○
OTYT	√	·	√	·	·	·	·	·	·	×	×
CC	√	·	·	√	√	·	·	·	·	×	○
PHER	√	√	√	√	√	√	·	·	·	○	○
FA1	√	√	·	·	√	√	·	·	·	×	○
FA2	√	√	·	·	√	√	·	·	·	×	○
T1	√	√	√	√	√	·	·	·	·	×	×
T2	√	√	√	√	√	·	·	·	·	○	×
T3	√	√	√	√	√	·	·	·	·	○	×
BMM	√	·	√	√	√	√	·	·	·	○	○

P1 : tag information leakage
P2 : tag location tracking
W1 : tag impersonation
W2 : replay attack
W3 : man-in-the-middle attack
W4 : denial-of-service attack
S1 : backward traceability
S2 : forward traceability
S3 : server impersonation
A1 : tag authentication
A2 : server authentication
√ : resists such an attack
∗ : partially resists such an attack, under certain assumptions
· : does not protect against such an attack
○ : provides the property
× : does not provide the property

Table 5.2: Performance properties

	C1	C2	C3	C4	C5
HAC	ID, HID	1HF	0	2/4	$O(1)$
RAC	ID	1HF	1	2/3	$O(n)$
OSK	s	2HF	0	2	$O(n)$
HM	ID, TID, LST	3HF	0	3	$O(1)$
MW	ID, k	2PRF	1	3	$O(\log n)$
MSW	$(k_0, k_1, \cdots, k_{d_1}), c$	dPRF	$d_2 + 1$	2	$O(\log n)$
D	ID	4HF	1	3	$O(1)$
DPLK	EPC, k, p	3CRC	2	3	$O(n)$
LK	s, w_T, c, m	4PRF	1	3	$O(m)$
OTYT	e, k	1HF	0	2/3	$O(n)$
CC	EPC, k, p	2CRC	3	3	$O(n)$
PHER	t, a, p	6HF	1	5	$O(n)$
FA1	k, k', c, c_{max}	1/5HF	1	3	$O(n)$
FA2	k, k', c, c_{max}	1/2SE	1	3	$O(n)$
T1	k, t, t_m	1HF	0/1	2	$O(1)/O(n)$
T2	k, t, t_m	2HF	1/2	2	$O(1)/O(n)$
T3	k, t, t_m, z	$(\gamma + 2)$HF	1/3	2	$O(1)/O(n)$
BMM	k, τ, q, b, c	1/2PRF	0	3	$O(1)$

C1	: the type of secrets stored in a tag
C2	: the type and number of cryptographic function computations required in a tag
C3	: the number of pseudo-random numbers generated in a tag
C4	: the number of exchanged messages
C5	: server complexity to identify/authenticate a tag
HF	: a hash function computation
PRF	: a pseudo-random function computation
SE	: a symmetric encryption operation
/	: or

Chapter 6

A Novel Security Requirement

Contents

In this chapter we introduce server impersonation attacks, a practical security threat to RFID security protocols that has not previously been described. We first give a general description of attack models and present server impersonation attacks on synchronisation-based RFID protocols. Section 6.2 describes how a server impersonation attack can bring about desynchronisation in a number of existing RFID schemes. In section 6.3 we then propose possible countermeasures to server impersonation attacks; finally, section 6.4 contains a summary of the results.

6.1 A Novel Security Threat

A variety of security and privacy threats to RFID authentication protocols have been widely studied, including eavesdropping, replay attacks, DoS attacks, and tag tracking, as described in chapter 4. In this section we introduce another practical threat, namely server impersonation attacks.

In the prior art, the main purpose of authentication of the server to the tag is to combat server impersonation. However, in this chapter the term server impersonation attack is used in a very specific sense, namely to refer to desynchronisation attacks involving impersonation of a server to a tag that are made possible by compromise of a tag's internal state.

6.1.1 RFID Protocols

Many protocols have been proposed for use in RFID systems (see, for example, chapter 5). We focus here on RFID authentication protocols requiring synchronisation between a tag and a server, which operate under the following assumptions as well as the assumptions identified in section 3.3.1.

- An RFID protocol consists of three flows; typically, the first flow is a query from a server to a tag, the second is the reply from the tag to the server for tag authentication, and the third is the response from the server to the tag for server authentication.
- A server and a tag share secrets used for mutual authentication. They update the shared secrets synchronously whenever they perform a successful authentication session; a server updates its stored tag secrets after receiving the second flow and having authenticated the tag, and the tag updates its stored secrets after receiving the third flow and having authenticated the server.

6.1.2 Attack Models

Security threats to RFID protocols can be classified into weak and strong attacks, as described in chapter 4. Weak attacks just rely on observing and manipulating communications between a server and tags. Strong attacks become possible for an attacker which has compromised a target tag. Backward traceability and forward traceability are both examples of strong attacks. Forward traceability is related to tag ownership transfer [41]. This is because, if an RFID scheme does not provide forward untraceability and the ownership of a tag is transferred, then the previous owners might be able to read communications between the new owner and the tag [41].

In addition to these traceability threats, an SA that has compromised a tag could impersonate a valid server using knowledge of the tag's internal state. Such an adversary might be able to ask the tag to update its internal state, with the effect that the tag can no longer communicate successfully with the real server. Such a *server impersonation attack* is a significant issue for secure tag ownership transfer, as well as in relation to backward and forward traceability. For example, suppose that an RFID protocol does not resist forward traceability and server impersonation. Suppose further that, using this protocol, a previous tag owner has passed ownership of a tag to a new owner, but knows the tag secrets at the time of transfer. The previous owner might be able to use this knowledge to impersonate the new owner's server to the tag, and change the tag secrets after ownership transfer. As a result, the new owner might no longer be able to read the tag successfully, and only the previous owner would be able to identify the tag. This attack does not appear to have been discussed previously, despite its potential importance. We discuss such server impersonation attacks in greater detail below.

6.1.3 Server Impersonation Attacks

Server impersonation means that an adversary is able to impersonate a valid server to a tag. One reason that this is a genuine threat is because desynchronisation can occur if a tag updates its stored data when the server does not. More specifically, in protocols satisfying the assumptions given in section 6.1.1, an attacker that has read a tag's stored secrets could impersonate an authorised server to the tag. If the attacker executes an authentication session with the tag, impersonating a valid server, then it could make the tag update its stored secrets, although the genuine server will not update its stored data. The tag and the real server would then be desynchronised, and incapable of successful communications.

Whether or not the compromise of tag stored secrets enables such a server impersonation attack is the main focus of the next section.

6.2 Server Impersonation Attacks on RFID Protocols

Our focus here is on desynchronisation attacks arising from server impersonation attacks, as discussed in section 6.1.3. We now review four recently proposed RFID schemes as typical examples of protocols fitting the model given in section 6.1.1; in each case we consider whether or not the compromise of a tag's secret data enables a server impersonation attack.

6.2.1 The Henrici-Müller Protocol

We first examine the HM protocol, described in section 5.4.

Server impersonation attack

It is straightforward for a strong attacker to carry out a server impersonation based desynchronisation attack on the HM protocol. If an attacker knows the secrets stored in a tag, i.e. ID_i, TID_i and LST_i, then the attacker can impersonate the server and complete a successful authentication session with the tag. The attacker can send a query to the target tag, receive M_1, M_2 and δ from the tag in reply, and then send a valid response, r and M_3, to the tag. The tag will then verify M_3, and update its stored value of ID_i to $ID_i \circ r$, but the genuine server will not update its value of ID_i. As a result, the server and the tag will become desynchronised. Figure 6.1 summarises this server impersonation attack.

6.2.2 The Dimitriou Protocol

We next examine the D scheme, described in section 5.8.

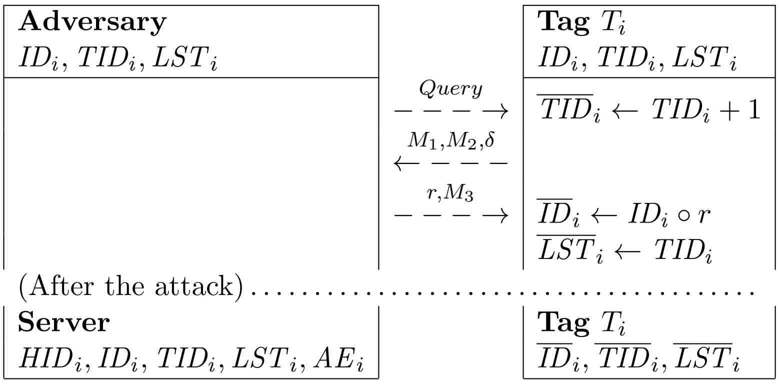

Adversary
ID_i, TID_i, LST_i
Tag T_i
ID_i, TID_i, LST_i
$\xrightarrow{Query}$
$\overline{TID}_i \leftarrow TID_i + 1$
$\xleftarrow{M_1, M_2, \delta}$
$\xrightarrow{r, M_3}$
$\overline{ID}_i \leftarrow ID_i \circ r$
$\overline{LST}_i \leftarrow TID_i$
(After the attack)
Server
$HID_i, ID_i, TID_i, LST_i, AE_i$
Tag T_i
$\overline{ID}_i, \overline{TID}_i, \overline{LST}_i$

Figure 6.1: Server impersonation attack on the HM protocol

Server impersonation attack

The D scheme is subject to a server impersonation attack analogous to that described on the HM protocol. If an adversary knows ID_i, then it can impersonate the server to conduct an authentication session with T_i. The adversary will receive r_2, M_1 and M_2 as a response from T_i, when it sends a query r_1 to the tag. Using the compromised tag information, the adversary can then respond with a valid M_3. As a result of receiving M_3, the tag will update its copy of the identifer ID_i to $g(ID_i)$; however, the genuine server will not update its stored data. The server and the tag will then become desynchronised. Figure 6.2 summarises this server impersonation attack.

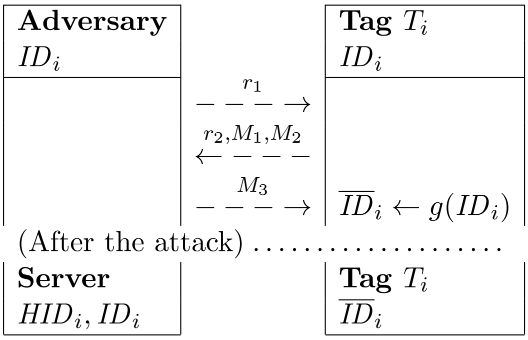


Figure 6.2: Server impersonation attack on the D protocol

6.2.3 The Chien-Chen Protocol

In this section we consider the CC scheme, described in section 5.12.

Server impersonation attack

In the CC scheme, server impersonation attacks remain a practical threat to synchronisation between the server and the tag.

An adversary that has read EPC_i, k_i and p_i from a tag T_i can commence a session with the tag by sending it a random number r_1. When the tag responds with r_2 and M_1, the attacker is able to send the expected value of M_2 back to the tag. As a result, the tag will update both its session key k_i and its access key p_i. The tag will then have stored keys that are different to those held by the server, as shown in Figure 6.3.

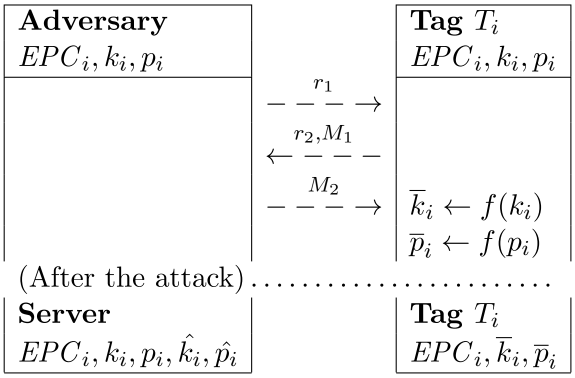

Adversary
EPC_i, k_i, p_i
Tag T_i
EPC_i, k_i, p_i
r_1
r_2, M_1
M_2
$\overline{k}_i \leftarrow f(k_i)$
$\overline{p}_i \leftarrow f(p_i)$
(After the attack)
Server
$EPC_i, k_i, p_i, \hat{k}_i, \hat{p}_i$
Tag T_i
$EPC_i, \overline{k}_i, \overline{p}_i$

Figure 6.3: Server impersonation attack on the CC protocol

6.2.4 The Lim-Kwon Protocol

In this section we consider the LK scheme, described in section 5.10.

Server impersonation attack

In the LK scheme, the server possesses a value $w_{i,S}$ that is used to authenticate the server to the tag, which the tag does not possess. This means that an adversary that has compromised a tag is not able to impersonate a server to the tag using an approach analogous to those described above for the other three protocols. That is, an adversary cannot masquerade as a valid server to a tag T_i, even if it knows both the tag secret s_i and $w_{i,T}$, because it does not know $w_{i,S}$ and hence is unable to compute M_3.

However, a server impersonation attack could still be feasible if an adversary performs a more elaborate attack; if an adversary first performs a DoS attack or a tag impersonation attack in a previous valid authentication session, and obtains $w_{i,S}$ from the messages sent in this session (using the compromised tag secret s_i), it could subsequently carry out a server impersonation attack in a session with the tag, as long as no valid session has been performed since the tag compromise. Such an attack will cause desynchronisation between the server and tag.

We describe two possible server impersonation based desynchronisation attacks on the LK protocol of this general type. In both cases we assume that the adversary has compromised the tag secrets s_i and $w_{i,T}$ before launching the attack.

- **Server impersonation attack after a DoS attack**

  Suppose that an adversary is able to eavesdrop on communications in a normal authentication session between the tag and a server, and also prevent the third flow from reaching the tag. As a result, the server will refresh the tag secret s_i to $\breve{s}_i = g(s_i \oplus (w_{i,S} \parallel r_1 \parallel r_2))$, but the tag will update its stored tag secret s_i to $\dot{s}_i = g(s_i)$. The adversary can now compute the server authentication secret $w_{i,S}$ from M_3 using the compromised tag secret s_i, as shown in Figure 6.4(a), and then calculate the session key $\dot{s}_i$ that the tag currently stores. The adversary can now start a new session with the tag, impersonating the server to the tag. If this authentication session is successful, then the tag will refresh its session key to $\tilde{s}_i = g(\dot{s}_i \oplus (w_{i,S} \parallel r_1' \parallel r_2'))$, but the server will not change its stored values for the tag. As a result of combining a DoS attack and a server impersonation attack, the server and the tag will become desynchronised.

- **Server impersonation attack after a tag impersonation attack**

  An attacker that knows the secret values for a tag T_i, i.e. s_i and $w_{i,T}$, can easily impersonate the tag to a server as follows. When a server sends a query r_1 to the tag, the attacker impersonates the tag to reply with r_2, M_1 and M_2.

> The server will then send M_3 to the attacker, and will update its copy of the tag secret from s_i to $\breve{s}_i = g(s_i \oplus (w_{i,S} \parallel r_1 \parallel r_2))$. The attacker is now able to compute the server secret $w_{i,S}$ from M_3. As a result, the attacker is capable of impersonating the server to commence a separate session with the tag. If the session between the attacker and the tag completes successfully, the tag will refresh its session key to $\tilde{s}_i = g(s_i \oplus (w_{i,S} \parallel r'_1 \parallel r'_2))$. However, the real server will keep the value $\breve{s}_i$ as the session key for the tag. Authentication between the server and the tag will now be impossible. This attack is summarised in Figure 6.4(b).

However, both the attacks we have just described are rather complex, and may be difficult to conduct in practice.

6.3 Countermeasures and Future Work

Once an RFID tag's stored secrets have been compromised, it is difficult to prevent server impersonation based desynchronisation attacks, as we have shown in our analysis of four existing RFID protocols in section 6.2.

We could attempt to design more robust RFID protocols that make such server impersonation attacks more difficult to perform. If an RFID protocol uses a digital signature scheme for authentication of a server to a tag, then a SA is unable to impersonate the server to a tag just by compromising the tag. However, the use of public key cryptography may be beyond the capabilities of many tags.

Another possible countermeasure involves a tag storing the most recent old secrets as well as its current secrets, as proposed for the server DB. As a result, the protocol would be more resistant to a desynchronisation threat caused by a server impersonation attack; an adversary would have to carry out at least two sessions to succeed in such a desynchronisation attack. However, such an approach might increase tag cost.

A further possible alternative approach would be to require the server to possess a secret that is used for server authentication to a tag and that is not known to the tag, like the value $w_{i,S}$ in the LK protocol. In this latter protocol, the server possesses tag-specific secrets $w_{i,S}$, which the tag does not have. The tag can authenticate the

Server	**Adversary**	**Compromised Tag:** T_i
Session (1)		
$[\ s_i, \{t_i^k\}_{k=0}^{m-1}, u_i, v_i, w_{i,S}, w_{i,T}\ ]$, $[\ \hat{s}_i, \{\hat{t}_i^k\}_{k=0}^{m-1}, \hat{v}_i, \hat{w}_{i,S}, \hat{w}_{i,T}\ ]$	$[\ s_i, w_{i,T}\ ]$	$[\ s_i, w_{i,T}, c_i, m\ ]$
	$\xrightarrow{r_1}$	
	$\xleftarrow{r_2, M_1, M_2}$	
	$\xrightarrow{M_3}$ (Block)	
Refresh $\breve{s}_i \leftarrow g(s_i \oplus (w_{i,S} \parallel r_1 \parallel r_2))$ $\breve{v}_i \leftarrow v_i - 1$ $\breve{w}_{i,T} \leftarrow w_{i,S}$ $\breve{w}_{i,S} \leftarrow h^{v_i}(u_i)$	$w_{i,S} \leftarrow M_3 \oplus f(s_i, r_2 \parallel r_1)$ $\dot{s}_i \leftarrow g(s_i)$	Update $\dot{s}_i \leftarrow g(s_i)$
Session (2)		
$[\ \breve{s}_i, \{\breve{t}_i^k\}_{k=0}^{m-1}, u_i, \breve{v}_i, \breve{w}_{i,S}, \breve{w}_{i,T}\ ]$, $[\ s_i, \{t_i^k\}_{k=0}^{m-1}, v_i, w_{i,S}, w_{i,T}\ ]$	$[\ \dot{s}_i, w_{i,S}, w_{i,T}\ ]$	$[\ \dot{s}_i, w_{i,T}, c_i, m\ ]$
	$\xrightarrow{r'_1}$	
	$\xleftarrow{r'_2, M'_1, M'_2}$	
	$\xrightarrow{M'_3}$	Refresh $\tilde{s}_i \leftarrow g(\dot{s}_i \oplus (w_{i,S} \parallel r'_1 \parallel r'_2))$ $\tilde{w}_{i,T} \leftarrow w_{i,S}$
$[\ \breve{s}_i, \{\breve{t}_i^k\}_{k=0}^{m-1}, u_i, \breve{v}_i, \breve{w}_{i,S}, \breve{w}_{i,T}\ ]$, $[\ s_i, \{t_i^k\}_{k=0}^{m-1}, v_i, w_{i,S}, w_{i,T}\ ]$		$[\ \tilde{s}_i, \tilde{w}_{i,T}, c_i, m\ ]$

(a) Server impersonation attack after a DoS attack

Server	**Adversary**	**Compromised Tag:** T_i
Session (1)		
$[\ s_i, \{t_i^k\}_{k=0}^{m-1}, u_i, v_i, w_{i,S}, w_{i,T}\ ]$, $[\ \hat{s}_i, \{\hat{t}_i^k\}_{k=0}^{m-1}, \hat{v}_i, \hat{w}_{i,S}, \hat{w}_{i,T}\ ]$	$[\ s_i, w_{i,T}\ ]$	$[\ s_i, w_{i,T}, c_i, m\ ]$
$\xrightarrow{r_1}$		
$\xleftarrow{r_2, M_1, M_2}$		
$\xrightarrow{M_3}$		
Refresh $\breve{s}_i \leftarrow g(s_i \oplus (w_{i,S} \parallel r_1 \parallel r_2))$ $\breve{v}_i \leftarrow v_i - 1$ $\breve{w}_{i,T} \leftarrow w_{i,S}$ $\breve{w}_{i,S} \leftarrow h^{v_i}(u_i)$	$w_{i,S} \leftarrow M_3 \oplus f(s_i, r_2 \parallel r_1)$	
Session (2)		
$[\ \breve{s}_i, \{\breve{t}_i^k\}_{k=0}^{m-1}, u_i, \breve{v}_i, \breve{w}_{i,S}, \breve{w}_{i,T}\ ]$, $[\ s_i, \{t_i^k\}_{k=0}^{m-1}, v_i, w_{i,S}, w_{i,T}\ ]$	$[\ s_i, w_{i,S}, w_{i,T}\ ]$	$[\ s_i, w_{i,T}, c_i, m\ ]$
	$\xrightarrow{r'_1}$	
	$\xleftarrow{r'_2, M'_1, M'_2}$	
	$\xrightarrow{M'_3}$	Refresh $\tilde{s}_i \leftarrow g(s_i \oplus (w_{i,S} \parallel r'_1 \parallel r'_2))$ $\tilde{w}_{i,T} \leftarrow w_{i,S}$
$[\ \breve{s}_i, \{\breve{t}_i^k\}_{k=0}^{m-1}, u_i, \breve{v}_i, \breve{w}_{i,S}, \breve{w}_{i,T}\ ]$, $[\ s_i, \{t_i^k\}_{k=0}^{m-1}, v_i, w_{i,S}, w_{i,T}\ ]$		$[\ \tilde{s}_i, \tilde{w}_{i,T}, c_i, m\ ]$

(b) Server impersonation attack after a tag impersonation attack

Figure 6.4: Server impersonation attacks on the LK protocol

server by checking that the value of $w_{i,S}$ sent by the server is valid, because the value $w_{i,T}$ held by the tag is a hash of $w_{i,S}$. Thus, such a scheme can resist server impersonation attacks resulting from tag compromise. As described in section 6.2.4, a server impersonation based desynchronisation attack on the LK scheme remains possible, but the method is rather complex. This may be sufficient to make such attacks impractical.

An important challenge for future work is to design a robust and practical RFID protocol that is able to resist strong attacks such as server impersonation, whilst minimising cost and maximising performance.

6.4 Summary

A server impersonation based desynchronisation attack is a feasible security threat because RFID tag memory is typically not tamper-resistant. In this chapter we have shown how, in cases where tag memory has been compromised, certain previously proposed RFID protocols can be desynchronised by a server impersonation attack; such an attack is relatively straightforward to perform on the MH, D and CC schemes, but more difficult for the LK scheme because of its use of an authentication key known to the server but not the tag.

We have also proposed possible countermeasures designed to make an RFID protocol more resistant to such server impersonation attacks — one problem is that implementing these measures might increase tag cost. That is, we have a trade-off between security and cost. We conclude that server impersonation attacks should be considered in any future security assessment of an RFID protocol.

Chapter 7

A Novel RFID Authentication Protocol

Contents

In this chapter we propose an RFID authentication protocol designed to address the identified privacy and security requirements, whilst also preserving desirable performance characteristics. The principles on which the novel protocol are based are described in section 7.1, and the protocol itself is presented in section 7.2. Section 7.3 analyses the protocol and compares it to the prior art. The final section summarises the results.

7.1 Design Principles

The goals of the new protocol are to meet the privacy and security requirements given in chapter 4, and also to maximise performance efficiency.

The protocol involves three messages, and uses a challenge-response approach. It uses random numbers to give anonymity for each tag response, as in the CC and LK protocols (see chapter 5). The server database stores both the most recent and the current data for each tag to protect against desynchronisation between the server and tags, like the HM, CC and LK protocols. If the authentication procedure is successful, then the server and tag refresh their shared secrets probabilistically using exchanged random numbers, thereby providing untraceability, as in the LK protocol.

One of the main features of the protocol is that a random number generated by a tag serves as a temporary secret for the tag. Another feature is that a tag only needs to store an identifier, where this identifier is the cryptographic hash of a bit-string assigned to the tag. The server database stores tag bit-strings as well as tag identifiers. The bit-string is used for server validation.

We use the following notation in the protocol description.

h	A hash function, $h : \{0,1\}^* \to \{0,1\}^l$
f	A keyed hash function, $f : \{0,1\}^* \times \{0,1\}^l \to \{0,1\}^l$ (a MAC algorithm)
l	The bit-length of a tag identifier
s_i	A string of l bits assigned to the ith tag T_i
t_i	Tag T_i's identifier of l bits, which equals $h(s_i)$
r	A random string of l bits

7.2 Protocol Description

We now introduce the novel authentication protocol, which is summarised in Figure 7.1.

7.2.1 Initialisation

The following steps must be performed prior to using the protocol.

- A server assigns a string s_i of l bits to each tag T_i, computes $t_i = h(s_i)$, and stores t_i in the tag. The string length l should be large enough so that an exhaustive search to find the l-bit values t_i and s_i is computationally infeasible (e.g. $l = 64$, 96 or 128).

- The server stores the entries $(s_i, t_i, \hat{s}_i, \hat{t}_i)$ for every tag that it manages. $\hat{s}_i$ and $\hat{t}_i$ are the previously assigned values for s_i and t_i, respectively. Initially $(\hat{s}_i, \hat{t}_i)$ is set to null.

7.2.2 Authentication Process

The protocol involves the following sequence of steps.

1. A server generates a random l-bit string r_1 and sends it to T_i.
2. The tag T_i generates a random l-bit string r_2 as a temporary secret for the session, and computes $M_1 = t_i \oplus r_2$ and $M_2 = f_{t_i}(r_1 \| r_2)$. T_i then sends r_1, M_1 and M_2 to the server.
3. When the server receives the values M_1 and M_2 from the tag, it performs the following steps.
   (a) The server searches its database as follows.
      i. It chooses the next value t_i from amongst the values t_i or $\hat{t}_i$ stored in the database.
      ii. It computes $M_2' = f_{t_i}(r_1 \| (M_1 \oplus t_i))$.
      iii. If $M_2' = M_2$, then it has identified and authenticated T_i. It then goes to step (c). Otherwise, it returns to step i.
   (b) If no match is found, the server stops the session.
   (c) The server computes $r_2 = M_1 \oplus t_i$ and $M_3 = s \oplus f_{t_i}(r_2 \| r_1)$ and sends r_1 and M_3 to the tag.
   (d) The server updates $\hat{s}_i$ and $\hat{t}_i$ for the tag T_i to the current value of s_i and t_i respectively, and then updates s_i and t_i so that $s_i \leftarrow (s_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2$ and $t_i \leftarrow h(s_i)$.
4. When T_i receives the value M_3, it computes $s_i = M_3 \oplus f_{t_i}(r_2 \| r_1)$ and checks that $h(s_i) = t_i$. If the check succeeds, the tag has authenticated the server, and sets $t_i \leftarrow h((s_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2)$. If the check fails, the tag keeps the current value of t_i unchanged.

S $[T_i : s_i, t_i, \hat{s_i}, \hat{t_i}]$		T_i $[t_i]$
Generate r_1		
	$\xrightarrow{r_1}$	
		Generate r_2 $M_1 = t_i \oplus r_2$ $M_2 = f_{t_i}(r_1 \| r_2)$
	$\xleftarrow{r_1, M_1, M_2}$	
Find t_i in the DB s.t. $M_2 = f_{t_i}(r_1 \| (M_1 \oplus t_i))$ $r_2 = M_1 \oplus t_i$ $M_3 = s \oplus f_{t_i}(r_2 \| r_1)$		
	$\xrightarrow{r_1, M_3}$	
$\hat{s_i} \leftarrow s$ $\hat{t_i} \leftarrow t_i$ $s_i \leftarrow (s_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2$ $t_i \leftarrow h(s_i)$		$s_i = M_3 \oplus f_{t_i}(r_2 \| r_1)$ If $h(s_i) = t_i$, $t_i \leftarrow h((s_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2)$

Figure 7.1: The authentication protocol

7.2.3 Evolution of the Protocol

The protocol proposed above is a revised version of a protocol (referred to as the SM protocol) we originally proposed in [65]. In [9, 75], attacks are described on the SM protocol; tag impersonation, server impersonation and desynchronisation by DoS attacks are all possible, which arise because the XOR operation is used to construct each of messages M_1, M_2 and M_3.

Cai et al. [9] presented a revised scheme designed to resist such attacks, in which the construction of M_2 uses concatenation instead of XOR, and M_3 uses a hash function $h(r_2)$ instead of $(r_2 \gg l/2)$.

The protocol presented in section 7.2 is a further revision of the original SM protocol. The scheme incorporates the following changes from the original version.

- The construction of M_2 has changed from $f_{t_i}(r_1 \oplus r_2)$ to $f_{t_i}(r_1 \| r_2)$. This is the same as in the updated scheme introduced by Cai et al. [9].
- The construction of M_3 has changed from $s_i \oplus (r_2 \gg l/2)$ to $s_i \oplus f_{t_i}(r_2 \| r_1)$.

7.3 Analysis of the Protocol

We analyse the protocol described in section 7.2 against the requirements identified in chapter 4.

7.3.1 Privacy and Security

The protocol has the following privacy and security properties.

- **Tag Information Privacy**: the detailed information of tag T_i is stored in the database of the server, which is assumed to be secure. Thus, only a legitimate server that has information related to the tag can extract a tag identifier t_i from the pair (M_1, M_2) sent by T_i, and can then access the information associated with the tag.
- **Tag Location Privacy**: the responses of the tag T_i are anonymous, since the tag only ever sends pairs (M_1, M_2), which cannot be linked to any particular tag. In other words, the eavesdropper can neither link tag responses to previous responses from the same tag, nor distinguish one tag's responses from another's. It is thus difficult to track the location of a tag.
- **Tag Impersonation attack**: for a WA to impersonate T_i, it must be able to compute a valid response (M_1, M_2) to a reader query. However, it is hard to compute such a valid pair without knowledge of t_i. Of course, an SA is able to clone a tag.
- **Replay attack**: the scheme is a challenge-response authentication protocol using random numbers to resist replay attacks. The messages M_1, M_2 and M_3 are functions of freshly generated nonces r_1 and r_2, and thus these messages cannot be reused in other sessions.
- **DoS attack**: if the message M_3 is blocked and does not reach the tag, the shared secrets of the server and T_i might become desynchronised, since the server will refresh s_i and t_i but the tag T_i will keep the current value of t_i. However, the server maintains both the old and new values of s_i and t_i for each tag T_i in its database, so that the server can resynchronise with the tag in such a situation.

- **Backward Traceability**: an SA cannot identify the past interactions of T_i, even if it knows T_i's present internal state. That is, an attacker is unable to discover the previous identifiers of T_i because they are the cryptographic hashes of values not available to the tag.
- **Forward Traceability**: the scheme provides forward untraceability if 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 s_i that are needed to refresh t_i, then it cannot compute the new identifier and track future transactions.
- **Server Impersonation attack**: a legitimate server responds with a message M_3 to a tag in order to enable the tag to authenticate the server. An SA cannot create a valid M_3 without knowing s_i, and thus is unable to impersonate a legitimate server to T_i just by compromising t_i. However, if an attacker has access to all the exchanged messages and knows the value of t_i used in a single authentication session, it can compute the refreshed s_i for the following session. Hence, our protocol only resists such an attack on the assumption that an adversary does not have access to at least one of the values r_1, M_1 and M_3 in an authentication session that is performed between an authorised server and a tag, for which the adversary knows the tag's secret t_i.

Table 7.1 compares the protocol with six of the schemes introduced in chapter 5 with respect to the privacy and security properties discussed in chapter 4. The six schemes have been selected for comparison since they and the protocol proposed here all involve three flows and use a challenge-response approach.

It is clear from Table 7.1 that the proposed protocol satisfies a greater number of the privacy and security properties than any of the previously proposed schemes.

7.3.2 Performance

The DPLK, CC and LK schemes and the novel protocol all update the secrets shared between a server and a tag in every successful session in the same order; moreover, in each of these schemes the server database stores both the most recent and the current

Table 7.1: Privacy and security properties

Property	HM	MW	D	DPLK	CC	LK	Sec. 7.2
Information leakage	√	√	√	√	√	√	√
Location tracking	·	√	·	√	·	·	√
Tag impersonation	·	√	√	·	·	√	√
Replay attack	·	√	√	·	√	√	√
DoS attack	√	√	·	·	·	√	√
Backward traceability	·	·	√	·	·	√	√
Forward traceability	·	·	·	·	·	∗	∗
Server impersonation	·	·	·	·	·	∗	∗

√ : resists such an attack

∗ : partially resist such an attack, under certain assumptions

· : does not protect against such an attack

data for each tag. These schemes also all use pseudo-random number generators and checksum algorithms to protect the integrity of messages.

We compare the storage and computation requirements of both a server and a tag for these four protocols. Table 7.2 gives the storage needed in the server database and the memory required in a tag for each of these protocols. In the case of the database, we restrict our attention to the storage required to support tag authentication. Table 7.2 indicates that the protocol given in section 7.2 requires the least tag-side memory of the four protocols assessed.

In order to compare the computational requirements of the four protocols, we take into account the checksum algorithms and secret updating functions that require the most computation in each protocol. The DPLK and CC protocols for the EPCglobal Class-1 Gen-2 RFID tag make use of a CRC as a checksum algorithm, which is efficient but less cryptographically strong, whereas the LK scheme uses pseudo-random functions (PRFs), and the protocol given in section 7.2 uses hash functions for checksum computation and secret updates, which provide enhanced security. Table 7.3 compares the number of times each such function is computed in these protocols, under the assumption that they utilise the same algorithm.

From Table 7.3 it follows that the protocol given in section 7.2 requires the same number of complex function invocations for the tag as the other three protocols.

To sum up, the novel protocol has tag storage advantages over the DPLK, CC and

LK schemes, whilst also providing the identified privacy and security properties.

Table 7.2: Storage requirements

Storage	DPLK	CC	LK	Sec. 7.2
Server	$(l + l_1 + l_2)n$	$(l + 2l_1 + 2l_2)n$	$2(l + 3l_3 + ml_4)n$	$4ln$
Tag	$l + l_1 + l_2$	$l + l_1 + l_2$	$l + l_3 + 2l_5$	l

n : The number of tags
m : The maximum number of authentication failures in the LK protocol (e.g., $m = 64$ [41])
l : The bit-length of a tag identifier (e.g., $l = 64, 96$ or 128 [10, 41])
l_1 : The bit-length of a PIN in the DPLK and CC protocols (e.g., $l_1 = 32$ [10])
l_2 : The bit-length of a session key in the DPLK and CC protocols (e.g., $l_2 = 16$ [10])
l_3 : The bit-length of a server validator in the LK protocol (e.g., $l_3 = 32$ [41])
l_4 : The bit-length of a tag secret transmitted in the LK protocol (e.g., $l_4 = 32$ [41])
l_5 : The bit-length of the number m and a tag counter in the LK protocol (e.g., $l_5 = 7$ [41])

Table 7.3: Computation requirements

Computation		DPLK	CC	LK	Sec. 7.2
Server	On receiving the 2nd flow	$(k + 1)F$	kF	$(k_1 + 1)F$	kF
	On sending the 3rd flow	$1F$	$1F$	$1F$	$1F$
	On updating or refreshing	$1F$	$2F$	$(k_1 + k_2 + m)F$	$1F$
	Total	$(k + 3)F$	$(k + 3)F$	$(2k_1 + k_2 + m + 2)F$	$(k + 2)F$
Tag	On sending the 2nd flow	$2F$	$1F$	$1F$	$1F$
	On receiving the 3rd flow	$1F$	$1F$	$2F$	$2F$
	On updating or refreshing	$1F$	$2F$	$1F$	$1F$
	Total	$4F$	$4F$	$4F$	$4F$

F : A computationally complex function (such as a CRC, PRF or hash function)
m : The maximum number of authentication failures in the LK protocol (e.g., $m = 64$ [41])
v : The length of the backward key chain in the LK protocol (e.g., $v = 2^{20}$ [41])
k : An integer satisfying $1 \leq k \leq 2n$
k_1 : An integer satisfying $0 \leq k_1 \leq m - 1$
k_2 : An integer satisfying $0 \leq k_2 \leq v - 2$

7.4 Summary

We have presented a new RFID authentication protocol designed to meet the requirements identified in chapter 4. This scheme is a revised version of a previously proposed protocol [65], which has been successfully attacked [9, 75].

The novel protocol has been compared with existing schemes with respect to both its privacy and security properties and its storage and computational requirements. The comparisons have shown that the protocol described here is both more secure than previously proposed schemes and has practical advantages over them, because it provides the greatest number of identified privacy and security features and requires the least tag-side storage.

Chapter 8

RFID Tag Ownership Transfer Protocols

Contents

In this chapter we examine the requirements for secure tag ownership transfer, and propose RFID authentication protocols satisfying these requirements. Section 8.1 describes the specific requirements for tag ownership transfer. Section 8.2 proposes new protocols for secure tag ownership transfer, designed to provide the identified properties. In the following section, the privacy and security properties of the protocols are analysed, together with their performance characteristics; the protocols for tag ownership transfer are also compared with the prior art. Section 8.4 concludes this chapter.

8.1 Tag Ownership Transfer

One functional requirement for RFID systems is secure tag ownership transfer, as described in chapter 4. In some applications, an RFID tag may change its owner a number of times during its lifetime. Ownership transfer means that the server of the new owner takes over tag authorisation, and so needs to be given the necessary private information to securely interact with and identify the tag. Thus all information associated with the tag will need to be passed from the old to the new owner.

However, at the moment of tag ownership transfer, both the old and new owners have the information necessary to authenticate a tag, and this fact may cause an infringement of tag owner privacy. More specifically, if the previous owner is malicious, it may still be able to read the tag using retained tag information after transfer, and/or trace the new owner's transactions with the tag. That is, the privacy of the new owner might be compromised by the previous owner. Conversely, if the new owner is malicious, then it might be able to trace the previous owner's past transactions with the tag. That is, the privacy of the previous owner might be compromised by the new owner.

The following requirements for secure tag ownership transfer have been identified [20, 41, 50]:

- **New Owner Privacy**: Once ownership of a tag has been transferred to a new owner, only the new owner should be able to identify and control the tag. The previous owner of the tag should no longer be able to identify or trace the tag.
- **Old Owner Privacy**: When ownership of a tag has been transferred to a new owner, the new owner of a tag should not be able to trace past interactions between the tag and its previous owner.
- **Authorisation Recovery**: In some special cases, the previous owner of a tag might need to temporarily recover the means to interact with it. For example, suppose that a customer has bought an RFID tagged item with guarantee provided by the retailer. If the customer finds a fault in the item, it will need to be returned to the retailer for repair or replacement. In such a case the retailer should be able to identify the item and check that it originally sold

> this item with an after-sales guarantee [20]. That is, the current owner of the tag should be able to transfer its authorisation rights over the tag to the previous owner.

The possible need for authorisation recovery in an RFID system was first raised in [20, 21]. However, it seems that no concrete protocol to address this possible requirement has been proposed. In the next section, we introduce RFID protocols for tag ownership transfer which support this requirement.

8.2 RFID Protocols for Tag Ownership Transfer

We propose RFID authentication protocols fulfilling the requirements described in section 8.1. We first introduce a secret update protocol that operates in conjunction with the protocol introduced in chapter 7 (referred to here as $\mathbf{P}_1$). We next present a novel authentication protocol designed to provide authorisation recovery.

The protocols are revised versions of protocols we previously proposed in [63]. The revisions have been made because the original protocols are based on the SM protocol, which has been shown to possess vulnerabilities (see also chapter 7).

The protocols introduced in this chapter work under the assumptions identified in sections 3.3.1 and 3.3.2. The protocols use one hash function $h : \{0,1\}^* \to \{0,1\}^l$ and two keyed hash functions $f : \{0,1\}^* \times \{0,1\}^l \to \{0,1\}^l$ and $g : \{0,1\}^* \times \{0,1\}^l \to \{0,1\}^{2l}$.

8.2.1 Tag Ownership Transfer and Secret Update

The owner of a tag can identify and authenticate it, and update its secrets, using $\mathbf{P}_1$. When tag ownership is to be transferred, the following process is performed. The current owner transfers all the information related to the tag, except for the most recent secrets $(\hat{s}, \hat{t})$, to the new owner via a secure channel. This information transfer should take place after the current owner has updated the secrets shared between its server and the tag, in order to protect its privacy against possible tracking by the new owner.

Once an entity has received all the tag information, it has ownership of the tag. However, in order to provide new owner privacy, as discussed in section 8.1, the server of the new owner needs to execute another protocol to establish new secrets with the tag.

A new owner can perform $\mathbf{P}_1$ to update the tag secrets. However, we propose using a novel secret update protocol (referred to as $\mathbf{P}_2$) to improve performance. In this protocol the cryptographic function computations performed by a tag, and the messages exchanged between a server and a tag, are less than in $\mathbf{P}_1$.

Prior to running $\mathbf{P}_2$, we assume that the server S of the new owner of tag T has the pair of current secrets (s, t) for T, obtained as a result of the tag information transfer; we also suppose that T has identifier t. The secret update protocol $\mathbf{P}_2$ involves the following steps.

1. S generates a new secret s' of l bits for T, and computes $t' = h(s')$. S generates a random string r_S, computes $M_S = g_t(r_S) \oplus (s \| t')$, and then sends r_S and M_S to T.
2. When T receives r_S and M_S from S, it performs the following steps:
   (a) T computes $(s \| t') = M_S \oplus g_t(r_S)$.
   (b) If $h(s) = t$, T has authenticated S as an authorised server. Otherwise, the session terminates.
   (c) T updates its secret t to t', and generates a random string r_T of l bits.
   (d) T computes $M_T = f_t(r_S \| r_T)$ using the new secret t, and sends r_T and M_T to S.
3. S checks that M_T is equal to $f_{t'}(r_S \| r_T)$. If the validation succeeds, S now knows that T has received the new secret t', and updates secrets s and t for T to s' and t', respectively. Otherwise, S goes to step 1, and starts a new session.

If $\mathbf{P}_2$ completes successfully (and the old owner does not eavesdrop on the messages), then S and T will share new secrets known only to them, and the old owner is no longer able to identify or trace T. Both the old and new owners can also keep the pair of the tag secrets provided by the old owner, denoted by $(\dot{s}, \dot{t})$, for use in the event that the old owner needs to identify T again. $\mathbf{P}_2$ is summarised in Figure 8.1.

S $[T : s, t]$		T $[t]$
Generate s' $t' = h(s')$ Generate r_S $M_S = g_t(r_S) \oplus (s \Vert t')$		
	$\xrightarrow{r_S, M_S}$	$(s \Vert t') = M_S \oplus g_t(r_S)$ If $h(s) = t$, $t \leftarrow t'$ Generate r_T $M_T = f_t(r_S \Vert r_T)$
If $M_T = f_{t'}(r_S \Vert r_T)$, $\dot{s} \leftarrow s,\ \dot{t} \leftarrow t$ $s \leftarrow s',\ t \leftarrow t'$	$\xleftarrow{r_T, M_T}$	

Figure 8.1: Secret update protocol

8.2.2 Authorisation Recovery

As discussed above, the previous owner of a tag may need to temporarily interact with the tag again. The following authentication protocol (referred to as $\mathbf{P}_3$) enables this to occur.

$\mathbf{P}_3$ enables server S to make T change its secret back to the value it had when S took ownership of T from the old owner. Prior to running $\mathbf{P}_3$, we assume that S stores the following information for tag T: the current secrets (s, t), the most recent secrets $(\hat{s}, \hat{t})$, and the old secrets also known to the old owner $(\dot{s}, \dot{t})$; we also suppose that tag T has identifier t.

$\mathbf{P}_3$ is the same as $\mathbf{P}_2$ except that t' is set equal to $\dot{t}$ without the step generating new secrets s' and t'. After successful execution of $\mathbf{P}_3$, T stores $\dot{t}$ as its identifier. As a result, the old owner can identify T again.

The current owner can recover authorisation rights over T from the old owner by executing $\mathbf{P}_2$.

8.3 Analysis of the Protocols

We evaluate the protocols proposed in section 8.2, namely $\mathbf{P}_2$ and $\mathbf{P}_3$, with respect to privacy, security, performance and tag ownership transfer, using the requirements presented in sections 4.1, 4.2 and 8.1.

8.3.1 Privacy and Security

$\mathbf{P}_2$ and $\mathbf{P}_3$ are mutual authentication protocols. In these schemes, when tag T receives r_S and M_S from server S, T authenticates S by obtaining s from the messages and checking that $h(s) = t$. This works because s is a secret specific to T known only by S. S authenticates T by checking that the received M_T is correct, since it is computed using a shared secret t which is known only to T and S. S can also confirm that T has the same new secret t' as S.

The schemes protect against tag information leakage because T's responses are a function of its secret t, and thus only the server that knows the secret is able to identify T and access the tag information. The schemes protect against tag location tracking because T's responses are anonymous, since they are a hash of random strings r_S and r_T, and are independent of one another.

The messages exchanged between the server and tag are computed using random strings r_S and r_T, secrets t and s, and keyed hash functions f and g. Thus, the protocols can resist replay attacks and man-in-the-middle attacks. To impersonate a tag, an attacker must be able to compute a valid response M_T. However, it is difficult to compute such a message without knowledge of the secret t, because an attacker cannot learn the updated key t' from the value M_S sent by S, and thus cannot compute M_T since it is a function of t'.

Denial-of-service attacks are not practical in $\mathbf{P}_2$ and $\mathbf{P}_3$. S knows the identity of a tag when it starts a session with the tag, and the purpose of these schemes is to update the tag secrets. Thus, if S does not receive a tag reply to its query, it can immediately detect the error and fix it. Suppose that message M_T does not reach S in $\mathbf{P}_2$ (or $\mathbf{P}_3$). Then T will update its identifier, but S will not. However, S knows the updated value of t, and can recover synchronisation with T using it.

Suppose that a tag T is compromised after $\mathbf{P}_2$ has been performed. The protocols do not enable backward traceability, because, in $\mathbf{P}_2$, T's new secret t' does not have any relationship to previous keys, and then, in $\mathbf{P}_1$, it is updated using a non-invertible hash function h. The schemes also resist forward traceability if an adversary does not obtain the values r_1, M_1 or M_3 exchanged between the server and tag in $\mathbf{P}_1$, because t is updated using r_1, r_2 and s after every successful session (see also chapter 7). Even if an adversary has compromised T, it cannot impersonate a legitimate server in $\mathbf{P}_2$ or $\mathbf{P}_3$ without additional information. This is because the server's message M_S is a function of the secret s known only to the server, and the adversary cannot obtain this value, even if it compromises T. To succeed in such an attack, the adversary must first perform other attacks to obtain s.

In Table 8.1, the privacy and security properties of $\mathbf{P}_2$ and $\mathbf{P}_3$ are compared to five schemes introduced in chapter 5. The five schemes have been selected for comparison since they and the protocol proposed here all involve tag authentication and secret update processes for tag ownership transfer. The table shows that the proposed schemes address the identified privacy and security threats.

Table 8.1: Privacy and security properties

	LK	OTYT	FA1	FA2	$\mathbf{P}_1$	$\mathbf{P}_2/\mathbf{P}_3$
Tag information leakage	√	√	√	√	√	√
Tag location tracking	·	·	√	√	√	√
Tag impersonation	√	√	·	·	√	√
Replay attack	√	·	·	·	√	√
Man-in-the-middle attack	√	·	√	√	√	√
Denial-of-service attack	√	·	√	√	√	√
Backward traceability	√	·	·	·	√	√
Forward traceability	∗	·	·	·	∗	∗
Server impersonation	∗	·	·	·	∗	∗

√ : resists such an attack
∗ : partially resists such an attack, under certain assumptions
· : does not protect against such an attack

8.3.2 Performance

$\mathbf{P}_2$ and $\mathbf{P}_3$ are efficient in terms of non-volatile memory and communication requirements, because a tag only needs l bits of non-volatile memory to store its secret t, and only two messages need to be exchanged to provide mutual authentication between the server and tag.

$\mathbf{P}_2$ and $\mathbf{P}_3$ have modest computational requirements. The only cryptographic computations (which are significantly more computationally complex computations than arithmetic and logical operations) required by $\mathbf{P}_2$ or $\mathbf{P}_3$ are at most three hash function computations in the tag and the server. In $\mathbf{P}_2$, both T and S need to compute each of h, f and g once. In $\mathbf{P}_3$, T needs to compute each of h, f and g once, and S needs to compute f and g once.

Therefore, in order to update a tag's secrets after transfer of ownership, performing $\mathbf{P}_2$ is more economical than performing $\mathbf{P}_1$ again.

Table 8.2 compares the performance characteristics of $\mathbf{P}_2$ and $\mathbf{P}_3$ with the secret update process for the four schemes analysed in Table 8.1. The table indicates the type and number of cryptographic function computations required of a tag, the type and number of entries stored in tag non-volatile memory, and the number of exchanged messages in each protocol.

Table 8.2: Performance of secret update protocol

	LK	OTYT	FA1	FA2	$\mathbf{P}_1$	$\mathbf{P}_2/\mathbf{P}_3$
Tag computations	4PRF	1HF	5HF	2SE	4HF	3HF
Tag storage	s, w_T, c, m	e, k	k, k', c, c_{max}	k, k', c, c_{max}	t	t
Message flows	3	3	3	3	3	2

8.3.3 Tag Ownership Transfer

The protocols proposed in section 8.2 meet the three requirements for tag ownership transfer identified in section 8.1.

First, $\mathbf{P}_2$ is designed to protect the privacy of the new owner from the old owner of a tag T. That is, future interactions between the new owner and T are secure against tracing by the old owner. When ownership of T is transferred to the new owner, the new owner and T establish new secrets using $\mathbf{P}_2$. As a result, the old owner is no longer able to read T.

Next, the protocols also protect the privacy of the old owner from the new owner of T. The old owner should update secrets s and t for T using $\mathbf{P}_2$, before transferring the updated secrets to the new owner. As a result, the new owner cannot trace previous transactions between the old owner and the tag since it only knows the updated secrets.

Finally, $\mathbf{P}_3$ provides authorisation recovery, the third requirement described in section 8.1. $\mathbf{P}_3$ causes T to change its secret t to $\dot{t}$, which the new owner received from the old owner when ownership of T was transferred. As a result, the old owner recovers authorisation rights for T, and thus can read T again and look up the tag information.

We compare the novel protocols to seven of the the previously proposed schemes introduced in chapter 5, in each of which tag ownership transfer is addressed. Table 8.3 provides a comparison of the novel protocols with these schemes with respect to the security of tag ownership transfer. The new protocols provide all the identified properties.

Table 8.3: Properties for tag ownership transfer

	MSW	SIS1	SIS2	LK	OTYT	FA1	FA2	Sec. 8.2
New owner privacy	×	○	○	○	○	×	○	○
Old owner privacy	×	×	×	×	○	×	○	○
Authorisation recovery	×	×	×	×	×	×	×	○

○ : provides such a property

× : does not provide such a property

8.4 Summary

In some RFID applications it is necessary to support transfer of tag ownership. We have identified three requirements for secure and privacy-preserving tag ownership transfer: new owner privacy, old owner privacy, and authorisation recovery. We have proposed novel RFID authentication protocols for tag ownership transfer that meet these requirements. The scheme consists of three protocols: an authentication protocol ($\mathbf{P}_1$), a secret update protocol ($\mathbf{P}_2$), and an authorisation recovery protocol ($\mathbf{P}_3$). We believe that $\mathbf{P}_3$ is the first proposed practical authentication scheme for authorisation recovery.

We have also analysed and compared $\mathbf{P}_2$ and $\mathbf{P}_3$ to the prior art. The schemes satisfy the identified privacy and security requirements. $\mathbf{P}_2$ and $\mathbf{P}_3$ have desirable performance characteristics; a tag needs less non-volatile memory than in previously proposed schemes, performs just three hash function computations, and the number of messages exchanged between the tag and the server is only two, with mutual authentication. $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ also provide all the identified requirements for tag ownership transfer.

Chapter 9

A Scalable RFID Pseudonym Protocol

Contents

This chapter considers the issue of scalability in anonymous RFID authentication protocols. We propose a novel RFID authentication protocol with desirable scalability properties, and that meets the identified desirable privacy, security and performance properties. We first, in section 9.1, introduce scalability issues in pseudonym protocols. In sections 9.2 and 9.3 we then present a novel scalable protocol, and in section 9.4 we analyse it against the identified requirements. Finally, in section 9.5, we summarise the contributions of this chapter.

9.1 Scalability Issues

One approach to protecting against privacy and security threats in RFID systems is to use a tag authentication scheme in which a tag is both identified and verified in a manner that does not reveal the tag identity to an eavesdropper. A large number of tag authentication protocols of this type have been proposed. Typically, pseudonyms are used to provide anonymity to tags; whenever a tag is queried, it responds with a different cryptographically derived pseudonym. In some of these pseudonym-based protocols, see for example the OSK, DPLK, OTYT, CC, PHER, FA1 and FA2 protocols, a server must perform a linear search of its database to identify a tag. That is, for each tag entry in the database in turn, it computes the pseudonym that would be produced by that tag (using stored secrets) and compares it with the received pseudonym. Such a linear search runs in $O(n)$ time, where n is the number of elements in the database (see section 4.2). Such a costly search function will potentially cause scalability issues as the tag population increases.

As discussed in section 4.2, scalability is a desirable property in almost any system, enabling it to handle growing amounts of work in a graceful manner [6]. A scalable RFID system should be able to handle large numbers of tags without undue strain, and a scalable RFID protocol should therefore avoid any requirement for work proportional to the number of tags.

Some RFID pseudonym schemes, see for example the HM, D, LK, T1, T2, T3, and BMM protocols, require only $O(1)$ work to identify a tag (see section 4.2). Most such schemes use look-up tables to match a value with a pseudonym received from a tag, thereby taking constant time without the need for a linear search. However, all previously proposed schemes of this type possess significant security, privacy or performance shortcomings, as discussed in chapter 5.

An alternative means of improving the scalability of an RFID system is delegation (see section 4.3). Tag delegation involves giving authorised entities the right to query and identify certain tags during a specified period. This clearly has the potential to reduce the server's workload. The MSW, LK, FA1 and FA2 schemes are examples of protocols supporting delegation.

In this chapter, we propose an RFID pseudonym protocol providing scalability that possesses two features, namely that a server takes only constant time to identify a tag, and that tag delegation is straightforward.

9.2 A Novel RFID Authentication Protocol

We introduce here a new RFID authentication protocol. This protocol provides scalability as well as satisfying the privacy, security and performance properties given in chapter 4. The RFID protocols considered here operate under the assumptions identified in sections 3.3.1 and 3.3.2.

9.2.1 Main Features

The protocol has the following main features:

- To improve scalability, the protocol makes use of a precomputed look-up table for tag authentication, as in the HM, D, LK, T1, T2, T3, and BMM schemes. As a result, the server takes $O(1)$ work to identify and authenticate a tag, without needing a linear search (see section 4.2).
- The look-up table contains a number of entries for each tag, one for each element of a tag-specific hash-chain. Elements from this hash-chain are used as tag identifiers (and as database keys to identify tags). A keyed hash function is used to generate each hash-chain, using a secret key shared by the tag and server. The hash-chain length, m, determines the number of tag identifiers that can be produced using any one key.
- The operation of the protocol, described in greater detail below, can be divided into three cases, as follows (see also Table 9.1):
    1. Case 1: for each of the first $m-1$ queries of a tag, the protocol process only involves tag authentication and requires just two messages. To authenticate a tag, the server searches a look-up table, taking constant time.
    2. Case 2: on the mth query of a tag, the protocol updates the secrets shared by the server and tag, as well as providing tag authentication.

This process requires an additional message. The server takes $O(1)$ work to authenticate a tag, as in case 1.

3. Case 3: if a tag is queried more than m times, which should not normally happen, then the protocol requires the server to perform a linear search with complexity $O(n)$.

Table 9.1: Operation of the protocol

Query number	$1, \cdots, (m-1)$	m	$(m+1), \cdots$
Operation	Case 1	Case 2	Case 3
State	Regular state		Irregular state

- In case 3 of the protocol, we apply the protocol defined in chapter 7 (referred to here as $\mathbf{P}_1$).
- In normal operation (cases 1 and 2), a tag does not need to generate pseudo-random numbers; however, in case 3, a pseudo-random number is needed to prevent tag tracking.
- For server authentication (in cases 2 and 3), for each tag the server holds a secret s that only it knows, as in the LK scheme and $\mathbf{P}_1$.

9.2.2 Initialisation

The server S chooses values for l and l_r and functions e, f, g and h as follows.

- l is the bit-length of a tag identifier and a shared key. It should be large enough to ensure that an l-bit key is a strong cryptographic key for the keyed hash functions, and in particular that an exhaustive search to find an l-bit tag identifier is computationally infeasible.
- l_r $(\leq l)$ is the bit-length of a random string. It should be large enough to ensure that an exhaustive search to find an l_r-bit value is computationally infeasible.
- $e : \{0,1\}^* \times \{0,1\}^l \rightarrow \{0,1\}^l$, $f : \{0,1\}^* \times \{0,1\}^l \rightarrow \{0,1\}^l$ and $g : \{0,1\}^* \times \{0,1\}^l \rightarrow \{0,1\}^{3l}$ are keyed hash functions.

- h is a hash function, $h : \{0,1\}^* \to \{0,1\}^l$.

The server S builds a look-up table which is used for tag identification. The table definition process involves the following steps for each tag T managed by S.

- S chooses a random l-bit string s, and computes the l-bit key $k = h(s)$, where s is used for server authentication and k is used as input to the keyed hash functions e, f and g.
- S chooses a random l-bit string x_0, and computes the hash-chain values $x_i = e_k(x_{i-1})$ for $1 \leq i \leq m$, where the values x_i are used as tag identifiers and m is the length of the hash-chain.
- S stores s, k and the identifiers $x_0, x_1, \cdots, x_m$ as the entries for T in its look-up table.

Each tag T stores k, x and x_m, where x is initially set to x_0 and functions as T's identifier.

9.2.3 Authentication and Secret Update

The novel protocol has three different stages in line with the cases described in section 9.2.1: tag authentication, secret update (I) and secret update (II). The stages are as follows (see also Figure 9.1).

Case 1: Tag Authentication
S generates a random l_r-bit string r, and sends r to T.

1. When T receives r, it compares its stored values of x and x_m. If $x \neq x_m$, then the following steps are performed.
   (a) T computes $M_T = f_k(r \| x)$ and updates its identifier x to $e_k(x)$. T sends r, x and M_T back to S. If the updated x is equal to x_m, T waits for a server response, keeping r and M_T in short term memory.
   (b) When S receives x and M_T, it performs the following steps.

i. S searches its look-up table for a value x_i equal to the received value of x. If such a value is found, S identifies T. Otherwise, the session terminates.

ii. S checks that $f_k(r \| x_{i-1})$ equals the received value of M_T, where k is the key belonging to the identified tag T. If this verification succeeds, then S authenticates T. Otherwise, the session terminates.

iii. If $x \neq x_m$, then the authentication session terminates successfully.

Case 2: Secret Update (I)

iv. If $x = x_m$, then S performs the following steps to update the secrets for T.

A. S chooses a random l-bit string s' and an integer m', and computes a key $k' = h(s')$ and a sequence of m' identifiers $x'_i = e_{k'}(x'_{i-1})$ for $1 \leq i \leq m'$, where x'_0 is set to x. (These values can be precomputed.)

B. S computes $M_S = g_k(r \| x \| M_T) \oplus (s \| k' \| x'_{m'})$, and sends r and M_S to T.

C. S updates the set of stored values for T from ($\hat{s}$, $\hat{k}$, s, k, x_0, $x_1, x_2, \cdots, x_m$) to (s, k, s', k', x, $x'_1, x'_2, \cdots, x'_{m'}$), where $\hat{s}$ and $\hat{k}$ are the most recent previous values of s and k, respectively.

(c) When T receives r and M_S, it computes $(s \| k' \| x'_{m'}) = M_S \oplus g_k(r \| x \| M_T)$. If $h(s)$ is equal to k, T authenticates S and updates k and x_m to k' and $x'_{m'}$, respectively. (The secret update session then terminates successfully.) Otherwise, the session terminates.

Case 3: Secret Update (II)

2. When T receives r, if T's stored values of x and x_m are equal, then the following steps are performed. (This irregular case arises if T did not update its shared secrets correctly in the previous session, that is, if the secret update (I) step fails.)

(a) T generates a random number r_T as a session secret, and computes $M_1 = f_k(r \| r_T)$ and $M_2 = r_T \oplus x$. T sends r, M_1 and M_2 back to S with a request for an update of the shared secrets. T waits for a server response, keeping r, r_T and M_1 in short term memory.

(b) When S receives M_1 and M_2, the following steps are performed.

i. S searches its look-up table for a value $x = x_m$ or $x = x_0$ for which $M_1 = f_k(r\|(M_2 \oplus x))$. If such a value is found, S authenticates T. Otherwise, the session terminates.

ii. If $x = x_m$, S performs the following steps. (This case arises when, although T sent $x = x_m$ to S in the previous session, S did not receive it correctly. Thus, neither S nor T have updated their shared secrets.)

A. S chooses a random l-bit string s' and an integer m', and computes a key $k' = h(s')$ and a sequence of m' identifiers $x'_i = e_{k'}(x'_{i-1})$ for $1 \leq i \leq m'$, where x'_0 is set to x. (These values can be precomputed.)

B. S computes $r_T = M_2 \oplus x$ and $M_S = g_k(r\|r_T\|M_1) \oplus (s\|k'\|x'_{m'})$, and sends r and M_S to T.

C. S updates the set of stored values for T from ($\hat{s}$, $\hat{k}$, s, k, x_0, $x_1, x_2, \cdots, x_m$) to (s, k, s', k', x, $x'_1, x'_2, \cdots, x'_{m'}$).

iii. If $x = x_0$, S computes $r_T = M_2 \oplus x$ and $M_S = g_{\hat{k}}(r\|r_T\|M_1) \oplus (\hat{s}\|k\|x_m)$ and sends r and M_S to T. (This case arises if M_S did not reach T correctly in the previous session, and thus T did not update its secrets, although S did. That is, this step resynchronises S and T.)

(c) When T receives r and M_S, it computes $(s\|k'\|x'_{m'}) = M_S \oplus g_k(r\|r_T\|M_1)$. If $h(s)$ is equal to k, T authenticates S and updates k and x_m to k' and $x'_{m'}$, respectively. (The secret update session then terminates successfully.) Otherwise, the session terminates.

9.3 Tag Delegation and Ownership Transfer

Tag delegation enables a server to delegate the right to identify and authenticate a tag to a specified entity for a limited time period [41, 45]. Such a procedure could be used to reduce the computational load on a server.

In the protocol described in section 9.2, tag delegation is straightforward. When S wants to delegate T to an entity, it transfers the secret k and the identifiers

S $[T : \hat{s}, \hat{k}, s, k, (x_0, \cdots, x_i, \cdots, x_m)]$		T $[k, x, x_m]$
Generate r		
	$\xrightarrow{r}$	
		If $x \neq x_m$, $M_T = f_k(r \Vert x)$ $x \leftarrow e_k(x)$
	$\xleftarrow{r,x,M_T}$	
Case 1: Find $x_i = x$ in the DB Check $M_T = f_k(r \Vert x_{i-1})$		
Case 2: If $x = x_m$, $M_S = g_k(r \Vert x \Vert M_T) \oplus (s \Vert k' \Vert x'_{m'})$		
	$\xrightarrow{r,M_S}$	
Update secrets for T $\hat{s} \leftarrow s, \hat{k} \leftarrow k$ $s \leftarrow s', k \leftarrow k'$ $x_0 \leftarrow x$ $x_i \ (1 \leq i \leq m) \leftarrow x'_i \ (1 \leq i \leq m')$		$(s \Vert k' \Vert x'_{m'}) = M_S \oplus g_k(r \Vert x \Vert M_T)$ If $h(s) = k$, $k \leftarrow k'$ $x_m \leftarrow x'_{m'}$
		If $x = x_m$, Generate r_T $M_1 = f_k(r \Vert r_T)$ $M_2 = r_T \oplus x$
	$\xleftarrow{r,M_1,M_2}$	
Case 3: Search for $x = x_m$(or x_0) for which $M_1 = f_k(r \Vert (M_2 \oplus x))$ $r_T = M_2 \oplus x$ If $x = x_m$, $M_S = g_k(r \Vert r_T \Vert M_1) \oplus (s \Vert k' \Vert x'_{m'})$ If $x = x_0$, $M_S = g_{\hat{k}}(r \Vert r_T \Vert M_1) \oplus (\hat{s} \Vert k \Vert x_m)$		
	$\xrightarrow{r,M_S}$	
Update secrets for T $s \leftarrow s'$ $k \leftarrow k'$ $x_0 \leftarrow x$ $x_i \ (1 \leq i \leq m) \leftarrow x'_i \ (1 \leq i \leq m')$		$(s \Vert k' \Vert x'_{m'}) = M_S \oplus g_k(r \Vert r_T \Vert M_1)$ If $h(s) = k$, $k \leftarrow k'$ $x_m \leftarrow x'_{m'}$

Figure 9.1: RFID authentication and secret update

$x_0, x_1, \cdots, x_m$ for T to the entity via a secure channel. As a result, the entity can authenticate T a maximum of m times. However, the entity receiving the delegation right cannot update the tag secrets, as it does not know s.

Multiple delegations of a tag T are also possible. If S transfers the secret k and the identifiers $x_0, x_1, \cdots, x_m$ for T to multiple entities, then these entities can all authenticate T during the same limited period, that is, until $x = x_m$ is reached.

If the delegated tag T is queried m times, then S will need to update T's secret and identifiers and, if necessary, S can now delegate the right to query the tag again. Note that it is always necessary for S to update the tag secret and identifiers, since, as noted above, only S knows s.

Unlike delegation, tag ownership transfer means that the tag owner transfers all rights over the tag to a new owner [41, 45]. In order to achieve ownership transfer of T, S must transfer the secrets s and k and the identifiers $x_0, x_1, \cdots, x_m$ for T, along with any other necessary information, to the new owner via a secure channel. This transfer should only take place after the old owner has updated the secrets and identifiers for T, in order to protect its privacy against possible tracking by the new owner. The server of the new owner should also update the tag secrets after receiving them from the old owner, in order to protect its privacy against possible tracking by the old owner. This update needs to take place in an environment where there is no possibility of eavesdropping by the old owner. Once this is complete, only the server of the new owner will be able to authenticate T and update the secrets for T.

9.4 Analysis of the Protocol

9.4.1 Privacy and Security

The protocol proposed in section 9.2 involves performing a tag authentication (TA) process to authenticate a tag. When a tag is queried for the mth time, the server and tag update their shared secrets using the secret update (I) (SU$_1$) process. If SU$_1$ does not complete successfully, in the following session the secret update (II) (SU$_2$) process is performed. SU$_1$ and SU$_2$ make use of a key transfer protocol and involve mutual authentication.

Note that both TA (case 1) and SU$_1$ (case 2) are 'normal' cases of the protocol, but SU$_2$ (case 3) will only occur if the protocol fails to operate as it should. This case arises if a message transfer in SU$_1$ fails.

The security of the protocol relies on the tag secrets k and s and the hash functions e, f, g and h. Under the assumption that the l-bit key k is a strong cryptographic key for e, f and g, an exhaustive search to find the l-bit values s and x is computationally infeasible. Also, given that hash functions e, f, g and h are one-way and collision-resistant, as assumed in section 3.3.2, the protocol has the following privacy and security properties.

- Tag Information Privacy (P1): we assume that the server database is secure. Thus only the server that has the secrets related to a tag can identify the tag and access the tag information.
- Tag Location Privacy (P2): a tag reply (x, M_T) is anonymous to an eavesdropper that does not know k, because x is updated to $e_k(x)$ in every query and M_T depends on x. A tag reply (M_1, M_2) in SU$_2$ is also anonymous to an eavesdropper, because M_1 and M_2 are computed using the key k and a session secret r_T. As a result, an adversary cannot track the location of a tag simply by eavesdropping on tag messages.

The protocol resists the following attacks feasible for a WA.

- Tag Impersonation (W1): to impersonate a tag, a WA needs to compute x and M_T (or M_1 and M_2). However, a WA cannot compute them without knowing k.
- Replay Attack (W2): a WA cannot reuse messages used in previous sessions because each response is a cryptographic function of a fresh random number. More specifically, M_T and M_S in TA and SU$_1$ depend on r, and M_1, M_2 and M_S in SU$_2$ depend on r and r_T.
- MitM Attack (W3): a WA cannot interfere with the exchanged messages by inserting or modifying messages, because of the use of the secrets k and s and random numbers r and r_T.

- DoS Attack (W4): if the second or third message in SU$_1$ is blocked, SU$_2$ will be performed in the following session. If the third message M_S in SU$_2$ is blocked, the server and tag will become desynchronised, because the server will update the shared secrets but the tag will not. However, in the next session, the server will detect such an event, because the tag will send as identifier the value x_0 in the server's look-up table. The server can thus recover synchronisation with the tag.

We next consider the degree to which the protocol can resist the security threats posed by an SA, identified in section 4.1.

- Backward Traceability (S1): one significant feature of the protocol is that, when $x = x_i$ in TA, M_T is computed as a function of x_{i-1}. As a result, it is difficult for an SA to trace transactions in previous sessions except for the immediately previous session in which x_i is included in the tag reply. An SA could intercept a tag identifier from a previous transaction, and compute the compromised identifier x by iteratively applying keyed hash function e to the previous identifier. However, the previous transactions were anonymous to the attacker at that time. Thus, in practice, tracing past transactions will not be simple. Obviously, if tag past transactions were computed using keys different from the compromised key k, it will be infeasible for an SA to trace them, because the previous keys will have no relation to the key k.
- Forward Traceability (S2): an SA can trace future transactions in which the compromised key is used. However, once the server and the compromised tag update their shared secrets, and assuming that the SA does not intercept the value of M_S sent from the server, it will not be able to compute the updated secrets and thus will no longer be able to trace tag transactions. Therefore, a server should immediately replace the tag secrets if it suspects that a tag may have been compromised.
- Server Impersonation (S3): an SA could try to update the secrets of a target tag by impersonating a legitimate server. If such an attack was possible, then the legitimate server would no longer be able to identify the tag, whereas the attacker would. One advantage of the protocol is that such a server impersonation attack is not straightforward. The reason for this is that an SA

cannot compute M_S just by compromising a tag, because s is known only by the server. An SA must perform a more sophisticated attack in which it intercepts M_S in order to learn s.

Table 9.2 summarises the protocol's privacy and security properties, and compares the protocol to seven of the previously proposed schemes introduced in chapter 5. These seven protocols have been selected for comparison since they and the protocol proposed here all are designed to take only $O(1)$ work to identify a tag, using a precomputed look-up table, when they are in regular states.

Table 9.2: Privacy and security properties

	P1	P2	W1	W2	W3	W4	S1	S2	S3
HM	√	·	·	·	·	√	·	·	·
D	√	·	√	√	√	·	√	·	·
LK	√	·	√	√	√	√	√	∗	∗
T1	√	√	√	√	√	·	·	·	·
T2	√	√	√	√	√	·	·	·	·
T3	√	√	√	√	√	·	·	·	·
BMM	√	·	√	√	√	√	·	·	·
TA	√	√	√	√	√	√	∗	∗	∗
SU_1	√	√	√	√	√	√	∗	∗	∗
SU_2	√	√	√	√	√	√	√	∗	∗

√ : resists such an attack

∗ : partially resists such an attack, under certain assumptions

· : does not protect against such an attack

9.4.2 Performance

The protocol proposed in section 9.2 has the following performance characteristics.

- Scalability: a server uses a look-up table for tag identification. As a result, a server can match a received anonymous identifier to a tag using its look-up table in $O(1)$ time, without needing a linear search. The protocol is scalable in the sense that a server only takes constant time to authenticate a tag, and tag delegation is straightforward, as stated in section 9.3. However, if a tag is queried more than m times without updating the tag secrets (case 3), the tag will reply with M_1 and M_2, and in this case the server needs to perform a linear search to authenticate the tag.

- Computation: in normal operation, i.e. when using TA and SU$_1$, a tag does not need to generate any pseudo-random numbers. However, in SU$_2$, a tag needs to generate a pseudo-random number in order to resist being traced. A tag needs to perform two hash function computations in the most common case (TA), four hash function computations in SU$_1$, and three hash function computations in SU$_2$. A server performs only one hash function computation in TA. In SU$_1$ and SU$_2$, a server must perform m' hash function computations in order to generate a new secret and new identifiers for a tag; fortunately these values can be precomputed.

- Communication: TA involves only two messages. SU$_1$ and SU$_2$ require one additional message.

- Storage Capacity: a tag needs a long term memory of $3l$ bits to store k, x and x_m.

The performance of the protocol is compared to the prior art in Table 9.3. The comparison shows that the performance of the proposed protocol compares favourably with existing schemes.

Table 9.3: Performance properties

	C1	C2	C3	C4
HM	ID, TID, LST	3HF	0	3
D	ID	4HF	1	3
LK	s, w_T, c, m	4PRF	1	3
T1	k, t, t_m	1HF	0/1	2
T2	k, t, t_m	2HF	1/2	2
T3	k, t, t_m	(γ+2)HF	1/3	2
BMM	k, τ, q, b, c	1/2PRF	0	3
TA	k, x, x_m	2HF	0	2
SU$_1$		4HF	0	3
SU$_2$		3HF	1	3

C1 : The type of secrets stored in a tag
C2 : The type and number of cryptographic computations required in a tag
C3 : The number of pseudo-random numbers required in a tag
C4 : The number of exchanged messages

9.5 Summary

The main contribution of this chapter is to propose a scalable RFID pseudonym protocol that meets the identified privacy and security requirements. The protocol has two features supporting scalability; a server takes only $O(1)$ work for tag authentication, and tag delegation is straightforward.

The protocol is divided into regular and irregular states. The regular state has two variants: tag authentication and secret update (I). In both cases, the server takes constant time to authenticate a tag. An irregular state occurs if the secret update (I) process fails. In such a case, the secret update (II) process is required. This process applies the protocol proposed in chapter 7.

We have compared the protocol to previously proposed scalable RFID protocols which take only constant time to identify a tag using a look-up table. The proposed scalable protocol is practical and provides the identified privacy and security properties. Tag authentication involves only two messages, and the secret update (I) and (II) procedures require only one more message. A tag does not need to generate any pseudo-random numbers in a regular session. The computational and storage loads on a tag are reasonable by comparison with the prior art. A server can perform in advance the most complex computations needed to update the shared secrets. In addition, the scheme protects against the identified privacy threats and weak attacks, and also provides partial security against the identified strong attacks.

Chapter 10

Conclusions

Contents

In this chapter we summarise the contributions of the thesis and identify directions for future research.

10.1 Research Achievements

This thesis is concerned with RFID authentication protocols that use symmetric cryptographic techniques to protect messages exchanged over an insecure radio frequency interface.

The main achievements of this research can be summarised as follows.

- In chapter 4 we have established privacy, security and performance requirements for RFID protocols, as well as functional requirements. The privacy requirements cover tag information privacy and tag location privacy. The security requirements cover resistance to both weak and strong attacks, that is, resistance to tag impersonation, replay attacks, man-in-the-middle attacks, denial-of-service attacks, backward traceability, forward traceability and server impersonation. The performance requirements relate to tag storage capacity, tag computation, communication and scalability. The additional functional requirements involve tag delegation and tag ownership transfer.

- In chapter 5 we have reviewed a total of 20 recently proposed RFID identification and authentication protocols. We have also assessed their privacy, security and performance properties against the requirements identified in chapter 4. We have assessed whether or not they provide the identified privacy and security properties. We have also summarised their performance properties; more specifically, we have evaluated the type of secrets stored in a tag, the type and number of cryptographic function computations required in a tag, the number of pseudo-random numbers that need to be generated in a tag, the number of exchanged messages and the complexity of the server computations to identify and authenticate a tag.

- In chapter 6 we have introduced the concept of server impersonation as a novel type of strong attack. A server impersonation based desynchronisation attack is a feasible security threat because RFID tag memory is typically not tamper-resistant. We have shown how, in cases where tag memory has been compromised, certain previously proposed RFID protocols can be desynchronised by a server impersonation attack; such an attack is relatively straightforward to perform on the MH, D and CC schemes, but more difficult for the LK scheme because of its use of an authentication key known to the server but not the tag. We have also proposed possible countermeasures designed to make an RFID protocol more resistant to such server impersonation attacks — one problem is that implementing these measures might increase tag cost. That is, we have a trade-off between security and cost.

- In chapter 7 we have proposed a new RFID authentication protocol designed to meet the requirements identified in chapter 4. The novel protocol has been compared with existing schemes with respect to both its privacy and security properties and its storage and computational requirements. The comparisons have shown that the proposed protocol is both more secure than previously proposed schemes and has practical advantages over them, because it provides the greatest number of identified privacy and security features and requires the least tag-side storage.

- In chapter 8 we have discussed tag ownership transfer. We first identified three requirements for secure and privacy-preserving tag ownership transfer:

new owner privacy, old owner privacy, and authorisation recovery. We next proposed novel RFID authentication protocols for tag ownership transfer that meet these requirements. The scheme consists of three protocols: an authentication protocol ($\mathbf{P}_1$), a secret update protocol ($\mathbf{P}_2$), and an authorisation recovery protocol ($\mathbf{P}_3$). The scheme proposed in chapter 7 is used for $\mathbf{P}_1$. We believe that $\mathbf{P}_3$ is the first proposed practical authentication scheme for authorisation recovery. We have also analysed and compared $\mathbf{P}_2$ and $\mathbf{P}_3$ to the prior art. The schemes satisfy the privacy and security requirements identified in chapter 4. $\mathbf{P}_2$ and $\mathbf{P}_3$ have desirable performance characteristics; a tag needs less non-volatile memory than in previously proposed schemes, performs just three hash function computations, and the number of messages exchanged between the tag and the server is only two, with mutual authentication. $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ also provide all the identified requirements for tag ownership transfer.

- In chapter 9 we have considered scalability issues in RFID systems. We have proposed a scalable RFID pseudonym protocol that meets the identified privacy and security requirements. The protocol has two features supporting scalability; a server takes only $O(1)$ work for tag authentication, and tag delegation is straightforward. The protocol is divided into regular and irregular states. The regular state has two variants: tag authentication and secret update (I). In both cases, the server takes constant time to authenticate a tag. An irregular state occurs if the secret update (I) process fails. In such a case, the secret update (II) process is required. This process applies the protocol proposed in chapter 7. We have compared the protocol to previously proposed scalable RFID protocols which take only constant time to identify a tag using a look-up table. The proposed scalable protocol is practical; tag authentication involves only two messages, and the secret update (I) and (II) procedures require only one more message; a tag does not need to generate any pseudorandom numbers in a regular session; the computational and storage loads on a tag are reasonable by comparison with the prior art; a server can perform in advance the most complex computations needed to update the shared secrets. In addition, the scheme protects against the identified privacy threats and weak attacks, and also provides partial security against the identified strong attacks.

10.2 Future Work

There are many possible topics for further study within the general area of RFID security. We mention four possible directions relating to work described in this thesis.

- This thesis only considers RFID protocols using symmetric cryptography, and primarily those using hash functions. There are a number of other general classes of RFID protocols that merit further study, including RFID protocols using asymmetric cryptography. There could also be many attacks on RFID systems that we have not identified. Thus, further study of such protocols and possible attacks on them would be desirable.
- We have assumed that the channel between the back-end server and the reader is secure. Hence, we have not dealt with security threats arising on that channel. However, in some applications, server-reader communications may be insecure, e.g. they may use a wireless channel. Thus, secure authentication protocols over this channel should be studied further.
- We have studied 'general' RFID protocols, i.e. protocols that can be used in any application. However, each RFID application has its own specific requirements. For example, RFID systems for E-passport, library book management and the supply chain have different security and performance requirements. Thus, it would be helpful to develop requirements for specific applications, and to specify RFID protocols to meet these requirements.
- We have not provided security proofs for the protocols proposed in this thesis. Work on formal models for RFID systems is only just emerging [2, 8, 12, 32, 77]. Thus, devising appropriate formalism for use in specifying and analysing RFID protocols remains a challenging and potentially fruitful topic.

Bibliography

[1] S. Aissi, N. Dabbous, and A. R. Prasad. *Security for Mobile Networks and Platforms*. Universal Personal Communications. Artech House, Norwood, MA, USA, 2006.

[2] G. Avoine. *Cryptography in Radio Frequency Identification and Fair Exchange Protocols*. PhD thesis, Ecole Polytechnique Federale de Lausanne (EPFL), Lausanne, Switzerland, December 2005.

[3] G. Avoine, E. Dysli, and P. Oechslin. Reducing time complexity in RFID systems. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography — SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 291–306, Kingston, Canada, August 2005. Springer-Verlag.

[4] G. Avoine and P. Oechslin. A scalable and provably secure hash based RFID protocol. In *International Workshop on Pervasive Computing and Communication Security — PerSec 2005*, pages 110–114, Kauai Island, Hawaii, USA, March 2005. IEEE Computer Society Press.

[5] H. Bar-El. Introduction to side channel attacks. White paper, Discretix Technologies Ltd., October 2002.

[6] A. Bondi. Characteristics of scalability and their impact on performance. In *the 2nd International Workshop on Software and Performance — WOSP 2000*, pages 195–203, Ottawa, Ontario, Canada, September 2000. ACM Press.

[7] M. Burmester, B. de Medeiros, and R. Motta. Anonymous RFID authentication supporting constant-cost key-lookup against active adversaries. *Journal of Applied Cryptography*, 1(2):79–90, 2008.

[8] M. Burmester, T. van Le, and B. de Medeiros. Provably Secure Ubiquitous Systems: Universally Composable RFID Authentication Protocols. In *the 2nd IEEE/CreateNet International Conference on Security and Privacy for Emerging Areas in Communication Networks — SecureComm 2006*, pages 1–9, Baltimore, Maryland, USA, August 2006. IEEE.

[9] S. Cai, Y. Li, T. Li, and R. Deng. Attacks and Improvements to an RFID Mutual Authentication Protocol and its Extensions. In *Second ACM Conference on Wireless Network Security — WiSec'09*, pages 51–58, Zurich, Switzerland, March 2009. ACM Press.

[10] H. Chien and C. Chen. Mutual authentication protocol for RFID conforming to EPC class 1 generation 2 standards. *Computer Standards & Interfaces*, 29(2):254–259, February 2007.

[11] Y. Choi, M. Kim, T. Kim, and H. Kim. Low power implementation of SHA-1 algorithm for RFID system. In *IEEE Tenth International Symposium on Consumer Electronics — ISCE '06*, pages 1–5, St.Petersburg, Russia, September 2006. IEEE.

[12] I. Damgård and M. Østergaard. RFID Security: Tradeoffs between Security and Efficiency. Cryptology ePrint Archive, Report 2006/234, 2006.

[13] T. Dimitriou. A lightweight RFID protocol to protect against traceability and cloning attacks. In *Conference on Security and Privacy for Emerging Areas in Communication Networks — SecureComm 2005*, pages 59–66, Athens, Greece, September 2005. IEEE.

[14] D. N. Duc, J. Park, H. Lee, and K. Kim. Enhancing security of EPCglobal gen-2 RFID tag against traceability and cloning. In *Symposium on Cryptography and Information Security — SCIS 2006*, Hiroshima, Japan, January 2006. The Institute of Electronics, Information and Communication Engineers.

[15] EPCglobal. *EPC Standard Specification, version 1.1 rev. 1.24*, April 2004.

[16] EPCglobal. *EPC^{TM} Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocols for Communications at 860 MHz – 960 MHz, Version 1.2.0*, October 2008.

[17] M. Feldhofer. *Low-Power Hardware Design of Cryptographic Algorithms for RFID Tags*. PhD thesis, Graz University of Technology, Institute for Applied Information Processing and Communications (IAIK), Graz, Austria, November 2008.

[18] M. Feldhofer and C. Rechberger. A case against currently used hash functions in RFID protocols. In R. Meersman, Z. Tari, and P. Herrero et al., editors, *On the Move to Meaningful Internet Systems 2006 — OTM 2006 Workshops*, volume 4277 of *Lecture Notes in Computer Science*, pages 372–381. Springer-Verlag, November 2006.

[19] K. Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. Wiley, second edition, 2003.

[20] S. Fouladgar and H. Afifi. An efficient delegation and transfer of ownership protocol for RFID tags. In *First International EURASIP Workshop on RFID Technology*, Vienna, Austria, September 2007.

[21] S. Fouladgar and H. Afifi. A simple privacy protecting scheme enabling delegation and ownership transfer for RFID tags. *Journal of Communications*, 2(6):6–13, November 2007.

[22] S. Garfinkel, A. Juels, and R. Pappu. RFID Privacy: An Overview of Problems and Proposed Solutions. *IEEE Security and Privacy*, 3(3):34–43, May-June 2005.

[23] B. Glover and H. Bhatt. *RFID Essentials*. O'Reilly, Gravenstein Highway North, Sebastopol, CA, USA.

[24] T. Haver. Security and privacy in RFID applications. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway, June 2006.

[25] A. Henrici and P. Müller. Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. In R. Sandhu and R. Thomas, editors, *International Workshop on Pervasive Computing and Communication Security — PerSec 2004*, pages 149–153, Orlando, Florida, USA, March 2004. IEEE Computer Society.

[26] S. Holloway. RFID: An Introduction. Technical report, Microsoft Developer Network, 2006.

[27] International Organization for Standardisation, Genève, Switzerland. *ISO 7498-2: 1989, Information processing systems — Open systems Interconnection — Basic reference model — Part 2: Security arichitecture*, 1989.

[28] A. Juels. Minimalist Cryptography for Low-Cost RFID Tags. In C. Blundo and S. Cimato, editors, *International Conference on Security in Communication Networks — SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 149–164, Amalfi, Italia, September 2004. Springer-Verlag.

[29] A. Juels. RFID security and privacy: A research survey. *IEEE Journal on Selected Areas in Communications*, 24:381–394, February 2006.

[30] A. Juels, D. Molnar, and D. Wagner. Security and Privacy Issues in E-passports. In *Conference on Security and Privacy for Emerging Areas in Communications Networks — SecureComm 2005*, pages 74–88, Athens, Greece, September 2005. IEEE.

[31] A. Juels and S. Weis. Authenticating Pervasive Devices with Human Protocols. In Victor Shoup, editor, *Advances in Cryptology — CRYPTO'05*, volume 3126 of *Lecture Notes in Computer Science*, pages 293–308, Santa Barbara, California, USA, August 2005. Springer-Verlag.

[32] A. Juels and S. Weis. Defining Strong Privacy for RFID. In *International Conference on Pervasive Computing and Communications — PerCom 2007*, pages 342–347, New York City, New York, USA, March 2007. IEEE Computer Society Press.

[33] S. Karthikeyan and N. Nesterenko. RFID security without extensive cryptography. In *Workshop on Security of Ad Hoc and Sensor Networks — SASN '05*, pages 63–67, Alexandria, Virginia, USA, November 2005. ACM Press.

[34] F. Kerschbaum and A. Sorniotti. RFID-Based Supply Chain Partner Authentication and Key Agreement. In *Second ACM Conference on Wireless Network Security — WiSec'09*, pages 41–50, Zurich, Switzerland, March 2009. ACM Press.

[35] J. F. Korsh. *Data Structures, Algorithms and Program Style.* PWS Publishing Co., Boston, MA, USA, 1986.

[36] J. Landt. Shrouds of time: The history of RFID. 1 October 2001. http://www.rfidconsultation.eu/docs/ficheiros/shrouds_of_time.pdf.

[37] A. Laurie. Practical attacks against RFID. *Network Security*, 2007(9):4–7, September 2007.

[38] Y. Li and X. Ding. Protecting RFID Communications in Supply Chains. In *the 2nd ACM Symposium on Information, Computer and Communications Security — ASIACCS '07*, pages 234–241, Singapore, Republic of Singapore, 2007. ACM Press.

[39] I. Liersch. Electronic passports — from secure specifications to secure implementations. *Elsevier Information Security Technical Report*, 14(2):96–100, May 2009.

[40] C. Lim and T. Korkishko. mCrypton — A Lightweight Block Cipher For Security of Low-Cost RFID Tags and Sensors. In J. Song, T. Kwon, and M. Yung, editors, *Workshop on Information Security Applications — WISA'05*, volume 3786 of *Lecture Notes in Computer Science*, pages 243–258, Jeju Island, South Korea, August 2005. Springer-Verlag.

[41] C. Lim and T. Kwon. Strong and robust RFID authentication enabling perfect ownership transfer. In P. Ning, S. Qing, and N. Li, editors, *Conference on Information and Communications Security — ICICS '06*, volume 4307 of *Lecture Notes in Computer Science*, pages 1–20, Raleigh, North Carolina, USA, December 2006. Springer-Verlag.

[42] K. Mayes, K. Markantonakis, and G. Hancke. Transport ticketing security and fraud controls. *Elsevier Information Security Technical Report*, 14(2):87–95, May 2009.

[43] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*, volume 6 of *Discrete Mathematics and Its Applications*. CRC Press, 1996.

[44] C. J. Mitchell. Cryptography for mobile security. In C. J. Mitchell, editor, *Security for Mobility*, IET Telecommunications, chapter 1, pages 3–10. The Institution of Engineering and Technology, December 2003.

[45] D. Molnar, A. Soppera, and D. Wagner. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography — SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 276–290, Kingston, Canada, August 2005. Springer-Verlag.

[46] D. Molnar and D. Wagner. Privacy and security in library RFID: Issues, practices, and architectures. In B. Pfitzmann and P. Liu, editors, *Conference on Computer and Communications Security — ACM CCS*, pages 210–219, Washington, DC, USA, October 2004. ACM Press.

[47] P. Najera and J. Lopez. RFID: Technological Issues and Privacy Concerns. In A. Acquisti, S. Gritzalis, C. Lambrinoudakis, and S. di Vimercati, editors, *Digital Privacy: Theory, Technologies and Practices*, chapter 14, pages 285–306. Taylor & Francis Group, 2008.

[48] D. W. Nance and T. L. Naps. *Introduction to Computer Science: Programming, Problem Solving and Data Structures.* West Publishing Company, second edition, 1992.

[49] M. Ohkubo, K. Suzki, and S. Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *RFID Privacy Workshop*, MIT, MA, USA, November 2003. `http://www.rfidprivacy.us/2003/agenda.php`.

[50] K. Osaka, T. Takagi, K. Yamazaki, and O. Takahashi. An efficient and secure RFID security method with ownership transfer. In Y. Wang, Y. Cheung, and H. Liu, editors, *Computational Intelligence and Security — CIS 2006*, volume 4456 of *Lecture Notes in Computer Science*, pages 778–787. Springer-Verlag, September 2006.

[51] K. Ouafi and R. C.-W. Phan. Traceable Privacy of Recent Provably-Secure RFID Protocols. In S.M. Bellovin, R. Gennaro, A. Keromytis, and M. Yung, editors, *6th International Conference on Applied Cryptography and Network Security — ACNS 2008*, volume 5037 of *Lecture Notes in Computer Science*, pages 479–489, New York City, New York, USA, June 2008. Springer-Verlag.

[52] P. Peris-Lopez, J. Hernandez-Castro, J. Estevez-Tapiador, and A. Ribagorda. Cryptanalysis of a novel authentication protocol conforming to EPC-C1G2 standard. *Computer Standards & Interfaces*, 31(2):372–380, 2009.

[53] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda. An efficient authentication protocol for RFID systems resistant to active attacks. In M. K. Denko, C. Shih, K. Li, S. Tsao, Q. Zeng, S. Park, Y. Ko, S. Hung, and J. Park, editors, *Emerging Directions in Embedded and Ubiquitous Computing — EUC 2007 Workshops: SecUbiq*, volume 4809 of *Lecture Notes in Computer Science*, pages 781–794, Taipei, Taiwan, December 2007. Springer-Verlag.

[54] R. D. Pietro and R. Molva. Information confinement, privacy, and security in RFID systems. In J. Biskup and J. Lopez, editors, *European Symposium Research Computer Security — ESORICS 2007*, volume 4734 of *Lecture Notes in Computer Science*, pages 187–202, Dresden, Germany, September 2007. Springer-Verlag.

[55] N. Pramstaller, C. Rechberger, and V. Rijmen. A compact FPGA implementation of the hash function Whirlpool. In *ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays — FPGA'06*, pages 159–166, New York, 2006. ACM Press.

[56] B. Preneel et al. Final report of European project IST-1999-12324: New European schemes for signatures, integrity, and encryption. `http://www.cosic.esat.kuleuven.be/nessie/`, April 2004.

[57] M. Rieback, B. Crispo, and A. Tanenbaum. The Evolution of RFID Security. *IEEE Pervasive Computing*, 5(1):62–69, January–March 2006.

[58] R. Roman, C. Alcaraz, and J. Lopez. A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. *Mobile Networks and Applications*, 12(4):231–244, 2007.

[59] J. Saito, K. Imamoto, and K. Sakurai. Reassignment Scheme of an RFID Tag's Key for Owner Transfer. In T. Enokido, L. Yan, B. Xiao, D. Kim, Y. Dai, and L.T. Yang, editors, *Emerging Directions in Embedded and Ubiquitous Computing — EUC 2005 Workshops*, volume 3823 of *Lecture Notes in Computer Science*, pages 1303–1312. Springer-Verlag, November 2005.

[60] B. Schneier. *Applied Cryptography: Protocols, Algorithems, and Source Code in C.* John Wiley & Sons, Inc., New York, NY, USA, 1996.

[61] A. Shamir. SQUASH — A New MAC with Provable Security Properties for Highly Constrained Devices Such as RFID Tags. In K. Nyberg, editor, *Fast Software Encryption: 15th International Workshop — FSE 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*, pages 144–157, Lausanne, Switzerland, February 2008. Springer-Verlag.

[62] W. Shieh and J. Wang. Efficient remote mutual authentication and key agreement. *Computer & Security*, 25(1):72–77, 2006.

[63] B. Song. RFID Tag Ownership Transfer. In *Workshop on RFID Security — RFIDSec 08*, Budapest, Hungary, July 2008.

[64] B. Song. Server Impersonation Attacks on RFID Protocols. In *Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies — UBICOMM 08*, pages 50–55, Valencia, Spain, October 2008. IEEE Computer Society.

[65] B. Song and C. J. Mitchell. RFID authentication protocol for low-cost tags. In V. D. Gligor, J. Hubaux, and R. Poovendran, editors, *ACM Conference on Wireless Network Security — WiSec '08*, pages 140–147, Alexandria, Virginia, USA, April 2008. ACM Press.

[66] B. Song and C. J. Mitchell. Scalable RFID Pseudonym Protocol. In *3rd International Conference on Network & System Security — NSS 2009*, pages 216–224, Gold Coast, Queensland, Australia, October 2009. IEEE Computer Society.

[67] B. Song and C. J. Mitchell. Scalable RFID Security Protocols supporting Tag Ownership Transfer. *Computer Communications*, submitted, 2009.

[68] W. Stallings. *Cryptography and Netwrok Security: Principles and Practice.* Prentice Hall, Upper Saddle River, New Jersey, second edition, 1999.

[69] D. Stinson. *Cryptography: Theory and Practice.* CRC Press, Boca Raton, Florida, second edition, 2002.

[70] H. Stockman. Communication by means of reflected power. *Proceedings of The Institute of Radio Engineers*, 36(10):1196–1204, October 1948.

[71] F. Thornton, B. Haines, A. M. Das, H. Bhargava, A. Campbell, and J. Kleinschmidt. *RFID Security.* Syngress, Massachusetts, USA, 2006.

[72] G. Tsudik. YA-TRAP: Yet another trivial RFID authentication protocol. In *Fourth IEEE Annual Conference on Pervasive Computing and Communications — PerCom 2006*, pages 640–643, Pisa, Italy, March 2006. IEEE Computer Society.

[73] G. Tsudik. A family of dunces: Trivial RFID identification and authentication protocols. In N. Borisov and P. Golle, editors, *Privacy Enhancing Technologies, 7th International Symposium — PET 2007*, volume 4776 of *Lecture Notes in Computer Science*, pages 45–61, Ottawa, Canada, June 2007. Springer-Verlag, Berlin.

[74] István Vajda and Levente Buttyán. Lightweight authentication protocols for low-cost RFID tags. In *Second Workshop on Security in Ubiquitous Computing — Ubicomp 2003*, Seattle, WA, USA, October 2003.

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

[76] T. van Le, M. Burmester, and B. de Medeiros. Universally composable and forward-secure RFID authentication and authenticated key exchange. In R. Deng and P. Samarati, editors, *ACM Symposium on information, Computer and Communications Security — ASIACCS '07*, pages 242–252, New York, NY, USA, March 2007. ACM Press.

[77] S. Vaudenay. On Privacy Models for RFID. In K. Kurosawa, editor, *Advances in Cryptology — Asiacrypt 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87, Kuching, Malaysia, December 2007. Springer-Verlag.

[78] R. D. Vines. *Wireless Security Essentials: Defending Mobile Systems from Data Piracy.* Wiley, August 2002.

[79] S. Weis, S. Sarma, R. Rivest, and D. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In D. Hutter, G. Müller, W. Stephan, and M. Ullmann, editors, *International Conference on Security in Pervasive Computing — SPC 2003*, volume 2802 of *Lecture Notes in Computer Science*, pages 201–212, Boppard, Germany, March 2003. Springer-Verlag.

[80] K. Yüksel. Universal hashing for ultra-low-power cryptographic hardware applications. Master's thesis, Dept. of Electronical Engineering, Worcester Polytechnic Institute, Worcester, MA, USA, 2004.

[81] Y. Zhang and P. Kitsos. *Security in RFID and Sensor Networks.* Auerbach Publications, April 2009.