University of Wollongong
Research Online

University of Wollongong Thesis Collection

University of Wollongong Thesis Collections

2012

Contributions to RFID security

Ching Yu Ng
University of Wollongong

Recommended Citation

Ng, Ching Yu, Contributions to RFID security, Doctor of Philosophy thesis, School of Computer Science and Software Engineering, University of Wollongong, 2012. http://ro.uow.edu.au/theses/3549

Research Online is the open access institutional repository for the University of Wollongong. For further information contact Manager Repository Services: morgan@uow.edu.au.

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Contributions to RFID Security

A thesis submitted in fulfillment of the
requirements for the award of the degree

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Ching Yu Ng

Computer and Information Security Research Lab
School of Computer Science and Software Engineering
May 2012

© Copyright 2012

by

Ching Yu Ng

All Rights Reserved

Dedicated to
My Dad, my Mum & my Wife

Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

Ching Yu Ng
May 10, 2012

Abstract

This thesis aims to create a secure and practical RFID security framework, particularly on providing an adequate privacy model and an adversary model for RFID applications where an authentication protocol for RFID tag is required. Our framework can be used to assess the performance of RFID authentication protocols that are conformed to a common system model. We first look into other proposed privacy models and compares their performances. We investigate their limitations on modelling some types of RFID authentication protocols. Our privacy model defines what we want to achieve when providing privacy protections to RFID systems. Examples like what we call a secure system, what are the privacy goals and how we test an RFID authentication protocols are defined in our privacy model. Our adversary model addresses the abilities of adversaries that cause harm to RFID systems. Its purpose is to capture the most common and possible attacks to RFID systems that can be launched in the real world. These modellings together provide us an effective tool to look into the limitations and possibilities of the RFID authentication protocol begin assessed. Based on our security framework, we give an example RFID application on ownership transfer.

Acknowledgements

I would like to thank Dr. Willy Susilo and Dr. Yi Mu, my advisors, for their patient guidance and constant support during my study. Many thanks for choosing me as their student. I admire their wealth of security and cryptographic knowledge, which inspired me a lot when I first entered into this extremely challenging research area. Their precious opinions and precise research directions were most valuable. I especially appreciate Dr. Willy Susilo's good memories on the publications of different researchers. I also appreciate Dr. Yi Mu's quick cryptanalysis skill, he can always point out the hidden vulnerabilities from my proposed cryptographic schemes. Finally I am grateful to my parents, without them, I will not be studying overseas.

Publications

1. Ching Yu Ng, Willy Susilo, Yi Mu and Rei Safavi-Naini. "RFID Privacy Models Revisited," in Sushil Jajodia and Javier Lopez (Eds.), Proceedings of the Thirteenth European Symposium on Research in Computer Security - ESORICS 2008, volume 5283 of Lecture Notes in Computer Science, pages 251-266, Malaga, Spain, October 2008. Springer.

2. Ching Yu Ng, Willy Susilo, Yi Mu and Rei Safavi-Naini. "New Privacy Results on Synchronized RFID Authentication Protocols against Tag Tracing," in Michael Backes and Peng Ning (Eds.), Proceedings of the Fourteenth European Symposium on Research in Computer Security - ESORICS 2009, volume 5789 of Lecture Notes in Computer Science, pages 321-336, Saint-Malo, France, September 2009. Springer.

3. Ching Yu Ng, Willy Susilo, Yi Mu and Rei Safavi-Naini. "Practical RFID Ownership Transfer Scheme," in Yingjiu Li and Jianying Zhou (Eds.), Proceedings of the Second Radio Frequency Identification System Security Asia Workshop - RFIDsec Asia 2010, volume 4 of Cryptology and Information Security Series, page 33-48, Singapore, February 2010. IOS Press.

4. Ching Yu Ng, Willy Susilo, Yi Mu and Rei Safavi-Naini. "Practical RFID Ownership Transfer Scheme," Journal of Computer Security – Special Issue on RFID System Security, 19(2):319-341, March 2011. IOS Press.

Contents

Abstract **v**

Acknowledgements **vi**

Publications **vii**

1 Introduction **1**

1.1 Motivations 1

1.2 Research Direction 4

1.3 Thesis Structure 5

2 RFID Basics **7**

2.1 Limitations of Low-Cost RFID Tags 7

2.1.1 A Basic RFID System 7

2.1.2 Security Dedication 8

2.2 Advances in RFID Technology 10

2.2.1 Advance in Data Processing 10

2.2.2 Advance in Data Communication 10

2.2.3 Advance in Data Storage 11

2.3 Security Threats Caused by the Advances 11

2.3.1 Threats Caused by Tag Collision 11

2.3.2 Threats Caused by RF Communication 13

2.3.3 Threats Caused by Storing Sensible Data 14

2.4 Example RFID Applications in Real World 15

2.4.1 ePassport 15

2.4.2 US Passport Card and Driver Licence 15

2.4.3 UK Transportation Card 16

2.5 Industrial Countermeasures 17

2.5.1 Countermeasure to Tag Collision Attack 17
2.5.2 Countermeasure to Unauthorised Tag Scan 18
2.5.3 Countermeasure to Tag Counterfeiting 20
2.6 Conclusion 21

3 RFID Protocols 22
3.1 Functionalities of RFID Protocols 22
3.1.1 Tag Identification 23
3.1.2 Access Control 23
3.1.3 Tag Authentication 24
3.2 Prerequisites for Secure RFID Protocols 25
3.3 Adversarial Attacks on RFID Protocols 26
3.3.1 Desynchronisation Attack 27
3.3.2 Man-In-The-Middle Attack 28
3.3.3 Relay Attack 31
3.3.4 Replay Attack 33
3.3.5 Timing Attack 33
3.4 Privacy in RFID Authentication Protocols 34
3.4.1 Personal Privacy 34
3.4.2 Location Privacy 35
3.4.3 Forward Privacy 36
3.5 Security Properties for RFID Authentication Protocols 37
3.5.1 Example Scenario 37
3.5.2 Anonymity 38
3.5.3 Unlinkability 39
3.5.4 Indistinguishability 39
3.6 Conclusion 40

4 Our Findings on RFID Privacy Modellings 42
4.1 RFID Privacy Models 43
4.2 Early Works on RFID Modellings 44
4.2.1 The Ari Juels Model 44
4.2.2 The Gildas Avoine Model 45
4.2.3 The Xiaolan Zhang and Brian King model 47
4.3 Inconsistencies in RFID Privacy Models 48
4.3.1 Untraceability in the Avoine Model 48

4.3.2 Indistinguishability in the Juels and Weis Model 51
4.3.3 Analysis of the Two Definitions 53
4.3.4 Comparison 53
4.4 A Strong Privacy Model for RFID by Vaudenay 55
4.4.1 System Model 55
4.4.2 Adversary Model 56
4.4.3 Privacy Experiment 57
4.4.4 Privacy Classes 57
4.5 Our Refined Vaudenay Privacy Model 58
4.5.1 Preliminaries 59
4.5.2 New Privacy Classification 60
4.6 Our Different Results From the Vaudenay Model 68
4.6.1 Strong Privacy is Possible 69
4.6.2 Truly Random Source is Required 70
4.6.3 Forward Privacy Without PKC 71
4.7 Conclusion 72

5 Family of Synchronised Authentication Protocols in RFID 74
5.1 Synchronised Authentication Protocols in RFID 75
5.1.1 Privacy Experiment 76
5.1.2 Achievable Privacy Levels 76
5.1.3 Protocol Constructions 77
5.2 Our New Privacy Results on SAPs 78
5.2.1 **Type 0** Protocols Can Never Achieve Forward Privacy Levels . 79
5.2.2 **Type 1** Protocols Can Never Achieve Non-Narrow Privacy Levels 81
5.2.3 **Type 2a** Protocols Can be Reduced to **Type 0** Protocols . . 83
5.2.4 **Type 2b** Protocols Can be Reduced to **Type 0** or **Type 1** Protocols 86
5.3 Seeking for a Better Solution 87
5.4 Our New RFID System Model 90
5.4.1 Core Assumptions 90
5.4.2 System Setup 91
5.4.3 RFID Authentication Protocol Construction 91
5.5 Our New RFID Adversary Model 92
5.5.1 Adversary Abilities 93
5.5.2 Privacy Experiment 95

5.6 Security Analysis of the Type R Construction 96
5.6.1 Type R Protocols Can Resist Querying-Attack 96
5.6.2 Type R Protocols Can Resist Blocking-Attack 97
5.6.3 An Example Protocol 98
5.7 Example Applications for Type R Protocols 100
5.8 Conclusion 101

6 Ownership Transfer Protocol in RFID 102
6.1 RFID Ownership Transfer 102
6.2 Motivation 103
6.2.1 Previous Works on RFID Ownership Transfer 104
6.2.2 Our Contributions 107
6.3 Preliminaries 108
6.3.1 System Model 108
6.3.2 Ownership Transfer Model 109
6.3.3 Basic Assumptions 111
6.3.4 Adversary Model 113
6.3.5 Security Properties 113
6.3.6 Building Blocks 114
6.4 Our Ownership Transfer Scheme 115
6.4.1 Setup 116
6.4.2 Key Change Protocol 117
6.4.3 Controlled Delegation Protocol 117
6.4.4 Ownership Transfer Protocol 118
6.4.5 Temporary Authorisation Recovery Protocol 119
6.5 Security Analysis 119
6.5.1 Previous Owner Privacy and New Owner Privacy 119
6.5.2 Controlled Delegation and Temporary Authorisation Recovery . 121
6.5.3 Tag Assurance 122
6.5.4 Current Ownership Proof and Undeniable Ownership Transfer . 123
6.5.5 Owner Initiation 123
6.6 Conclusions 124

7 Conclusion 125

Bibliography 126

List of Tables

2.1 Security primitives for low-cost RFID tags 9

4.1 Untraceability experiment in the Avoine privacy model 50
4.2 Indistinguishability experiment in the Juels and Weis privacy model . . 52
4.3 Simulation results of the `Result`(π) oracle 64

5.1 Achievable privacy levels of **Type 0** construction 81
5.2 Achievable privacy levels of **Type 1** construction 83
5.3 Achievable privacy levels of **Type 2a** construction 85
5.4 Achievable privacy levels of **Type 2b** construction 87

List of Figures

3.1 HB$^+$ Protocol 30
3.2 Distance Bounding Protocol 32
3.3 Personal privacy preserving protocol 35
3.4 Location privacy preserving protocol 35
3.5 Forward privacy preserving protocol 36

4.1 Protocol secure in the Juels and Weis model but not in the Avoine model 53
4.2 Relationship of the 8 privacy classes in the Vaudenay model 58
4.3 An example PKC RFID protocol 69
4.4 An example PKC RFID protocol providing strong privacy 70
4.5 An OSK protocol variant 71

5.1 **Type 0** SAP construction 79
5.2 **Type 1** SAP construction 81
5.3 **Type 2a** SAP construction 84
5.4 **Type 2b** SAP construction 86
5.5 Traditional three-round RFID mutual authentication protocol construction 91
5.6 **Type R** SAP construction 93
5.7 A simple **Type R** RFID mutual authentication protocol 98

6.1 Key change protocol 117
6.2 Controlled delegation protocol 118
6.3 Ownership transfer protocol 120
6.4 Temporary authorisation recovery protocol 121

Chapter 1

Introduction

Chapter Overview

In the first chapter of this thesis, we give an overall idea of what this thesis is all about. First a brief introduction to RFID security researches is discussed, which leads to the motivation behind that has driven our interest in choosing this as our research topic. Next we state what we have achieved in this work by giving a brief summary about the outcomes of our novel contributions to the RFID security research community. Part of these achievements have been composed into academic papers and published in world recognised conferences and journal. Finally, we give an overall structure of this thesis.

1.1 Motivations

The Problem

To use one sentence to describe about this thesis, this thesis is about "An investigation into the ability of low-cost RFID tag systems in providing an adequate level of privacy protections in theory". The first question to ask is why RFID? The use of *Radio frequency identification* (RFID) has gained its momentum since a few years back. In Australia, RFID services license has been granted to GS1 Australia by Australian Communication and Media Authority (ACMA), a government agency that governs the use of RF in Australia. EPC Global Australia has been working with GS1 Australia on the RFID standard in Australia. Some of the early RFID applications including livestock and pet identification and auto tolling for drivers have been using for many years in Australia. Other developing or potential applications include supplies and equipments tracking being used by Australian Defense Force (ADF), aircraft parts

tracking and verification, mail monitoring by Australia Post, books and CDs tracking in the State Library, RFID PayPass credit cards, ePassport, etc. Not to mention retails, logistics, supply chains where RFID is originally targeting. It is not long before Australian to adopt this technology into our daily life.

RFID is always being compared to the well accepted barcode systems. The very nature of both technologies is to identify (or authenticate in some applications) any objects from their digitalised attached information, which can be used to link up to other stored records in a back-end database for further referencing. Although they share the same nature, RFID on the other hand opens up many new possibilities that are missing in barcodes. These include scanning without line of sight, much longer reading range, batch processing etc. An object tagged with these tiny RFID tags can be scanned and identified within a distance using an RFID reader, be it legitimate or illegitimate. Such convenient technology has drawn people's attention but we also see a lot of concerns coming with it. For example, anyone with a compatible RFID reader can obtain a full reading of all the RFID tagged items from any passersby without their consent. The readings may provide immediate useful information like names, addresses, product types etc. The best example and the one that raised the most concern was the launch of electronic passports [45], which was when people became aware of their personal privacy could be in jeopardise if these tiny devices are to be infested into their everyday life.

This personal privacy concern can be separated into two categories: *Data privacy* and *Location privacy*. The former is relatively easy to protect, simply encrypt the data stored inside an RFID tag with a good encryption scheme and then manage the decryption key properly within the system, which is what the e-passport did, only that they have the key printed on the passport. The latter is a real challenge. Even if the scanned readings are pseudonyms or encrypted, these unique numbers together serve as a "personal number plate" for identifying any individuals. People's concerns are reasonable, especially when individual identities are bounded to these RFID tags. We have been using RFID chips to track and locate house pets and we see human implantation coming [44]. RFID technologies will not reach its full deployment so much as barcodes have done if this tracking problem is not solved. This gave us enough motivation to look into the matter deeper and answers the question why RFID.

The Challenge

The next question to ask is how difficult is the issue? RFID tag tracking clearly violates the location privacy of RFID tags bearers. By matching the collected radio signals sent from many RFID tags to a particular tag (or a batch of tags) using any rogue reader, the bearer's location, past and current, can be traced [14]. This privacy issue has been one of the main topics in RFID research. Like when people talk about the networks and attacks, you can always get an attack-free machine if you disconnect yourself from the networks. A pessimistic way to prevent tag tracking is to "kill" the tag by executing a form of deactivation command [81]. As pointed out in [75], this will only sacrifice the benefits and convenience provided by RFID technologies if we ever want to take advantage of any potential services that rely on live working tags. In order to keep the tags "alive" for any future uses while protecting the tags from being traced, it is essential to create a communication protocol between legitimate tags and readers that guarantees untraceability. In a real world application, RFID tags may emit distinguishable radio signals due to hardware manufacturing diversities, which allow simple tracing in the physical layer [11]. Of course, nothing could be built if it already fails at the lower layers. Therefore we assume in this work that this physical diversity has removed and focus ourselves on the protocol layer only.

If we consider RFID tags as some decent wireless devices, then we can easily port any already established industrial proprietary cryptosystems to be used in RFID and start preserving users' privacy. In fact, the issue is even worsen when low-cost RFID tags are in concerned. Due to pricing pressure, these low-cost RFID tags are characterised by their passive nature (batteryless), low computational power and non-tamper-proof. We see there are many ways to crack tamper resistant smart cards [1], so we are not relying our privacy protection on this. Because of their lack of computational power, public key cryptography is simply not feasible in these low-cost RFID tags, even standard implementation of AES means too expensive for them [78]. Leaving us only simple bitwise operations, XOR, CRC, and PRNG, etc. as the primitives to secure these tags. Fortunately, highly optimised AES [26, 28] and SHA [27, 66] implementations for RFID tags had recently become available, we can use them as the security building blocks. But still, their implementation footprints are marginally over the current mid-range priced tags. These altogether creates a very unique yet challenging environment for RFID privacy researches. If we look at the published RFID papers list maintained at [87], there are more than hundreds of proposed schemes aimed to solve this privacy issue in RFID. However, many times a scheme is proposed, there is/are papers to attack

it [6, 49, 90]. This shows the difficulties in trying to solve the privacy issue in low-cost RFID tags with their limited resources and the urge to restore privacy for them.

1.2 Research Direction

In order to assess the performance of various RFID protocols, a formal security framework is needed. Moreover, we see the needs to have a unified security formalisation for RFID to put a stop to the propose and then attack pattern due to the differences in security definition. Our research focuses on seeking a generally accepted formal security framework to the challenging low-cost RFID environment. A security framework consists of four parts:

- A system model that defines the components and the construction of the underlying RFID application.
- A security model that defines the security goals of the application and the experiments for privacy assessment.
- An adversary model that defines the abilities of the adversary.
- A protocol structure that has to be conformed.

We aware that we are not the first one to propose a security framework for RFID. With the many already proposed RFID security frameworks, we start our research by reviewing their effectiveness, in a sense how well they can capture the real world attacks in many different RFID application scenarios. Learning from each of their strengths, we modify them into our security framework. Then we use this security framework to test the performance of a family of RFID protocols, namely the *Synchronised Authentication Protocol* (SAP). We separate them into five different classes and assess them individually. Finally, we create an RFID application base on this security framework that conforms to one of the SAP classes. Hence, we have the following objectives:

1. Identify the security threats in RFID applications and the real world practices.
2. Point out the security protections that are absent in current RFID products.
3. Review selected RFID security frameworks and comment on their effectiveness.
4. Modify the reviewed frameworks to create a new one that suits our needs.

5. Classify current RFID protocols into types with similar structure.
6. Suggest the limitations of each of the RFID protocol types base on the modified security framework.
7. Propose a new type of RFID protocol that performs better in the modified security framework.
8. Create an example RFID application (RFID ownership transfer) base on the new protocol type.

1.3 Thesis Structure

This thesis is composed of 7 chapters. Chapter 1 (this chapter) gives an overall idea of what this thesis is about. First a brief introduction to the general state of current RFID researches is presented. The needs identified here have driven this work and the objectives are clearly stated.

Chapter 2 provides a basic understanding to the security threats of RFID from a high-level view. It compares RFID with barcodes to see the advance in this new technology and the source of the security concerns. Some real world applications are reviewed here and the industrial practices and countermeasures to the security threats are discussed.

Chapter 3 introduces RFID protocols. It lists the system requirements that is required in order to provide a secure RFID protocol. It also looks at most of the common adversarial goals and the security protections usually seen in RFID protocols.

Chapter 4 reviews some early notable works on formal RFID modelling. Out from the several modelling works, we selected the model by Vaudenay [93] and critically reviewed it. We obtained some inconsistence results to the Vaudenay model and justified them here. We also suggested a fix to this model and removed some redundancies to give a more compact model under reasonable assumptions. These contributions were put together into a paper and published in [70].

Chapter 5 presents a family of RFID authentication protocols with similar structures. We created four classifications for this family of protocols and assessed their performance with the modified Vaudenay model obtained from chapter 4. We identified their impossibilities in achieving some certain level of privacy protections. Then we go onto propose a fifth type of protocol structure and use it to create protocols

with better performance on privacy protections. These contributions were presented and published in [71].

Chapter 6 describes our RFID ownership transfer protocol that takes advantage of our fifth protocol structure to provide a secure and practical RFID application example. Our new ownership transfer protocol covered all of the identified security properties of other ownership transfer protocols and we added four new properties firstly proposed by us. This new protocol was presented and published in [72] and later extended into [73].

Chapter 7 concludes this thesis.

Chapter 2

RFID Basics

Chapter Overview

In this chapter, we will give more backgrounds about the potential security threats that could imperil the adoption of RFID into our daily life. First we provide a basic understanding to the physical limitations of low-cost RFID tags. A comparison between RFID and barcodes is given to clearly show the source of the security threats. Even with all these issues in RFID, we are beginning to see RFID being embedded into sensitive documents like passports, credit cards, driver licenses and transportation cards, etc. Not surprisingly, all these applications of RFID have their own security issues. We review some of these applications, namely the transportation cards in the UK (which use MIFARE chips) and the ePassport, passport cards and driver licenses in the US (which use EPC Gen-2 chips). We discuss existing practical attacks to these RFID products. We also summarise some of the best practices supplied by the homeland security department of US and the industry for the use of RFID. We draw important lessons that can be learnt from their experiences. Following that we discuss some industrial countermeasures.

2.1 Limitations of Low-Cost RFID Tags

2.1.1 A Basic RFID System

A basic RFID system consists of multiple tags, a reader and a back-end database server. Similar to barcodes, each tag contains a unique ID stored in a microchip and the microchip is attached to an antenna. Most of the RFID systems use low-cost RFID tags, usually below 5 cents per tag [81]. These tags are passive tags, which do not have

battery attached. They are powered by the strong RF signals emitted from the reader. Because of this, the responding signals from tags to reader are comparatively much weaker than the signals from reader to tags. Passive tags can be read within 6 metres and sometimes even up to 45 metres under certain circumstances. Generally, the read range will varies depends on the power output of the reader and the environment. RF signal eavesdropping is believed to be elementary in the reader to tags communication for both close proximity and long distance eavesdroppers; while it is usually harder for the long distance eavesdroppers to eavesdrop on the tags to reader communication. When an RFID reader broadcasts a query, all the surrounding tags are powered up by its RF signal and response with their unique IDs or other information that can be used as the unique pointers to their record in the back-end database. The reader will then use these pointers as the reference key to load up any further information with the help of the back-end database. It is not common for the tags to store any more information other than their unique IDs to minimise the possibility to breach personal privacy. Some of the first generation RFID embedded credit cards do contain the card holder's name and expiration date in clear text. ePassports store even more personal information like DOB, nationality and digitalised photo of the holder but in an encrypted form. A privacy preserving application, on the other hand, will avoid storing unique IDs as that will lead to easy tag tracking.

2.1.2 Security Dedication

Passive RFID tags are always equipped with very limited computational power and very limited logic gates in the microchip dedicated to performing cryptographic operations. Sophisticated or industrial standard cryptographic operations are always lacking in these tags. It is expected to have only 250-3000 logic gates within the microchip dedicated to security functions, ranging from low- to mid-cost RFID tags [78]. Asymmetric encryptions or public key infrastructure (PKI) are not possible for low-cost passive tags to perform at least in a few years time. Standard symmetric encryptions like DES, AES, or cryptographic hash functions are also hard to realise in these tags (standard AES requires 20000-30000 gates to implement [94] and standard SHA-1 requires about 15000 gates [4]). Recent researches have looked into this matter and tried to create low-cost RFID tag ready block ciphers and one-way hashing functions, notably the 8-bit architecture encryption only AES-128 implementation by Feldhofer et al. [26], which uses 3628 gates and the 8-bit architecture SHA-1 implementation by McLoone [66], which uses 5527 gates. With these implementation footprints, however,

Primitive	Logic gate	Clock cycle
PRNG [14]	1435	517
8-bit AES-128 [26]	3628	992
8-bit SHA-1 [66]	5527	344
32-bit SHA-1 [27]	8120	1274
32-bit SHA-256 [27]	10868	1128

Table 2.1: Security primitives for low-cost RFID tags

they can only fit into higher cost RFID tags currently. Table 2.1 gives a general idea on the gate counts and clock cycles needed to get one output from some security building blocks that can be used in RFID tags.

Notice that these limitations do not apply to the RFID reader. For example, ePassport use PKI to digitally sign the information stored on the tag. The tag here serves as a transponder only, which simply responses with the signed information. It is the reader that performs complex PKI operations to verify the digital signature from the information received. Active tags (tags that have their own power sources attached) on the other hand are more capable to perform these operations but are also more expensive. New PKI with elliptic curve can be realised with 12000-15000 gates for high-cost RFID tags. Expensive active tags that are capable in running PKI or WEP/WPA are less to be worried because these primitives can be used to construct secure wireless communication protocols, which have been studied for long. Because of the reasons above, the only possible security protections remaining in real world low-cost RFID tags are : tag access key, tag kill pin code (which can be found in EPC Gen-2 chips), random challenge generator and proprietary stream cipher (which can be found in the MIFARE chips). We will go into details when we review these RFID applications. Because there is still a big gap in the gate counts requirement for PKI than symmetric key primitives, we will simply separate RFID tags into high-cost and low-cost tags by those who can and cannot perform PKI. In other words, we are expending the range of low-cost RFID tags to include also those who are capable to perform symmetric key cryptographic and hashing. This is meaningful as we expect the cost of RFID tags to go down in the future and as technology advances, we expect symmetric key cryptographic and hashing to become standard features in low-cost RFID tags.

2.2 Advances in RFID Technology

When barcodes were first introduced, it did not cause people to concern so much about the security issues. Clearly, it is not what barcodes nor RFID is designed for that raised the alarm. What matters is how the RFID technology has advanced to cause people's concern. Theses considerable differences between barcodes and RFID become the origins of the security threats. Three main advances from barcodes to RFID can be identified.

2.2.1 Advance in Data Processing

It would not be so attractive to switch from barcodes to RFID if it does not increase the efficiency in reading items. Consider a trolley full of items after shopping in the market. It is very time consuming to scan the barcode of each item manually, one at a time. It would be nice if all the items are read at the moment the trolley passes through the cashier. RFID readers are capable to process multiple tags simultaneously at a time, which is much more efficient than barcodes where only one code is scanned each time. This advance in data processing method has effectively reduced the time needed in the labour intensive scanning task, human errors created and extra handling due to shrink or wear off of barcode printing. This is especially important in the supply chain environment where usually a container contains hundred to thousand of goods. But this also creates a problem. A reader read attempt will result in a failure if more than one tags response at the same time because of the jamming or overlapping of the RF signals. Some mechanisms are required to resolve this situation. Security threats targeting these failure moments and exploiting the security holes in these mechanisms exist. Extra effort is needed to make sure these fundamental parts will not become the tools for attackers to invalidate other security measures implemented for the system.

2.2.2 Advance in Data Communication

When a barcode is read, there must be a reader within line of sight. Hence most of the time users can be sure that when the barcode is being scanned and by whom. Or at least they can be rest assured that no one could have read the barcode if the printed code is being coved. However, the penetrating RF signals in RFID mean that readers can be anywhere within the signal range of RFID tags. Also, simply putting the tags inside a shopping bag does not prevent anyone from reading them. This advance in communication method effectively allows the tags to be read by unseen readers, which

creates a security threat because tags data can now be accessed without the users' consent. Extra security measures are needed as a consequence to make sure either: 1. Only known/legitimate readers can access the tags. 2. One who accesses the tags must be known. 3. Those who are unknown/unauthorised should learn nothing by studying the responses of the tags.

2.2.3 Advance in Data Storage

The only information about the attached object in barcodes is printed out obviously. It can be decoded and read by everyone, hence no one would have wanted to put down any sensible information in such format. While RFID usually equips with some memory storage spaces in the microchip, it is tempting to store more than just one unique ID there in order to provide more kind of services RFID systems can handle. This creates possible security threats when the stored information is something sensible or beneficial. It draws opportunists' interests to reveal the microchip content, which results in much more adversarial attacks than barcodes. Obviously, the stored information should not be sent in plaintext when responding to reader queries (most of the time the unique ID is needed to be protected as well) if there is also a certain level of privacy requirement to meet other than the security requirements. On the contrary, the extra memory can also be utilised to enhance the security of the RFID communications. Extra effort is needed to investigate on the proper use of these additional memory spaces to provide secure protocols to protect the stored information from being read in an unauthorised manner or leaked during the communication.

2.3 Security Threats Caused by the Advances

Based on the three identified areas of advance in RFID, we give an overview on most of the known security threats targeting these areas in this section. We aim to create public awareness for RFID users.

2.3.1 Threats Caused by Tag Collision

Although an RFID reader can batch process a group of tags and in the user's view it is only a single tag scan, the communication between the reader and the tags is in fact still one to one, just because the switching from tag to tag is so quick to be noticed. Example like the EPC Class 1 Gen 2 RFID system has enhanced the read rate from

230 reads per second to 880 reads per second in the US (different country has different regulation on the use of RF, which also affects the actual performance). However, no matter how quick the reader can process one tag response, there are still chances where problem arouses. Tag collision happens when two or multiple tags response together at the same time to the reader query. The tags have no problem on this, but the reader will have trouble to process all these responses and the collision will cause a read failure at the reader. An anti-collision mechanism is needed to solve this problem. The basic idea is to reduce the number of responding tags per reader query such that only one tag is *singulated* to respond to the reader at a time; hence the term singulation protocol is named for this mechanism. There are two major singulation protocols employed in RFID: tree-walking and ALOHA [11, 94]. Protocol ideas are presented to help understand the nature of the security threat.

Tree-walking

In tree-walking, when a collision occurs, the reader will execute the protocol by instructing only tags with their unique ID start with '0' (starting from the left most significant bit) to continue to response. Tags that match the request will response with the next bit of their unique ID. If there is another collision, it means there are tags with ID prefix '00' and '01' present. The reader will continue to send out another query for tags with their unique ID start with '00'. This process continues until a full tag ID can be determined. Then the reader will back track to the last collision point in the tree to send out another query for the tags with unique ID start with '$prefix||1$'. The whole singulation protocol continues until no more collision is detected, which is when all the collision points have been taken care of. With a list of presented tags completed, the reader can start one to one communication with each of these tags.

An attack to reveal the unique ID in tree-walking

With the tree-walking protocol, however, the unique ID of any tag can be revealed, even when it is stored in a secure format inside the tag. It is simply because the reader is giving out 1 bit of information of the unique ID of a tag per each collision. The attack is quite simple: consider an attacker who has any RFID device which is capable to simulate tag responses. Whenever a reader broadcast a tag query, the attacker will simulate a tag collision by sending out both 0 and 1 with its two antennas. In that case, the reader will keep executing the tree-walking protocol until the second last bit of any possible unique ID (if there are k bits altogether in a unique ID, the reader will

give out all the bits information from the 1st to the k-1th bit). A tag that continues until the very end becomes the victim tag because k-1 bits of its unique ID have been revealed by the reader. With such attack, even though the attacker does not possess the access key to read any tags or even if the communication between tags and reader is properly encrypted, the vital information of the unique ID can be eavesdropped by listening to the reader bit by bit.

ALOHA

In ALOHA, instead of giving out bit by bit instructions by the reader, when there is a collision, the reader will instruct all the tags to response again but they will have to first wait for a random idle time before sending out the response. Base on the different random choice of each tag, the collision has a better chance to be solved in the next tags to reader transmission. One may consider the tree-walking protocol as a deterministic approach while the ALOHA protocol as a probabilistic approach.

2.3.2 Threats Caused by RF Communication

Clandestine Skimming

Skimming refers to unauthorised scanning of RFID tags without the tag holder's consent. Obtained information can be used to clone tags or track any individuals. The nature of passive RFID tags has provided a convenient way for attackers to study the tags responses. These tags basically reply to every reader query. This is unavoidable because neither the tags nor the reader are aware of the others presence before any communication happens. Also because passive tags are powered by the RF signals from the reader, it has to be the reader that starts the communication first. As a result, attackers can always obtain any tag responses at will. To allow the reader to identify and authenticate a tag, the tag responses must have included its unique information, be it in plaintext or even encrypted, for the reader to locate its record in the back-end database; otherwise, even the legitimate reader could have no way to distinguish them. Hence more or less some information about the responding tag is available to any attacker with a rogue reader. The challenge becomes whether the attackers can extract useful information from these tag responses or not. Some physical means also exist to protect the tags from being scanned clandestinely by blocking the RF signals.

Eavesdropping with hidden rogue readers

Proper communication protocol design may prevent RFID tags from leaking unnecessary information to active rogue reader scanning. Tags can stop the communication when the rogue reader cannot provide proper responses in correct format. But this does not apply to passive eavesdroppers. While physical protection can only stop clandestine scans when the RFID tags are not in use, an attacker eavesdropping on the communication between tag and legitimate reader has the same effect as skimming. Legitimate reader always completes the protocol with legitimate tags. Hence these are chances for eavesdroppers to learn extra tag information where they could not have been able to do it without the help of a legitimate reader. The high power RF signals originate from the reader to tags are unavoidably received by attackers. Depends on the distance between the tag and the rogue reader, tag responses can be sniffed as well. Some may argue that this is not practical due to the much weaker output of the passive tags. But since there is such possibility, communication protections at both reader and tag ends are required.

2.3.3 Threats Caused by Storing Sensible Data

Tag Cloning

Tag cloning is especially easy for those RFID tags that only output a static value as their unique ID. The required information is obtained with a scan to the tag. Attackers simply record one of the responses sent from the victim tag to a legitimate reader. The unique ID (so as other information) obtained from the victim tag can be copied to another empty tag, which effectively creates a legitimate clone. The clone tag simply replays the same unique ID, which allows it to pass tag identification and leads to impersonation attack in some applications. Access control can help lowering the chance an attacker to obtain the tag information. Random challenge response protocols can further mitigate the success rate of tag cloning as replaying the same response will most likely to fail in front of these protocols.

Tag Counterfeiting

For tags that generate different responses per query, counterfeiting takes a further step to emulate the responses of a legitimate tag. Even with random challenge response protocols, full memory dumping and reverse engineering to these tags are also possible. This requires the learning of the stored key and simulation of the cryptographic

algorithms if the RFID system has tag authentication protection. An attacker is assumed to be capable to perform such delicate task. Adding tamper resistance may at first glance seems to be promising but it comes with a higher production cost, which is not suitable for low-cost RFID tags; besides, there is no guarantee on their resistance ability [1]. Most of the time the emulation (counterfeiting) is done on a more complex device like a pda rather than to recreate a similar RFID tag. Hence an honest and trustful operator can help spotting such attack attempt. Other than that, the remaining solution to these threats is to mitigate the motivations to attack a tag.

2.4 Example RFID Applications in Real World

2.4.1 ePassport

International Civil Aviation Organisation (ICAO) defined standard for ePassport. ePassport stores what are printed in the personal details page as well as the digitised photo of the holder. These information are digitally signed to guarantee data integrity but they are not encrypted. There are also access control and tag authentication in ePassport. The access key is the combination of personal details printed in machine readable format inside the passport. The cover have metallic materials inside to block RF signals when the passport is closed.

Researchers have published their assessments to the security and privacy issues in ePassport [45]. Like other RFID applications it is subject to tag tracking, which violates the location privacy of the holder. The non-encrypted personal details and photo can be eavesdropped easily, which violates personal privacy. Clandestine skimming is also possible because most of the information used to derive the access key are easily available (passport number, DOB) and guessable (expiry date). In other words, the access key is not random enough. It is expected to brute force the remaining unknown bits within a few hours.

2.4.2 US Passport Card and Driver Licence

These RFID embedded cards provide much less security protections than ePassport. They are essentially wireless barcodes as described in a security assessment report [57]. The only protection is the protective sleeve that comes with the card. These cards use EPC Gen-2 chips, which do not have tag authentication protection. The access control feature is not turned on either in these applications. The unique ID is sent in

clear text, meaning that tag cloning is possible and was demonstrated in the report. Since there is no tag authentication, tag counterfeiting can be done even using only off-the-shelf EPC tags. Emulation using complex device does not require an exact copy to be made. If the border officer has not verify the facial details with the card holder, then any fake copy can get passes.

The US Homeland Security also provide their assessment with countermeasures for these cards, which we summarise below:

- The biographic data obtained from the back-end database using the card unique ID will be verified by the officer with the card holder to detect cloned cards.
- Individuals should always put their cards in the supplied protective sleeves to avoid skimming and tracking.
- Readers are placed within sufficiently large physically protected area such that illegitimate readers outside this perimeter cannot eavesdrop on the tag and reader communication.
- Requires where possible that only the unique ID is transmitted during communication to minimise the impact to personal privacy.
- Individuals are educated during card enrollments about the permissible use of RFID cards.

2.4.3 UK Transportation Card

The transportation card in the UK is called Oyster card, which is used as a mean to replace tickets for public transport in London. Smartrider in Australia also use this chip. These cards use MIFARE classic chip, which implements a stream cipher and use a 48-bit access key (ePassport use 56-bit) for tag authentication purpose. The manufacturer had not published the details of the design of the stream cipher until recently, attacks to the stream cipher was found by researchers [33]. It turns out that through cryptanalysis, the messages exchanged during the authentication protocol contains enough information to recover the symmetric key in less than a second. This is an example that the symmetric key can in fact be revealed through tag and reader communication, although in our research, we assume in our adversary model that an adversary can always obtain the symmetric key by dumping the tag memory content. The authentication algorithm and the implementation of the stream cipher is now fully

understand and full functioning code to counterfeit the card is available online. Two important lessons can be learnt from this experience:

- Security through obfuscation (by hiding the details of the cryptographic algorithms) does not last long. Only through public reviews, tests and formal proves can a good algorithm be created. As suggested by the RFID CUSP group, open design is one of the three principles in security design.

- The security of RFID systems cannot rely solely on the tag or reader or the underlying cryptographic algorithm. In the case of the Oyster cards, although their cards can now be cloned easily, the impact is not that much. Every transaction of ticket purchases are logged in a database and processed everyday. Duplicate IDs can be found and the cloned cards are marked invalid for further use. This means that a cloned card can have at most one day life. In the case of the US passport card, cloning is extremely easy but an honest officer can still catch the cloned card since the biometric information will not match with the card holder. These are examples that a secure RFID application should be backed by more than one security components.

2.5 Industrial Countermeasures

Some industrial practices for each of the security threats we have mentioned will be summarised in this section. For the example RFID products, we will refer to the EPC RFID system and the ePassport in the US. It is better to have these as the examples to see what the industrial practices are and how the industry deals with the various security threats. Although from a research point of view, RFID systems are far from secure, it is still worth to sum up the practices being adopted by the industry provided that RFID products are gaining more and more interest commercial-wise.

2.5.1 Countermeasure to Tag Collision Attack

The bottom line here is to try to avoid leaking any information in any situation, no matter how small the portion is. Hence it is best to avoid using tree-walking as the singulation protocol if the RFID product provides options to choose from. Comparatively, although the ALOHA protocol is probabilistic, it leaks no information compare to the tree-walking protocol. This becomes a good practice adopted by the industry. EPC class 1 Gen 2 has advanced to use the ALOHA protocol instead of the tree-walking

protocol, which was used in Gen 1. The advice here is that users should be aware of the anti-collision or singulation protocol that is being implemented in their RFID products as well. Higher level security measures like encryption or access control could be invalidated at a lower level if an insecure anti-collision protocol is used.

2.5.2 Countermeasure to Unauthorised Tag Scan

Limited-range transmissions

To combat eavesdropping, reader should not send out excessive power signals or it will just extend the eavesdropping range. By lowering the power output of the reader, this range can be reduced and it can exclude those out-of-range hidden rogue readers. This results in a closer proximity for the attackers to launch their attacks and forces these eavesdroppers to get closer to the subject. It becomes easier to spot any hidden readers. By doing so, those who try to read the tags will most likely expose themselves.

Encryption

Another important practice is to encrypt all the communication between tags and reader. Low-cost RFID tags may not be capable to perform encryptions on the fly. But any stored content including the unique ID can be pre-encrypted and leave the reader to do the decryption. This is always a best practice to prohibit active scanners from harvesting useful information from tags responses by querying the tags. With a proper encryption, those who are unauthorised will learn nothing from tags responses.

Access control

One of the main security measures of both EPC tag and ePassport is the implementation of basic access control (BAC). Keys are used in BAC to provide verification and authentication, sometimes they even provide encryption for a secure communication protocol. With access control, only those who are authorised can read the information stored inside the tags.

Take EPC as an example. Class 1 Gen 2 tags have a 32bits access key stored (it was only 8bits in Gen 1) inside their reserved memory bank. When a tag has been singulated, the reader holds a random 16bits handle generated by the tag during the singulation process. The handle is used as a reference to this specific tag. If access control is turned on in this tag, the handle also acts as a randomised key to mask (XOR) the access key. Upon verifying the access key provided by the reader after

two message exchanges (two 16-bit random handles masking for each half of the 32-bit access key), the tag is unlocked for access to the information stored. The same process can be used again to relock the tag or lock again a locked tag to turn it into a read only tag forever. Notice that the 16-bit random handles are generated by the tag and send in plaintext in the tag to reader communication channel. Clearly, using them to protect the access key requires an assumption that it is hard for the attackers to sniff on the tag to reader communication channel. Because it is easy to sniff on the reader to tag channel, which is why the access key is needed to be XORed by these "half"-secret random handles.

ePassport in the US also has BAC. First there is a symmetric en/decryption key already printed on the passport pages. It combines of the passport number, DOB, passport expiry date and three check digits. ePassport RFID tags are capable to perform triple-DES encryption. First the tag generates a random number R_T for the reader. Reader will use the encryption key to encrypt this number together with two of its own generated random numbers R_R and K_R for the tag. Upon receiving the encrypted value, the tag will decrypt it and check if it contains the random number R_T sent to the reader earlier. Next the tag generates another random number K_T and sends to the reader the encryption of R_R, R_T and K_T. After the reader has verified R_R is contained in the encrypted value, both the reader and the tag have established a session access key using $K_R \oplus K_T$ as the common reference string to generate the actual key. Again, security is based on the hardness for the attackers to obtain the en/decryption key printed on the passport. Or we can say that once the attackers have obtained all these information, there are nothing left that store in the ePassport will interest them anymore. In fact, the only biometric information stored in the ePassport that seems to have some value is the photo of the passport holder. Some countries also store fingerprints, which we would not recommend provided that the security is built base on such a weak defense.

Faraday cages

A physical measure to solve these attacks is to simply block all the RF signals from reaching the tags. RF shielding is one of the security countermeasures of the e-passport in the US. Metallic mesh is wrapped inside the passport cover to prevent the passport from being read when the passport is closed. Retail products are also available from third parties in the form of shielded wallets which provides equivalent protection should RFID VISA cards and RFID banknotes become more popular. In some sense, this may

be the most practical security protection method but the reason to use RFID has lost.

Signal blocking

A more adaptive physical measure is the use of a signal blocking device, a blocker tag [47]. It is a special RFID tag that simulates the presence of every possible RFID tag within the signal range of the reader in order to obstruct the identification of any particular tags. It reverses the purpose of singulation protocols or anti-collision protocols which allow a reader to determine which tags are present when multiple tags are within range of the reader simultaneously. It works by sending out two responses at the same time to trigger a tag collision whenever it receives a reader query. However, like the ID revealing attack we discussed before, a blocker tag is exactly the tool attackers need to launch such attack. Abusing the use of blocker tag also creates troubles for other applications running nearby or even turns into a denial-of-service attack.

2.5.3 Countermeasure to Tag Counterfeiting

No sensible data to be stored in RFID tags

Separating all the information that will attract attacks from the tags to more secure back-end database server is a good practice. Use only the unique ID extracted to bring up other related records and information. By doing so, only the ID of the tag is compromised at the worst case. There can be other means like revocation to stop attackers from using the compromised ID to do other misbehaviour.

Tag killing

Passive RFID tags are believed to have infinite life. They keep on functioning even when they are thrown away. Users should be aware of this and a proper way to depose them is required. Instead of burning them in a microwave, EPC tags have this kill command that can deactivate the tags at time of their final use. Killing the tag can eliminate the possibility that information inside the tag is leaked to anyone. The killing protocol is similar to the one with access key. Both the kill password and the access key are 32-bits values. However, this is not a reversible process, hence it should be done with caution.

2.6 Conclusion

RFID is not only an advance version of barcodes, it has many other potential that cannot be found in barcodes. Yet such an advance in technology also brings in many new forms of security threats. In this chapter, we have discussed some of the best practices when using this new technology. We have also looked at some of the practical industrial security countermeasures to these threats. We would like to add in here that user education is especially important in addition to all the technical means. As our research proved in the later chapters that low-cost RFID tags are impossible to avoid tag tracking attacks, two of the measurements: shielding the tag at all other idle times and user education, are important to mitigate the impact of tag tracking. The first one largely reduces the possible time of attack to the moments when tags are removed from the RF shield. The second one tells the public what are the possibilities and impossibilities of RFID tags. This is especially important in order to raise the public awareness to the potential security and privacy issues coming with this new technology. Best practices are of no use if users are not aware of the threats surrounding RFID. There will not be a single answer on how to secure RFID. It is always situation and application dependent. Hence knowing what RFID can and what RFID cannot should be the base of all security measures.

Chapter 3

RFID Protocols

Chapter Overview

In the previous chapter, we have looked into the RFID basics. We compared RFID with barcode, explained the security threats when deploying RFID, listed some of the possible attacks and discussed some of the industrial countermeasures. This gives us a general understanding of the challenging environment that RFID is facing. In this chapter, we will give an introduction on RFID protocols. First we discuss the system requirements, which is an important prerequisite for a secure RFID protocol. Then we will look at some common security goals that RFID protocols can provide. The settings are specially catered for low-cost RFID tags, which have the claimed security constraints as discussed in the previous chapter.

3.1 Functionalities of RFID Protocols

So far we have discussed the security threats to the whole RFID system, together with some physical means to protect the system, like shielding the tag, jamming the RF signals, limiting the signal strength/range and killing the tag. Although all these physical countermeasures can actually thwart the threats to RFID systems, it is not difficult to see their nature are devastating, in a sense that they are sacrificing the communication capability among tags and reader. Of course they are still good practices for real world RFID applications because most of the low-cost RFID systems still do not have adequate security protections on their RFID protocols. To restore the full functionalities of RFID system, security measures should be moved back to the protocol level.

We summarise the most common characteristics of low-cost RFID tag systems as follow:

- Secure back-end database server
- Secure and reliable reader to back-end communication
- Trustable reader (operator)
- Tag and reader are capable to generate random numbers
- Tag is at most capable to perform symmetric key cryptography and secure hashing
- Tag is unable to perform PKI cryptography
- Tag is not tamper resistant

With the above settings, low-cost RFID tag systems define a communication protocol for tags and reader, which can provide the following three main functionalities:

- Tag identification
- Access control
- Tag authentication

3.1.1 Tag Identification

Most of the RFID chips are supplied with a unique identification number called tag ID at manufacturing time, which gives the uniqueness to every RFID tag. Apart from the tag ID, custom ID can also be stored in the tag memory. These IDs are entered into the back-end database before the tags are being deployed. When a tag is being queried by a reader, it will response with its unique ID (in clear text, encrypted form or some pseudonym replacements). If the tag has been registered in the system, readers that have access to the back-end database can use the information obtained to identify the tag and locate its record in the database. This is the most basic functionality of RFID protocol.

3.1.2 Access Control

RFID systems that provide tag identification only will usually have their tags response to every reader queries (even if the reader is malicious). Access control provides better protection by having the tags respond to readers that hold a correct access key only. The access key unlocks the tag to release further information stored in its memory

[94]. Some systems allow these information and the access key to be rewritten. The access key can also be used to lock a tag to become a read only tag, this feature can be found in EPC Gen-2 tags. Access keys can also be unique to every individual tag, like the ePassport example where it is derived from optically scannable data printed on the passport pages. Random number generator is usually required on the tag side to provide a challenge to the reader and to mask the access key. Because of the first three RFID system characteristics, it is assumed that an adversary cannot obtain the access key via legitimate readers nor from the back-end database. However, since the access key is involved in the RF communication, weak protocol design may reveal the access key to the adversary, which is the case of the MIFARE cards attacks [33].

3.1.3 Tag Authentication

Access control allows the tag to verify the access key before further information is released, which effectively authenticate the reader. Tag authentication, on the other hand, offers authentication for the tag. Compare to tag identification, the unique ID supplied in a tag response tells the reader the claimed identity of the tag but does not prove its validity (e.g. think about a message replay). In authentication, a cryptographic protocol using shared symmetric key is usually required for the tag to prove that it knows the access key and is able to response with a correct value under random challenge supplied by the reader. This can be done in two message passes: reader first sends a random challenge and the tag replies with a response related to that challenge. Reader authentication is done implicitly: if the reader is not legitimate, it cannot obtain useful information about the tag identity; if it can, then it is legitimate. For some applications, the tag requires additional actions like key updating only when it has communicated with a legitimate reader. In that case, reader authentication can be done explicitly by using mutual authentication protocol. This is usually done in three message passes: reader sends a challenge, then the tag replies to that challenge together with its own challenge for the reader, the reader produces a response related to the tag challenge so that the tag can verify and subsequently updates its key.

Authentication protocol is not commonly seen in daily RFID products (e.g. the passport cards and driver licences in the US) but is seen in more security critical applications (e.g. ePassport and the transportation card in the UK). This protection guarantees that simple message replay by an adversary is not possible thanks to the random challenge and the symmetric key cryptosystem. However, since the low-cost RFID tags are not tamper-proof, there is no guarantee on the secrecy of the symmetric

key and even the random number generation and the authentication algorithms can be revealed if the adversary is able to dismantle the tag to study the circuit design and dump the memory content.

3.2 Prerequisites for Secure RFID Protocols

In an RFID protocol, the communication can be separated into two parts: at the front-end where tag and reader communicates and at the back-end where reader and back-end server communicates. A secure RFID application must be backed by a secure back-end server. It is the first defense of the whole system. We do not consider applications where the back-end server is insecure. We require the back-end server to be non-compromisable. Also, the communication channel between the reader and the back-end server has to be secured by some secure network protocol before we can focus on the security at the front-end. It is the responsibility of the system providers to maintain a reliable and secure back-end or else all the security measures implemented at the front-end will be rendered useless.

The second defense is the reader, which also means the operators of the service if humans are involved. Legitimate readers are assumed to be used by honest operators. Other system users may sometimes be malicious but never the operators. Besides honest, they should also be trustful to perform security measures. For example, the custom and border officers in the ePassport example should follow the protocol to verify the facial details of passport holders faithfully even though the digital signature verification passes. In most RFID applications, legitimate readers always have a secure online connection to the back-end server in order to perform any immediate information lookups and any necessary information updates. Hence, most of the time the back-end server and each of the legitimate readers are considered as a single entity to simplify the technical context. Because of this, compromising any legitimate reader means also gaining access to the secure back-end server, which is not allowed (i.e. not protected) in most of these RFID applications. Some [9, 34] consider compromisable legitimate readers in off-line RFID systems, i.e. these legitimate readers do not always have a continuous connection with the back-end server in order to see how they perform in terms of privacy protections. The results are either quite negative [34] or require a non-compromisable time period [9] in order to preserve privacy. Comparatively, these systems are less common to be considered.

For the above prerequisites, we have the following suggestions:

1. To guarantee the first defence, the back-end database server is better to stay private and not to be publicly accessible.

2. Following point 1., those who can access the back-end database server must be legitimate. A security mechanism is suggested to authorise and authenticate all the connecting readers (or operators if human is involved) such that no unauthorised access is allowed.

3. Following point 2., there must be physical protection to protect curious or even malicious outsiders from accessing the legitimate readers easily. Depending on the applications, one may combine the back-end database and the reader into a single device to be protected together physically if the risk for losing the reader is low. This can satisfy point 1 and 2 at the same time if done correctly.

4. Following point 3., it is suggested to have another system to monitor the use of legitimate readers, by whom, when and where. This also helps guarantee the honesty of the operators.

5. Following point 4., it is suggested to keep log of all the access to the back-end database server (e.g. access time, reader ID, queried information, etc.) such that any adversarial activities can be traced should point 3. or 4. fails.

6. In case there is a legitimate reader being stolen, there must be some contingent plans to stop the stolen reader from accessing the back-end database server again. Best if the plans do not affect the user experience of other current legitimate readers.

3.3 Adversarial Attacks on RFID Protocols

From the point of view of a user, the security concern is more than just the authentication of the tag or reader but also his/her privacy of location (i.e. tag tracking). Tracking is a unique security threat for RFID systems that targets personal privacy. Through eavesdropping, an adversary can at least obtain some information about a tag. Although these information can be encrypted or look random (if not in plaintext), they are still unique per tag, otherwise not even the legitimate reader can identify (and authenticate) this tag. These information serve as a personal number plate that aids the adversary to track the movement and location of the tag bearer. Some secure RFID authentication protocols try to randomise the tag key after each tag scan, but

then again, the changes made to the tag outputs due to the randomised key must still be identifiable by the legitimate reader, which shares the same symmetric key with the tag. How well the low-cost RFID tag systems can perform under this limitation in protecting tags from tracking is the main research topic of this thesis. We will discuss more about the privacy issues in section 3.4.

Let us first look at some common attacks that adversaries use on RFID protocols to achieve their adversarial goals. Remember the three main functionalities of RFID protocols: *tag identification*, *access control* and *tag authentication*. Tag identification says nothing about whether the claimed ID of the tag is true or not. It suffers from easy tag cloning. Access control authenticates the reader to the tag, but still the identity of the tag is not verified. In a secure application, this is not enough. For now on, we will only focus on tag authentication.

Adversaries who attack RFID authentication protocols are commonly assumed to have the following abilities:

- Record communication content between legitimate reader and tags
- Communicate to any tags and reader
- Control the communication medium
- Dismantle tag to learn the cryptographic algorithms and obtain any stored data

With these abilities, adversaries can launch various attacks to RFID protocols. To give ourselves a clear picture, we assume there are three parties: the RFID reader, an RFID tag and an adversary. The goal for the reader is to identify the tag and requires the tag to authenticate itself. The tag on the other hand replies to every reader queries, authenticates itself to the reader without leaking any useful information that will consequently violates any security concerns during the authentication protocol. And the adversary tries to stop both the reader and the tag to achieve their goals by a combination of the abilities listed above.

3.3.1 Desynchronisation Attack

To authenticate a tag, there must be some secret information that is "known" by the tag only but not the others. Using this secret information, the tag can compute a unique response and prove to the reader it is who it is. This is common as in other wireless systems when computational power is not an issue. For low-cost RFID tags,

a standard cryptographic calculation may become a burden. Hence it is proposed in [43] a minimalist for low-cost RFID systems. At the system setup time, a list of pseudonyms is generated for every tag. These values are also stored at the back-end server, associating with the tag ID. The tag can reply with the next pseudonym for each reader query and the reader will match it with the database to find the true tag ID.

Clearly, this protocol can authenticate the legitimate tags and differentiate the fake ones because of the extremely small probability that an adversary can guess the next pseudonym correctly. However, there is a problem when a tag runs out of pseudonyms. Normally it may take a reasonable time before this happens, but an attacker can launch a desynchronisation attack by querying the tag repeatedly. Since the tag will always reply with the next pseudonyms, such attack will cause the tag out-of-sync with the reader (i.e. the next pseudonym assumed by the reader is different from the tag). Eventually, the tag will exhaust all its pseudonyms and the reader cannot authenticate this tag anymore. Hence a desynchronisation attack can lead to a denial-of-service (DoS) attack. This DoS attack can result in simple tag tracking: desynchronise a victim tag and if DoS happens later on, the victim tag is there. More desynchronisation attack techniques are discussed in [49].

3.3.2 Man-In-The-Middle Attack

To overcome the limited pseudonyms exhaustive problem, it is best for the tag to be able to refresh its pseudonyms or generate a new one for each reader query. Because of this, [37] proposed an approach called "universal re-encryption". In this approach, the pseudonym stored inside a tag can be refreshed by the reader or any external trusted devices. One application to this protocol is the RFID-enabled banknotes system [46, 96, 98]. However, the re-encryption process has to be done privately and by a trusted party because the new value is sent directly to the tag to replace the old one, it suffers from easy denial-of-service attack, simply replaces the pseudonym with some invalid value. [4] discussed other security issues in this approach. [85, 86] on the other hand proposed a different approach where the tag and the reader keep track on a timestamp. The reader will send the current timestamp t_c to a tag and the tag will compare it with its own timestamp t. If $t < t_c$, the tag will reply with a new pseudonym computed using t_c and update its timestamp t to t_c; or else the tag will reply with a random value. The reader can compute on the same pseudonym using t_c to authenticate the tag. However, both of the proposed protocols are later found to be

vulnerable to man-in-the-middle attack in [49].

A man-in-the-middle attacker puts herself between a reader and a tag, connecting them as a middle man where normally the signal range of the reader cannot reach the tag. This means that all the communication between the reader and the tag has to go through the attacker. During a protocol session, the attacker may adaptively replace/swap/delay the communication content. This may require some cryptographic knowledge. Like the attack on [37] requires the attacker to be able to compute a re-encryption of the pseudonym. While the attack on [85, 86] is more elementary. It is also addressed in the same paper. First the attacker replace t_c to a large value t_m (a future time). Since $t_m > t$, the tag will update its timestamp to t_m. After that, this "future" tag will not be authenticated by the reader because $t_m > t_c$ and the tag will always reply with a random value until t_c catches up with t_m. The man-in-the-middle attacker has effectively desynchronised this tag and causing a DoS to the tag temporary, which as a result also leads to tag tracking.

Man-in-the-middle attack can also be used to reveal the shared symmetric secret key. An example is the HB protocol and also its family. HB protocol is first proposed by Hopper and Blum [42], hence the name HB protocol. The protocol was targeted at human to computer authentication, but Juels and Weis later on found the similarities between RFID tags and human, where both have very limited computational power, and based on the HB protocol, they proposed an RFID specific protocol called HB^+ in [48]. Following their work, a lot of other variants were proposed. To prevent running out of pseudonyms, HB protocols use challenge-response to allow adaptive responses based on the challenges given and the shared symmetric secret key, which can be reused polynomially without worrying the exhaustive attack above. The computations required are very easy. The original HB protocol requires only the tag to perform an inner dot product between a k-bit random challenge and the k-bit secret key. The response is an one bit message. Of course, any attacker can guess the result right half the time. So HB protocols require the challenge-response round to repeat r times and hence the probability of guessing all the results right (blind guess only) becomes $\frac{1}{2^r}$.

However, a passive eavesdropper who captures enough valid messages can compute the secret key by solving the linear equations. To overcome this, a random noise is added to each response by the tag through out all r rounds. The idea is to alter the response η of the time so that the reader will authenticate the tag if less than ηr responses are invalid. The security of HB protocols is based on an NP-hard problem called *learning parity in the presence of noise*. Juels and Weis noticed that the original

Tag$\{x_{ID}, y_{ID}\}$		Reader$\{x_{ID}, y_{ID}\}$
Challenge-response round:		
Choose $\nu \in \{0,1\}$ randomly where $P[\nu = 1] = \eta$		
Pick $b \in \{0,1\}^k$ randomly	$\xrightarrow{b}$	
	$\xleftarrow{a}$	Pick $a \in \{0,1\}^k$ randomly
Compute $(a \cdot x_{ID}) \oplus (b \cdot y_{ID}) \oplus \nu = z$	$\xrightarrow{z}$	Verify if $z = (a \cdot x_{ID}) \oplus (b \cdot y_{ID})$

Figure 3.1: HB$^+$ Protocol

HB protocol is not secure against an active attacker. The attacker can repeatedly query the same tag with the same challenge. With high probability, the correct response can be obtained if enough samplings are recorded because the noise only happens η of the time. This is undesirable in RFID environment. They remedied this by an improved version of the HB protocol called HB$^+$, where an additional blinding value is added. Figure 3.1 illustrates this protocol.

Here the random noise ν is chosen to be 1 with probability η. The tag is authenticated if the accumulated number of incorrect z after r rounds is less than ηr. The improvement made by Juels and Weis is the blinding factor b. If an active attacker is to query a tag with the same challenge a, the response is still XORed by $b \cdot y_{ID}$, hence the effect of the noise is further blinded. However, Gilbert et al. showed a man-in-the-middle attack in [36] for the HB$^+$ protocol and subsequently for the rest of the HB protocol family in [35]. They call their attack the GRS attack. The attack exploited an additional information coming from the reader: the authentication result. A reader will not send out an authentication result explicitly but instead it simply grants access or not to the tag based on the result. By seeing an opened door or a successful transaction for example, this becomes a side-channel information for the attacker. Take the HB$^+$ protocol as example, the attacker launches her man-in-the-middle attack by altering the challenge a by 1 bit. If after the alternation the tag still passes the authentication, then the attacker knows the corresponding bit in x_{ID} based on the altered bit position is 0, as changing the bit in a does not affect the result. Otherwise, the corresponding bit in x_{ID} is 1. The attacker repeats the attack for all other bit positions and finally the whole x_{ID} can be revealed. This shows a powerful use of man-in-the-middle attack to reveal the secret key in protocol level without physically tampering the tag.

3.3.3 Relay Attack

RFID is especially vulnerable to relay attack. This is a powerful attack and requires no cryptographic knowledge to launch such attack. Like the man-in-the-middle attack, an attacker again stands between a reader and a tag but this time she does not need to change any communication contents. The attacker who carries out this attack will act as a tag to be authenticated and records the challenge sent from the reader. Since the attacker lacks the secret information that is required to compute a correct response to the challenge, she will need to find a legitimate tag as the attack target. Later on when an attack target appears, this challenge is relayed to the target by the attacker and the valid response from the target is forwarded back to the reader. As a result, the attacker can deceive the reader from believing that it reads a legitimate tag but it was the attacker actually.

Timing is an important measure to defend such attack. Notice that in the above scenario, the attacker has stored the challenge for a period of time before relaying it to the attack target. A simple method is to assign an expiry time for each challenge and response in the protocol (i.e. replying with a response computed from a challenge that has expired will not pass the authentication). However, [53] has demonstrated a nearly real time relay attack. In its settings, the attacker requires two devices namely the ghost (which communicates with the reader) and the leech (which communicates with the tag). Consider a scenario that the attack target and the attacker is queueing at the cashier and use RFID credit card to purchase items. The attacker slips the leech close to the target credit card. Through a wireless link, the leech is connected with the ghost on a much higher bandwidth and faster link than the RF link between the reader and the credit card. The ghost is presented to the reader to collect the challenge and send to the leech in real time. The leech then relays the challenge to the credit card and the response is forwarded back to the ghost through the fast wireless link. Finally, the target credit card will be charged instead. It is estimated in [53] that the longest distance between the ghost and the reader can be 50m apart. The leech and the RFID credit card, although the card has a much less power output, can also be separated about 50cm away from each other. It turns out that the close proximity assumption (that the data acquired by the reader must be come from the card presented) does not hold anymore. It is suggested that additional counter measures are needed. For example, add a switch to turn on or off the RFID card, or an additional PIN code is required every time a transaction is started.

Tag$\{K_{ID}\}$		Reader$\{K_{ID}\}$
Slow phase:		
	$\xleftarrow{N_V}$	Generate random nonce N_V
Compute $\mathcal{H}_{K_{ID}}(N_V) = R$ where $R = R_1^0...R_n^0 \| R_1^1...R_n^1$		Compute $\mathcal{H}_{K_{ID}}(N_V) = \hat{R}$ Generate random bits $C_1, ..., C_n$
Fast phase:		
	$\xleftarrow{C_i}$	for $i = 1...n$, sends C_i
	$\xrightarrow{R_i^{C_i}}$	Verify if $R_i^{C_i} = \hat{R}_i^{C_i}$

Figure 3.2: Distance Bounding Protocol

Another approach is to calculate the round-trip time of the messages being exchanged between the reader and the tag. This is called proximity check and is used in distant bounding protocols. Distant bounding protocols are not designed only for RFID originally. The first distant bounding protocol was proposed in [13] by Brands and Chaum. An RFID specific one was proposed in [40] by Hancke and Kuhn. Figure 3.2 illustrates this protocol.

Here K_{ID} is the shared symmetric key of the tag ID with the reader. $\mathcal{H}$ is a secure hash function that hashes the nonce into a $2n$-bits value R using the tag key during the slow phase. R is then split into two n-bits portions R^0 and R^1. During the fast phase, if the tag receives a random bit C_i from the reader, it immediately responses with the i-th bit R_i^0 if $C_i = 0$ or R_i^1 if $C_i = 1$. After n iterations the reader will receive n random bits from the tag. If all n bits match the corresponding bits from the pre-computed value $\hat{R}$, then the reader can guarantee that the tag is within a distance $d = c \cdot \frac{t_m - t_d}{2}$ where c is the propagation speed of radio wave (or speed of light), t_m is the round-trip time and t_d is the computation delay on the tag.

Distant bounding protocols are specifically designed to thwart relay attacks. The idea behind is to use the round-trip time measured and the speed of light to calculate the maximum possible distance between the reader and the tag. Hence any delay in receiving a message will result in a conclusion that the message is possibly being relayed or the tag is not within proximity. The protocol consists of two phases: the slow phase and the fast phase. The slow phase is for the tag to prepare computation extensive calculations and the fast phase requires the tag and the reader to rapidly exchange bitwise results of the computed value obtained in the slow phase. The bounded distance will cause relay attacks infeasible and force man-in-the-middle attacks to expose within close proximity. This area of research is another big topic in RFID, because it does

not require complex computation to achieve its goal, it especially suits the constrained environment of RFID. But because its security is based on some physical assumptions, it is out of scope for our research. We will only give a brief introduction here.

3.3.4 Replay Attack

Replay attack is quite similar to relay attack, but this time, the attacker does not need to carry out the attack in real time. During an authentication session, the reply from an attack target is eavesdropped by the attacker and stored for later use. Next time when there is a reader query, the attacker can response with the same reply and the reader will authenticate the attacker as the attacked target. [94] is an example protocol that stands against the desynchronisation attack and man-in-the-middle attack but is vulnerable to replay attack as shown in [49]. A naive approach to fix this is to maintain a list at the back-end server of the replies provided by each tag. [14] suggested that it is more appropriate to defend this attack by requiring the tag responses be unique and unpredictable for every reader query. Usually this requires the use of cryptographic hash functions. An example protocol that is resistant to replay attack, desynchronisation attack and man-in-the-middle attack is proposed in [17].

3.3.5 Timing Attack

The protocol proposed in [17] is quite satisfactory as an authentication protocol for RFID, it stands against most of the attacks discussed above (excepts the powerful relay attack). There is more to consider when privacy is of concern. It is pointed out in [49] that this protocol is vulnerable to timing attack. An attacker will try to gain advantage using the timing information available during the protocol (e.g. the different elapse times taken for the reader to output pass or fail for the tag authentication) to identify different tags. First, the attacker keeps querying the attack target, in order to desynchronise it. The protocol in [17] is not vulnerable to desynchronisation attack as a legitimate tag can always re-synchronise with the reader. Although the attacker will not be succeed in making the tag denial-of-service, she has a different goal this time. All she does is to “mark” her attack target and releases it back to the system. This action does cause some changes to the authentication process. The reader, although it can authenticate the attack target eventually, it does take a longer than usual processing time (compares to other non-attacked tags) to finish the authentication. If the attacker can distinguish this timing difference, she can identify her attack target again at a later

time after the "marking" action. This is a violation to the tag bearer's privacy. This attack is considered as a side-channel attack. There is no solution to this yet but we suggest the authentication process to take a constant minimum processing time that is long enough to cover the longest processing time needed to authenticate a tag (possibly been attacked). Hence it is not only the communication content that matters when designing a new authentication protocol, how the algorithm is implemented such that no attacker can gain advantage is also important.

3.4 Privacy in RFID Authentication Protocols

Because the communication between the reader and tags is over an insecure channel, one has to assume that all the communication contents are available to an adversary. Requiring a reader to authenticate itself to a tag does not help much as an adversary can always eavesdrop on the communication without actively involving in it. It is also possible for the tag to break the privacy of the tag bearer during the authentication protocol if the protocol is poorly designed. Hence both authentication and privacy have to be considered equally when attempting to design a secure RFID authentication protocol. In this section we will look at those privacy concerns that have been considered in the design of RFID authentication protocols to protect the tag bearer's privacy.

3.4.1 Personal Privacy

The most basic privacy concern that can be seen in nearly every RFID authentication protocol is *personal privacy*. It was first seen in [94]. Though it was not formally defined in the paper, the author related this to *"keeping the RFID tagged content private"*. Later on in [74], this is referred to *data leakage* or *data privacy* as in [41]. In that sense, one can understand this as a requirement that the content stored inside an RFID tag (so as the tag ID) is kept secret to everyone and should not be revealed during the authentication process. Only a legitimate reader within the same system can understand these anonymous data to authenticate and identify the RFID tag.

An example protocol to demonstrate this can be found in [94]. Figure 3.3 is a simplified version to illustrate this protocol.

Here $H(.)$ is a cryptographic hash function and n is the total number of tags in the system. In the protocol, every time when an RFID tag is queried, it will reply with an $metaID$. In that way, the real ID of the tag is kept secret from adversary, who do not

Tag$\{metaID\}$		Reader$\{ID_i, K_{ID_i}\}$
	$\xleftarrow{Request}$	
	$\xrightarrow{metaID}$	for $i \in \{1, n\}$, find $\{ID_i, K_{ID_i}\}$ s.t. $H(K_{ID_i}) = metaID$

Figure 3.3: Personal privacy preserving protocol

Tag$\{K_{ID}\}$		Reader$\{ID_i, K_{ID_i}\}$
Generates random R	$\xleftarrow{Request}$	
$C = H(K_{ID}\|\|R)$	$\xrightarrow{R, C}$	for $i \in \{1, n\}$, find $\{ID_i, K_{ID_i}\}$ s.t. $H(K_{ID_i}\|\|R) = C$

Figure 3.4: Location privacy preserving protocol

have any knowledge about the linkage between ID and $metaID$. In contrast, a legitimate reader that is connected to the back-end server can access all these information such that the real ID (so as any other related information) of a tag can be looked up.

3.4.2 Location Privacy

This privacy concern appears in both [94] and [41]. This is another most important privacy concern in RFID. It is referred to "*tracking of individuals*" or "*behavioral tracking/personal identification by tracing tag IDs*" as in [74] and defined as *Tag output must be indistinguishable from truly random values. Moreover, they should be unlinkable to ID of the tag. If the adversary can distinguish that a particular output is from a target tag, she can trace the tag.*

It is noted that the $metaID$ protocol in [94] described previously, although it can protect *personal privacy*, it fails to protect *location privacy*. Given that a tag will always reply with the same $metaID$, any adversary can distinguish this $metaID$ from a random number (i.e. the $metaID$ is itself a unique pointer). Although the adversary learns nothing about the true ID and the related information of the tag, she can location her target whenever she reads the same $metaID$. We will instead, use the enhanced protocol in [94] to illustrate this privacy concern. Figure 3.4 is a simplified version of this protocol.

Here || means concatenation. Every time when the tag is being queried, it generates a new random number and hash it with its tag key. Now since the output from the tag

Tag$\{s_1 = H(K_{ID})\}$		Reader$\{ID_i, K_{ID_i}\}$
	$\xleftarrow{\quad Request \quad}$	
$C = F(s_k)$	$\xrightarrow{\quad C \quad}$	for $i \in \{1, n\}$, find $\{ID_i, K_{ID_i}\}$
$s_{k+1} = H(s_k)$		s.t. $F(H^k(K_{ID_i})) = C$

Figure 3.5: Forward privacy preserving protocol

is random every time, trying to trace a particular tag is impossible. Recall from the previous section, however, this protocol is vulnerable to the replay attack. An attacker can replay the random response R, C every time when a reader sends a query.

3.4.3 Forward Privacy

Since RFID tags are not tamper resistant, it becomes a privacy concern when an adversary obtained the data stored inside the memory of the tag. If there is any stored personal information, it is revealed to the adversary (i.e. breaking *personal privacy*). If the adversary has also recorded the pervious communications between the tag and reader, it may be possible for the adversary to link up the past authentication sessions of the tags (i.e. breaking the *location privacy*). Hence making sure that the privacy protections can be pushed forward from the beginning to the point where the tag secret is revealed is of the same importance especially in the low-cost RFID tag systems. [74] stated this *forward privacy* as *"Even if the adversary acquires the secret tag data stored in the tag, she cannot trace the data back through past events in which the tag was involved. Needless to say, the adversary who only eavesdrops on the tag output, cannot associate the current output with past output."*.

We will use the protocol in [74] as an example here to demonstrate this. Figure 3.5 shows this protocol.

Here $H^k(.)$ denotes the number of hashes applied. i.e. for $k = 3$, it means $H(H(H(K_{ID_i})))$. There are two cryptographic hash functions $H(.)$ and $F(.)$. Each time when a tag is being interrogated, its internal secret information s_k is updated to a new value by $H(.)$ at the end. This is here for achieving *personal privacy* and *forward privacy* such that even if the current s_k value is revealed to an adversary, she cannot find out the previous s_j values for $j < k$ because $H(.)$ is non-invertible (one-way). While the function $F(.)$ is here for *location privacy* and *forward privacy*. If the tag simply outputs $s_k = H^k(K_{ID})$, $s_{k+1} = H(s_k)$, ... in each authentication session, then any adversary can identify the same tag as she computes $s_{k+2} = H(s_{k+1})$ and see if

there is a tag that outputs s_{k+2}.

3.5 Security Properties for RFID Authentication Protocols

We have just looked at the three privacy issues in RFID. *Personal privacy* is easy to protect, simply moves all the personal information from the tag memory to the back-end server and stores only a reference key in the tag memory. *Location privacy* is also easy, just makes sure the outputs of the tag are randomised, then the tag cannot be traced. *Forward privacy* is relatively the hardest to guarantee. The tag key has to be constantly changed and the tag outputs have to be appeared random. But even with these, it may still be traceable in front of a timing attack adversary who can spot the subtle difference of the time taken to identify a desynchronised tag than a normal tag. The hard problem here is how to make an RFID authentication protocol that cannot be made desynchronised and provides forward privacy at the same time? This is one of the research topics in this thesis.

3.5.1 Example Scenario

To better understand the traceability problem in RFID, it is best to have an example scenario in mind:

*A military officer, named **Bob**, is going to carry out an important mission, which is to deliver an advance high-tech weapon from the military base to a strategic site. The weapon is equipped with RFID tag for authentication purpose. During the delivery, Bob's truck has to go through some check points, so that Bob's assistance, **Alice**, who is monitoring the progress of the mission at the military base, can verify the integrity of the weapon through RFID scans at these check points. With the help of a secure database server at the base, she can identify and authenticate the weapon based on the RFID signals obtained. Fearing that the final destination of the weapon will be exposed, they have arranged some decoy trucks to join up during the mission. In the last check point, they have possibly unloaded and swapped the actual weapon with one of the decoys and then they split up and go on different routes. An adversary, **Eve**, driving close to these trucks on the road is now trying to find out where the destination of the weapon is. She is equipped with all the required RFID equipments to interact with the RFID tags on the weapon and the decoys and also the readers at the check points. So she*

can either eavesdrop whenever Alice makes a scan or she can readily scan the tags and even interfere with Alice's scans. She has been scanning since the beginning to learn about the signal content obtained from the weapon tag. She is going to use this trained knowledge to identify the truck with the actual weapon out from those decoys. She is now on her own to relate each piece of information she obtained through eavesdropping and/or interactions to decide which truck she should follow so that the final destination is revealed.

In the above scenario, for simplicity, we assume there are only two trucks on the final leg. Of course, Eve could have successfully guessed the correct truck to follow with a probability of $\frac{1}{2}$ by a blind guess. The RFID protocol here is to limit Eve from increasing her success probability non-negligibly higher than $\frac{1}{2}$. Initially before the decoy trucks join up, by eavesdropping on Alice's scans or actively scan the truck, Eve can obtain a tag response that comes from the actual weapon. Hence in an RFID protocol where a tag is designed to always give out the same response will simply allow Eve to win. A more secure RFID protocol would have encoded the responses in a way that only with the help of the secure back-end server should these responses be properly decoded. As a result, the best Eve can do is to collect and study each response from the weapon tag she obtained before the crew split up and relate these responses to the responses now scanning from each of the trucks. This setups a challenge to Eve: *To find out her target using the obtained RFID information from two trucks.* Carefully investigating the matter, we can have four different security properties to assure Bob that Eve will not success.

3.5.2 Anonymity

This is the most basic property: *The outputs of the RFID tag should* ***appear random*** *to anyone without access to the secure database server.* i.e. each scan to a tag T will result in a random string R. Formally speaking:

$$Pr[R : R \leftarrow \texttt{Scan}(T), R \in \mathcal{M}] \approx Pr[R : R \xleftarrow{rand} \{0,1\}^*]$$

This says that the probability of getting R within the output message space $\mathcal{M}$ from $\texttt{Scan}(T)$ is approximately the same as the probability of getting a random string drawing from random space. This implies that an adversary has only a negligible chance to distinguish between a meaningful output and a random string.

3.5.3 Unlinkability

Having *anonymity* only is not enough to assure Bob. It is sufficient to protect tags from being traced if the adversary only has non-continuous access to tag responses. e.g. Consider a protocol that the tag outputs $\{R_i, r_i\}$ where r_i is a random number and $R_i = H(R_{i-1}||r_i)$ with $H(.)$ being a secure hash function. Clearly, the outputs all appear random thanks to the secure hash function and an adversary who obtained only $\{R_i, r_i\}$, $\{R_{i+2}, r_{i+2}\}$, $\{R_{i+5}, r_{i+5}\}$, ... non-consecutive responses will have no way to trace the tag.

But if the adversary is more powerful, who is able to monitor the communication channel continuously, then by obtaining the tag outputs $\{R_i, r_i\}$, $\{R_{i+1}, r_{i+1}\}$, $\{R_{i+2}, r_{i+2}\}$, ..., she can easily verify if this is the same tag by checking if $R_{i+1} = H(R_i||r_{i+1})$ or if $R_{i+2} = H(R_{i+1}||r_{i+2})$. Hence we should add another property, namely *unlinkability*: *The outputs of the RFID tags in each scan should be* ***unpredictable*** *to anyone without access to the secure database server.* Formally speaking:

$$\forall i, j, R_i, R_j \in \mathcal{M}, Pr[R_i : R_i \leftarrow \texttt{Scan}(T)] = Pr[R_j : R_j \leftarrow \texttt{Scan}(T)]$$

This says that the probability of outputting R_i in the ith scan is the same as outputting R_j in the jth scan, even if $j = i + 1$. This implies that every output from a tag in the output message space $\mathcal{M}$ is evenly possible.

3.5.4 Indistinguishability

Having the two properties above are not enough to assure Bob. They are sufficient to protect tags from being traced if the adversary does not have access to another reference tag. e.g. Consider a modified OSK protocol [74] that the tag outputs $\{C_i, C'_i\}$ and $C_i = F(R_i), C'_i = F(R'_i), R_{i+1} = H(R_i), R'_{i+1} = H(R'_{i+1})$ where $H(.)$ and $F(.)$ are both secure hash functions. Clearly, the outputs are all appear random and are unlinkable thanks to the one-wayness of the hash functions.

So far the adversary has only considered the information leak from a single tag with respect to a random source. Now if she turns to consider the outputs from another tag instead of a random source, then she is able to relate the two tags and finds out her target. Let the outputs of tag A be $\{C_i, C'_i\}$ and the outputs of tag B be $\{\hat{C}_i, \hat{C}'_i\}$. Say the tags will initialise R_1 ($\hat{R}_1$) as r and R'_1 ($\hat{R}'_1$) as $\frac{r+2}{2}$. If tag A has $r = 2$ and tag B has $r = 4$ then we have $\{R_1, R'_1\} = \{2, 2\}$ and $\{\hat{R}_1, \hat{R}'_1\} = \{4, 3\}$. This glitch causes tag A to always output $C_i = C'_i$. If a random tag is given to the adversary at the end, she can instantly point out tag A from tag B by checking if $C_i = C'_i$. Hence we have

indistinguishability to guarantee: *The outputs of* ***any two RFID tags*** *in each scan should be* ***indistinguishable*** *to anyone without access to the secure database server.* Formally speaking:

Define a game for indistinguishability (**Game-IND**):

1. An adversary is allowed to interact with any tag she likes, obtaining their outputs $R_i^{T_*}$.

2. After q_1 interactions, she has to nominate two tags T_A and T_B in which she thinks she can distinguish them.

3. One of the tags is picked randomly and gave back to her as $T_\perp$ and the other one is discarded.

4. She can then interacts with this unknown tag $T_\perp$ and the other remaining tags, obtaining their outputs.

5. After q_2 interactions, she has to guess whether $T_\perp = T_A$ or $T_\perp = T_B$, with $q_1 + q_2 \leq q$.

we say that for all $poly(k)$ time adversary bounded by q:

$$Pr[success\ in\ \textbf{Game-IND}] = \frac{1}{2} + \frac{1}{poly(k)}$$

This says that the probability of guessing correctly whether $T_\perp = T_A$ or $T_\perp = T_B$ in q number of interactions is only negligibly better than making a blind guess. This implies that an adversary can only get very little advantage through the interactions. Note also that what we say about something that is indistinguishable also includes the sense that it is random and unpredictable because something that is static or predictable can be distinguished easily. So any system that has the indistinguishability property also has the anonymity and unlinkability properties.

3.6 Conclusion

Low-cost RFID tags are subject to many security threats: cloning, eavesdropping, tracking etc. Under the current adversary ability assumptions, it seems that it is impossible for these constrained tiny chips to avoid some of the stronger attacks on the protocol level. Clearly, security should not be built solely on the protocol level. In fact, security threats can be mitigated with the help of other parts of the systems, like

an honest officer can help spotting a counterfeited tag. Besides, we believe through user educations, the public awareness to the potential security threats in RFID can be raised, which also helps mitigate the impacts. On the other hand, what interested us most is the performance of RFID authentication protocols on the tag tracking problem. Location privacy is the most important privacy issue in RFID application and a secure solution is yet to be found. To make the situation even more complicate, we are unavoidably compelled to consider also *forward privacy* (tag compromise problem) and *timing attack* (a side-channel attack) due to the unique characteristics of low-cost RFID tags. These problems are hard but challenging, which also form the core part of this work.

Chapter 4

Our Findings on RFID Privacy Modellings

Chapter Overview

In the previous chapter, we introduced RFID protocols. We looked at the system requirements and summarised most of the security goals commonly seen in secure RFID authentication protocols. This gives us a picture on the possibilities that one may expect to see in these protocols. In this chapter, we will move onto discuss how we can formalise these security goals into formal models for security evaluations. First we will give a critical review on some of the early notable works about RFID privacy modelling, especially the famous Vaudenay privacy model [93]. Then we will present our findings on their inconsistency and discuss some contradictory results that we obtained. Part of the results in this chapter has been published in [70] at ESORICS 2008. We have the following contributions in this chapter:

- We give a critical review on most of the early notable works on RFID privacy modellings
- We select two privacy models and show their inconsistence by constructing protocols that have different results in each of them
- We fully review the Vaudenay privacy model and comment on its results
- We revise the Vaudenay privacy model and present our revised version based on some reasonable assumptions
- We present our different results comparing to the Vaudenay model and justify our findings

4.1 RFID Privacy Models

There are easily more than hundreds of proposed RFID protocols according to the extensive RFID related publication list maintained in [87]. These RFID protocols are of various types: tree-based, hash-based, symmetric, asymmetric, stateless, stateful, lightweight, ultralightweight, deterministic, probabilistic, etc. A lot has proposed, yet we are seeing more to come. It has been a phenomenon in the literature that in a year a new protocol is published then there is/are paper(s) attacking it in the next year. Some are fundamentally flawed to being with, but more are because of the different understandings or definitions to the term "privacy" in RFID applications; furthermore, what kind of adversary is being considered in the paper can also turn the tables. Take PKC as an example, *semantic security* was used as a merit until *IND-CCA2* was introduced.

Security protocols nowadays are required to be formally proven secure under a formal model before they can be widely accepted. It is very important for us to emphasise that only with a formally defined model, we can be sure that given an RFID protocol, one can tell if it guarantees the claimed security or not and to what extend. RFID, however, as an emerging topic in cryptography, lacks a unified common model to formalise the security and privacy requirements for its special environment. An RFID protocol can be proven secure in a paper on its own model, but the problem is whether we agree that to be a good RFID model? Are the security goals properly set? Is the adversary given with reasonable capabilities? Does it captures most of the real world attacks? How is the application of the underlying system adequately defined?

A formal model serves to model any possible real world behaviours of an adversary does to the system. By defining what are the resources available in the system, which resources can be controlled by the adversary, what are the goals of the adversary and how does she claim to be succeed, we can show that an underlying system is secure or not against such adversary. To put it together, one is expected to see the following components in an RFID modelling:

- **System model** – defines the system environment, parameters, constrains, assumptions and application
- **Adversary model** – defines the capabilities of the adversary, i.e. to capture the targeting form of real world attacks that the protocol aimed to protect from
- **Security model** – defines the security goals of the protocol, i.e. short-listing the security features that the protocol aimed to provide

Since the main concern of the many security threats in RFID is privacy, we will refer to the three models above as *the privacy model*. In fact, most of the proposed RFID protocols aimed to restore privacy in the constrained RFID environment.

4.2 Early Works on RFID Modellings

We started our research in 2008, interestingly, among the hundreds of published works in RFID researches, there were only very few attempts to provide a unified privacy modelling for RFID. RFID has its uniqueness in its constrained environment, especially with those low-cost RFID tags, which should receive a different cryptographic view then those more capable wireless devices. Hence, most of the case a direct reuse of other wireless application modellings is not fit for the purpose of evaluating RFID applications, unless we are looking at mid to high cost range RFID tags, but that does not worth the effort for a meaningful research. In particular, low-cost RFID tags are always considered to bear no tamper resistance, such that we have to also consider the situation when a tag falls into the hands of an adversary, the stored content (including any secret keys) inside the memory chip of the tag can always be extracted by the adversary. This key-exposure-by-default setting is very essential in RFID applications and plays a major role in RFID privacy modellings. Compared to other wireless applications, security against key exposure may only be an added security feature. This shows us the need of a unique privacy model for RFID. Of course, some proposed RFID protocols come with their own privacy model, but because they are not created in mind to be generalised into wider applications, porting them to other protocols may not give fruitful results. To the best of our knowledge, we could identify six notable works purely devoted for general RFID privacy modelling. We will give an overview to some of them below.

4.2.1 The Ari Juels Model

Dated the earliest, Juels's work [43] was the first among the six to formalise a practical privacy model with the physical boundaries of low-cost tags in mind. To reflect the constrained environment of RFID in his model, the adversary's abilities are also restricted. What he proposed in the paper is the notion *minimalist cryptography* for those low-cost (five cents) tags. The given example protocol uses only pre-stored pseudonyms in the tag shared with the reader and uses XOR operation to update these pseudonyms with one-time pads. For this simple bitwise only, without any cryptographic operation

protocol to remain secure, the adversary is restricted to have only *limited successive tag queries* and *limited interleaving* between tags and readers. The former aimed to solve the tracking problem and says due to the close read range of RFID tags, a passerby attacker would have difficulties to harvest more than two pseudonyms in most cases. The number of successive tag queries can also be controlled by prolonging the tag response time through throttling mechanisms such that a tag can have more than enough time to be *refreshed* back to a secure state with new pseudonyms before all the unused pseudonyms are harvested. The latter aimed to solve the man-in-the-middle attack thanks to the mobility of tags (i.e. tags wearers) and reader throttling (e.g. refuse to engage in rapid succession with the same tag). Under these restrictions, tags can always have endless supply of new pseudonyms, so that tag tracking by matching any repeated pseudonyms is not possible.

We agree that this restricted model indeed guarantees non-trackable tag movements, mainly due to the assumption *limited successive tag queries*. But the question is how close this model captures the real world scenarios. Clearly, an online attacker will be affected by the close read range of tags and the proposed throttling mechanisms, but if we consider also an offline attacker, who is capable to launch a "holiday attack" (accessing the target tag when the tag wearer is gone for a holiday as analogous to lunch-time attack), that gives a much longer time for the attacker to harvest more pseudonyms out from the tag. If we further consider an attacker who is capable to spot the subtle response time difference between a normal tag and the target tag, which has recently been queried a number of times to prolong its next response time due to the throttling mechanism, then tracking will become possible again. These attacks do not require any empowered adversaries to launch and are quite reasonable in a real world scenarios.

4.2.2 The Gildas Avoine Model

Next in line is the work by Avoine (first obtainable at [5] and later extended into a technical report in [6]). According to [6], this work was the first to formalise the *traceability* problem in RFID. There he defined several *oracles* to represent the abilities of the adversary as in other formalisations. Accessing these oracles is equivalent to the interaction between the adversary and the RFID system. Instead of a restrictive adversary modelling, Avoine considered a more general adversary commonly seen in RFID by giving her an oracle access to reveal the content of a tag, only with the limitation that such oracle can only be the last oracle to be accessed (i.e. no more

oracle accesses are allow once the reveal oracle is called). All together there are five oracles, other than `Reveal` (`R`), the rest are `Query` (`Q`), `Send` (`S`), `Execute` (`E`) and `Execute`* (`E`*). The traceability problem in RFID is formalised into two adversarial goals, namely *Existential Untraceability* (`Existential-UNT`) and *Universal Untraceability* (`Universal-UNT`). Both are defined as game-based security by playing a game with the adversary on distinguishing the target tag out from two challenge tags. Thus the privacy of a given RFID protocol is attained if the winning probability of the adversary in either of the games is only negligibly higher than 2^{-1} over all the random choice of tags. `Existential-UNT` models an adaptive adversary who can choose the attack instances to interact with the two challenge tags, while `Universal-UNT` models a non-adaptive adversary who receives the interaction results of unknown instances for each of the two challenge tags. The discontinuity between instances is to model a real world adversary who may not monitor the target tag constantly but only temporarily. Intuitively, breaking `Existential-UNT` means there is at least one possible instance that an adversary can exploit to trace a tag and the opposite is there exists no instance that an adversary can trace a tag; breaking `Universal-UNT` means a tag can be traced at any instances and the opposite is there exists at least one untraceable instance. By definition existential untraceability is stronger (i.e. the adversary is given with more power) than universal untraceability, hence the former is harder to achieve. Therefore an RFID protocol secures against `Existential-UNT` attackers is automatically secures against `Universal-UNT` attackers, or in other words, an RFID protocol that fails `Universal-UNT` is also not `Existential-UNT`. But failing `Existential-UNT` does not necessarily fail `Universal-UNT`. Hence the highest attainable privacy protection in [6] is `Existential-UNT`.

We will cover more about Avoine's model in the next section. What catches our eyes is the flexibility of this model. Especially the ability to choose the attack instances by the adversary in the `Existential-UNT` game and the ability to choose the learning instances in both games are not commonly seen in other RFID formalisations. Also, with the different combinations of the choice of oracles formalise a variety of adversaries. For example, there can be `UNT-E`, `UNT-Q`, `UNT-QSE`, `UNT-QSER` adversaries, in which they formalise an eavesdropping only adversary, a querying only adversary, an actively interacting adversary and an even stronger tampering adversary respectively. Notice that when the model says `Existential-UNT` implies `Universal-UNT`, it is over the same choice of oracles. For instance, `Existential-UNT-QSE` does not implies `Universal-UNT-QSER`,

but `Existential-UNT-QSE` implies `Universal-UNT-QSE`. In addition, the chronicle order of learning and attack instances further defines `Existential`$^+$/`Universal`$^+$ and `Existential`$^-$/`Universal`$^-$ adversaries, where it depends on whether the attack instances *all* (either chosen by or given to the adversary) happen chronologically later than or before *all* the learning instances respectively. If they *all* happen before *all* the learning instances, it is essentially a *forward* privacy attacker; otherwise, it is a *backward* privacy attacker. This shows that this is a very flexible model that can accommodate many real world attack scenarios. Several RFID protocols are assessed using this model and only the famous OSK hash-based protocol [74] is the most secure, which provides both `Existential-UNT-QSE` privacy and `Forward`(`Universal`$^-$)`-UNT-QSER` privacy. Another hash-based protocol WSRE [94] provides only `Existential-UNT-QSE` privacy but not `Forward`(`Universal`$^-$)`-UNT-QSER`. These are examples to clearly show the separation results between privacy protections offered by different RFID protocols. However, the usefulness of all the oracles are not fully demonstrated, like the oracle `E`* is not used at all.

4.2.3 The Xiaolan Zhang and Brian King model

The ZK model (not to confuse with a zero-knowledge model) was first published in [97] and later extended into a journal version in [99], our review is based on the latter. Instead of focusing on the abilities of the adversary, the ZK model tried to parameterise some security quantities in RFID. First, it defined five security properties, namely *Perfect Identification of tag Identity* (`PII`), *Authorized Perfect Identification of tag Identity with κ-history* (`APII`), *Authorized Perfect Identification of tag Bearers with κ-history* (`APIB`), *Indistinguishability of tag Identity with κ-history* (`INDI`) and *Indistinguishability of tag Bearers with κ-history* (`INDB`). `PII` essentially means tag availability, `APII` and `APIB` mean tag and owner authentication, `INDI` means tag data privacy and `INDB` means indistinguishability of owner. Next, it proposed to parameterise the acceptable error tolerance for non-legitimate tags (δ), the rejection error tolerance for legitimate tags (ϵ), the maximal adversary advantage obtainable (γ) and the bounded history size (κ). With all these security parameters, there can be $(\delta, \epsilon, \gamma, \kappa)$ `RFID APII/APIB` and/or (γ, κ) `RFID INDI/INDB` secure RFID protocols.

In fact, we are not too keen on the non-deterministic RFID protocols. During our research, we only came across the HB genre and its variants [35, 36, 48] and the protocols that build on top of PUF [12, 32] are of this category. They are not of our interest, firstly because they are non-deterministic, meaning that even a legitimate tag

can get rejected by a legitimate reader under normal circumstances, not related to any adversarial attacks. And secondly because they all use static shared symmetric keys, which is definitely not privacy friendly in front of a tampering adversary. Therefore, under the ZK model, we will have to consider only $\delta = \epsilon = 0$, which is $(0, 0, \gamma, \kappa)$ `RFID APII/APIB`. This effectively reduces to a common probabilistic polynomial time-bounded adversary with a winning advantage γ under the system security parameter κ. Since the Juels and Weis model [49] also features this kind of adversary and is stronger, we will focus on their model instead of this ZK model.

4.3 Inconsistencies in RFID Privacy Models

In an effort to design a widely accepted privacy model for RFID, we can see there are different innovative designs proposed from the previous section. But before a model is deemed widely accepted, we found inconsistencies in these models. A contradictive result is shown by the well known OSK protocol [74] where this protocol was shown to be secure under the Avoine model [6], but is considered insecure in the Juels and Weis model [49]. In this section, we will look at these two privacy modellings of RFID in details and discuss their implications. To better compare the two models, we have simplified some of the notations a bit in order to focus on their core structures.

4.3.1 Untraceability in the Avoine Model

System model

There are communication channels between the back-end database and readers, and also between readers and tags. Only the communication between readers and tags is relevant to the traceability issue in RFID. Hence the communication channel between the back-end database and readers is assumed to be secure or of no interest to an adversary. The communication channel between readers and tags are further separated into two channels: one from a reader to a tag (the forward channel) and one from a tag to a reader (the backward channel). The contents of the data stored in the memory of tags is also considered as a separate channel (the memory channel). Each of these channels can be read or written by an adversary, except the memory channel where writing to it is not allowed. It is also assumed that the contents of the tags are independent.

Adversary model

The adversary's abilities are limited to the following oracles to obtain a realistic and applicable model. $\mathcal{A}$ denotes the adversary. A tag is denoted as $\mathcal{T}$, a reader as $\mathcal{R}$ and a protocol as $\mathcal{P}$. An instance of a protocol $\mathcal{P}$ from the point of view of tag and reader is denoted as $\pi^i_{\mathcal{T}}$ and $\pi^j_{\mathcal{R}}$ respectively.

- `Query`$(\pi^i_{\mathcal{T}}, m_1, m_3)$: this oracle models $\mathcal{A}$ sending a message m_1 to $\mathcal{T}$ through the forward channel and subsequently sending it the message m_3 after a response from $\mathcal{T}$ is received.
- `Send`$(\pi^j_{\mathcal{R}}, m_2)$: this oracle models $\mathcal{A}$ sending the message m_2 to $\mathcal{R}$ through the backward channel and receiving its response.
- `Execute`$(\pi^i_{\mathcal{T}}, \pi^j_{\mathcal{R}})$: this oracle models $\mathcal{A}$ executing an instance of $\mathcal{P}$ between $\mathcal{T}$ and $\mathcal{R}$, obtaining the transcript of the communication on both the forward and the backward channels.
- `Execute*`$(\pi^i_{\mathcal{T}}, \pi^j_{\mathcal{R}})$: this oracle models $\mathcal{A}$ executing an instance of $\mathcal{P}$ between $\mathcal{T}$ and $\mathcal{R}$, but only the messages sent on the forward channel is obtained.
- `Reveal`$(\pi^i_{\mathcal{T}})$: this oracle models $\mathcal{A}$ revealing the content of the date stored in $\mathcal{T}$'s memory channel at instance $\pi^i_{\mathcal{T}}$. This oracle can be accessed only once and after that `Query`, `Send`, `Execute`, and `Execute*` cannot be accessed anymore.

Untraceability definition

Let $\mathcal{R}$ denotes the reader, $\mathcal{T}_1$ and $\mathcal{T}_2$ denote the two tags, $\pi^i_{\mathcal{T}_*}$ and $\pi^i_{\mathcal{R}}$ denote the interactions obtained (i.e. messages sent and received) from a tag $\mathcal{T}_*$ at instance i and from the reader $\mathcal{R}$ at instance i. Let $\mathcal{I} = \{i_1, i_2, ..., i_n\}$ be a set of instance indexes and $\Pi^{\mathcal{I}}(\mathcal{T}_*) = \{\pi^i_{\mathcal{T}_*} | i \in \mathcal{I}\}$ and $\Pi^{\mathcal{I}}(\mathcal{R}) = \{\pi^i_{\mathcal{R}} | i \in \mathcal{I}\}$ be sets of interactions obtained from a tag $\mathcal{T}_*$ and the reader $\mathcal{R}$ at instances $\mathcal{I}$ respectively. Let $\mathcal{O}(\mathcal{I}, \mathcal{T}_*)$, $\mathcal{O}(\mathcal{I}, \mathcal{R})$ be oracles access to tag $\mathcal{T}_*$ and reader $\mathcal{R}$ that return the interactions $\Pi^{\mathcal{I}}(\mathcal{T}_*)$ and $\Pi^{\mathcal{I}}(\mathcal{R})$ respectively. Let q be the total number of allowed attack instances. Finally, let $\mathcal{C}$ be a uniformly random function that models a challenger who gives the adversary $\mathcal{A}$ necessary inputs. A re-phased version of Avoine's untraceability game is defined in table 4.1.

The advantage of $\mathcal{A}$ for a given protocol $\mathcal{P}$ is $\mathbf{Adv}^{untraceability}_{\mathcal{P}}(\mathcal{A}) = 2 \cdot \mathbf{Pr}(\mathcal{T}_b = \mathcal{T}) - 1$. If the advantage is negligible then $\mathcal{P}$ is said to be $q - \mathbf{UNT_{Avoine}}$ secure.

$\mathbf{Exp}_{Avoine}^{untraceability}[q, \mathcal{C}]$:

Learning stage

1. $\mathcal{A}$ requests $\mathcal{C}$ thus receiving her attack target $\mathcal{T}$
2. $\mathcal{A}$ chooses instances $\mathcal{I}$ where $|\mathcal{I}| = n$
3. $\mathcal{A}$ calls $\mathcal{O}(\mathcal{I}, \mathcal{T})$ and/or $\mathcal{O}(\mathcal{I}, \mathcal{R})$ to receive $\Pi^{\mathcal{I}}(\mathcal{T})$ and/or $\Pi^{\mathcal{I}}(\mathcal{R})$

Challenge stage

1. $\mathcal{A}$ requests $\mathcal{C}$ thus receiving her challenges $\mathcal{T}_1$ and $\mathcal{T}_2$
2. $\mathcal{A}$ chooses instances $\mathcal{I}_1$ and $\mathcal{I}_2$ where $|\mathcal{I}_1| + |\mathcal{I}_2| + n \leq q$ and $(\mathcal{I}_1 \cup \mathcal{I}_2) \cap \mathcal{I} = \emptyset$
3. $\mathcal{A}$ calls $\mathcal{O}(\mathcal{I}_1, \mathcal{T}_1)$ and/or $\mathcal{O}(\mathcal{I}_1, \mathcal{R})$ and $\mathcal{O}(\mathcal{I}_2, \mathcal{T}_2)$ and/or $\mathcal{O}(\mathcal{I}_2, \mathcal{R})$ to receive $\Pi^{\mathcal{I}_1}(\mathcal{T}_1)$ and/or $\Pi^{\mathcal{I}_1}(\mathcal{R})$ and $\Pi^{\mathcal{I}_2}(\mathcal{T}_2)$ and/or $\Pi^{\mathcal{I}_2}(\mathcal{R})$
4. $\mathcal{A}$ decides which of $\mathcal{T}_1$ or $\mathcal{T}_2$ is $\mathcal{T}$ and output her guess $\mathcal{T}_b$

Table 4.1: Untraceability experiment in the Avoine privacy model

Implication

Avoine defined *Existential Untraceability* and *Universal Untraceability* in [6]. The only difference is whether $\mathcal{A}$ or $\mathcal{C}$ controls the choice of $\mathcal{I}_1$ and $\mathcal{I}_2$. Since *existential* is a less restrictive definition than *universal*, and hence stronger, we selected it for review here. According to the definition above, we can see that the attack target is fixed at the first step (by a random choice of $\mathcal{C}$). The adversary then choose her attack instances (need not be consecutive) to interact with either the target tag or the reader or both freely. She just need to weight and balance her number of attack instances in the learning stage and the challenge stage such that the total number does not excess q. During the challenge stage, she will receive two tags as her challenges, one of them being the original attack target. She then interacts further with these tags and/or with the reader if the total number of interactions has not excess q. Finally from all the interactions she obtained, she needs to decide whether $\mathcal{T}_1$ or $\mathcal{T}_2$ is $\mathcal{T}$.

Notice that we did not specific what exactly are the available oracle accesses (i.e.$\mathcal{O}(\mathcal{I}, .)$) to the tags and reader by the adversary. This is only a choice of the RFID system. Depending on different applications and scenarios, there can be different oracle accesses that are available to the adversary to model her ability. We aimed to compare on the untraceability definitions only so it is enough at its current simplified form. For instance, we can allow the adversary to query on the tags and reader,

or execute the protocol faithfully just as an eavesdropper will do. This is the same for the following security modelling.

4.3.2 Indistinguishability in the Juels and Weis Model

System model

The system model is nearly the same as Avoine excepts that Juels and Weis consider the contents of tags can be correlated and the adversary is allowed to write into the memory channel.

Adversary model

It is simpler than Avoine but with some essential differences. The adversary is stronger by giving her a new ability to write new content into the memory of tag.

- `ReaderInit()`: this oracle models $\mathcal{A}$ requesting $\mathcal{R}$ to produce a fresh challenge c_j
- `TagSend(`$\mathcal{T}_j, c_j$`)`: this oracle models $\mathcal{A}$ sending the challenge c_j from $\mathcal{R}$ to $\mathcal{T}_j$ through the forward channel and receives a response r_j from $\mathcal{T}_j$
- `ReaderSend(`$\mathcal{T}_j, r_j$`)`: this oracle models $\mathcal{A}$ sending the response r_j from $\mathcal{T}_j$ to $\mathcal{R}$ through the backward channel and receives its result
- `SetKey(`$\mathcal{T}_j, k_j$`)`: this oracle models $\mathcal{A}$ writing a new key into the memory of $\mathcal{T}_j$ and at the same time overwrites the old key. $\mathcal{T}_j$ also outputs the old key hence $\mathcal{A}$ can reveal it

Indistinguishability definition

Let $\mathcal{O}(\mathcal{T}_*)$ and $\mathcal{O}(\mathcal{R})$ be oracles access to a tag $\mathcal{T}_*$ and the reader $\mathcal{R}$, which return the interactions $\pi_{\mathcal{T}_*}$ and $\pi_{\mathcal{R}}$ respectively. Let q be the total number of oracle access and $\mathcal{C}$ be a uniformly random function that models a challenger who gives the adversary $\mathcal{A}$ necessary inputs. Juels and Weis's indistinguishability is defined as the game in table 4.2.

The advantage of $\mathcal{A}$ for a given protocol $\mathcal{P}$ is $\mathbf{Adv}_{\mathcal{P}}^{indistinguishability}(\mathcal{A}) = 2 \cdot \mathbf{Pr}(b' = b) - 1$. If the advantage is negligible then $\mathcal{P}$ is said to be $q - \mathbf{IND}_{\textbf{Juels and Weis}}$ secure.

$\mathbf{Exp}^{indistinguishability}_{Juels\ and\ Weis}[q, \mathcal{C}]$:

Learning stage

1. $\mathcal{A}$ can call a total of $n \leq q$ $\mathcal{O}(\mathcal{T}_*)$ and/or $\mathcal{O}(\mathcal{R})$ to receive $\pi(\mathcal{T}_*)$ and/or $\pi(\mathcal{R})$ following her strategies

Challenge stage

1. $\mathcal{A}$ selects two tags $\mathcal{T}_1$ and $\mathcal{T}_2$ and submits them to $\mathcal{C}$ and receive back $\mathcal{T}_b$ from $\mathcal{C}$
2. Remove $\mathcal{T}_1$ and $\mathcal{T}_2$ and add $\mathcal{T}_b$ to the current set of tags
3. Again $\mathcal{A}$ can call the remaining number of allowed oracle access $\mathcal{O}(\mathcal{T}_b)$ and/or $\mathcal{O}(\mathcal{R})$ to receive $\pi(\mathcal{T}_b)$ and/or $\pi(\mathcal{R})$
4. $\mathcal{A}$ decides whether $\mathcal{T}_b$ is $\mathcal{T}_1$ or $\mathcal{T}_2$ and output her guess b'

Table 4.2: Indistinguishability experiment in the Juels and Weis privacy model

Implication

Juels and Weis defined indistinguishability in [49]. Juels and Weis allow the adversary to adaptively choose her challenges. She can decide on her own choice of tags $\mathcal{T}_1$ and $\mathcal{T}_2$ (if the system has more than two tags, otherwise, they will be a real and fake tags) based on the information gathered in the learning stage. According to her strategies, she may find $\mathcal{T}_1$ and $\mathcal{T}_2$ more vulnerable than the other tags such that she can increase her chance of success in the experiment. $\mathcal{C}$ randomly picked one of the submitted tags and returns it as the adversary's challenge. Further oracle access is allowed if the total number does not excess q. Finally she needs to decide whether $\mathcal{T}_b$ is $\mathcal{T}_1$ or $\mathcal{T}_2$.

We have simplified the original indistinguishability definition in [49] by removing the system specific available oracle accesses. There is actually a `SetKey`$(\mathcal{T}, k)$ oracle access to a tag $\mathcal{T}$, which allows the adversary to set the internal key of $\mathcal{T}$ to her choice of value k. This oracle is suggested to model an adversary's ability to corrupt "correlated" tags. Since the tags in our system are independent to each other, being able to set a new key on some non-attack target brings no advantage to the adversary. On the other hand, setting a new key on the attack target makes the adversary win the indistinguishability experiment instantly. This powerful adversary ability is not necessary in our context.

Tag$\{K_{ID}\}$		Reader$\{ID_i, K_{ID_i}\}$
	$\xleftarrow{\quad Request \quad}$	
$C = PRNG_k(K_{ID})$	$\xrightarrow{\quad C \quad}$	for $i \in \{1, n\}$, find $\{ID_i, K_{ID_i}\}$ s.t. for $j \in \{0, q\}$, find $PRNG_{k+j}(K_{ID_i}) = C$

Figure 4.1: Protocol secure in the Juels and Weis model but not in the Avoine model

4.3.3 Analysis of the Two Definitions

There are some essential differences between the two definitions. Specifically the ability to choose the attack instance in the Avoine model is missing in the Juels and Weis model. As we already have the OSK protocol [74] proven secure in the Avoine model and proven insecure in the Juels and Weis model. This shows some of the inconsistences between them. We further construct an example protocol that is insecure in the Avoine model but is secure in the Juels and Weis model to complete the inconsistency analysis. Figure 4.1 presents the protocol.

4.3.4 Comparison

In the example protocol, we have a PRNG that takes in the tag key K_{ID} as the seed and k is the $k-th$ output of the PRNG. The outputs are pseudo-random. At the system setup time, every tag is assigned a different random key K_{ID} that serves as the unique random seed per tag. These random seeds are also stored in the database server in a table, which has two columns: the true tag ID and the tag key.

Now if we have a system parameter l, which represents the number of bits of the output of the PRNG. Then we should set $q = 2^l$ (the maximum number of queries allowed in the experiment). Otherwise the PRNG would have exhausted all of its possible outputs when being queried the $2^l + 1 - th$ time and cycled back to its initial output. Such repeated values allows the adversary to trace the same tag. i.e. the tags are untraceable if the total number of queries is less than or equals to q.

However, the above is true only if the attacks are carried out consecutively. If the adversary is able to specify her attack instances, then the tags in this protocol will become traceable by the following attack sequences even with q-bounded queries:

Claim: Protocol in figure 4.1 is insecure in the Avoine model but secure in the Juels and Weis model.

Proof: Avoine model

1. $\mathcal{A}$ requests the Challenger to receive her target $\mathcal{T}$.
2. $\mathcal{A}$ chooses instance 1 and runs the protocol on the target tag $\mathcal{T}$ and receives $\{C_1\}$.
3. $\mathcal{A}$ requests the Challenger to receive her challenges $\mathcal{T}_1$ and $\mathcal{T}_2$.
4. $\mathcal{A}$ chooses instance $2^l + 1$ and runs the protocol on tag $\mathcal{T}_1$ and receives $\{\hat{C}_{2^l+1}\}$.
5. $\mathcal{A}$ checks if $\hat{C}_{2^l+1} = C_1$, if true, then outputs $\mathcal{T}_1$ as her guess; else outputs $\mathcal{T}_2$.

As it is shown that it only requires 2 queries (which is $\ll q$) for $\mathcal{A}$ to trace the same tag. $\mathcal{A}$ only fails if there is a collision in the PRNG (i.e. $PRNG_1(K_{ID_1}) = PRNG_{2^l+1}(K_{ID_2})$, where K_{ID_1} and K_{ID_2} are the random seeds assigned to $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively), which is negligible. Hence $\mathcal{A}$ can success with high probability. □

In fact, it makes sense to allow an adversary to request for specific attack instances when launching her attacks. In reality, even though an adversary is not attacking, those tags and reader are constantly communicating to each other. This changes the internal states of them and may benefits the adversary. In some applications, the number of tag scans is quite predictable (e.g. an RFID-enabled door pass used at an entrance gate). If the protocol has a shorter cycle or the random function has some defects that repeats a pattern once a while, then this can be exploited by an adversary even she is not monitoring her target continuously.

Proof: Juels and Weis model

The adversary is bounded by q as the maximum number of oracles accesses it can make. The repeating cycle for the PRNG in the example protocol is $2^l = q$. By definition, a PRNG is pseudo-random if it is hard to distinguish from a random permutation drawn from it and a random permutation drawn from $\{0, 1\}^l$. Hence within the bounded number of oracle accesses, the PRNG performs as a random number generator and the adversary sees no difference from C_i and a random reply. □

The analysis on the Avoine model and the Juels and Weis model tells us that a unified RFID formalisation is needed to obtain consistent results, which leads to our next step: to look into an even stronger RFID privacy model by Serge Vaudenay.

4.4 A Strong Privacy Model for RFID by Vaudenay

We presented here an even stronger privacy model for RFID. This gives new meaning to what it means by a secure protocol. Clearly from the example protocol in the previous section, PRNGs with known cycle will not be secure under our new definition, which gives a strong message to RFID protocol designers to abandon the use of PRNGs in their protocols. This leave us with random oracle (hash functions) and encryption (both symmetric and asymmetric) schemes as our tools to secure RFID protocols. It is still an open question that whether untraceability can be fulfilled with hash functions only. Because of the limited computation power in RFID tags, public key cryptography is unlikely to fit in and made practical. Serge Vaudenay proposed a new strong privacy model for RFID in [93] with eight classes of privacy levels. He concluded his paper by showing that strong privacy in RFID is impossible. Furthermore, an open questions whether forward privacy without requiring public key cryptography (PKC) is achievable or not was raised. We first summarize the Vaudenay model here, in particular the terms that will be used frequently in the following sections. Readers should refer to [93] for the complete definition and more information.

4.4.1 System Model

An RFID system is defined by the composition of two setup algorithms and the actual RFID protocol.

- $\texttt{SetupReader}(1^s)$ is used to generate the required system parameters K_P and K_S by supplying a security parameter s. K_P denotes all the public parameters available to the environment and K_S denotes the private parameters stored inside the reader and will never be revealed to the adversary.

- $\texttt{SetupTag}^b_{K_P}(ID)$ [1] is used to generate necessary tag secrets K_{ID} and S_{ID} by inputting K_P and a custom unique ID. K_{ID} denotes the key stored inside the tag, rewritable when needed according to the protocol. S_{ID} denotes the memory states pre-set to the tag, updatable during the protocol. A bit b is also specified to indicate this newly setup tag is legitimate or not. An entry of the pair (ID, K_{ID}) will be added into the database of the reader to register this new tag when $b = 1$. Otherwise, if $b = 0$, the reader will not recognize this tag as a legitimate tag and

[1] this b notation was not explicitly specified originally in [93] for this algorithm, we see the need to add it to make the description more precise.

no entry is added. Notice that K_{ID} and S_{ID} are not public and are not available to the adversary unless the tag is corrupted.

- the actual protocol used to identify/authenticate tags with the reader.

4.4.2 Adversary Model

The following eight oracles are defined to represent the abilities of the adversary. We may remove and omit some details in some of the defined oracles but their main functionalities are still maintained.

- $\texttt{CreateTag}^b(ID)$ allows the creation of a free tag. The tag is further prepared by $\texttt{SetupTag}^b_{K_P}(ID)$ with b and ID passed along as inputs.
- $\texttt{DrawTag}()$ returns an ad-hoc handle $vtag$ (unique and never repeats) for one of the free tags (picked randomly). The handle can be used to refer to this same tag in any further oracles accesses until it is erased. A bit b is also returned to indicate whether the referencing tag is legitimate or not.
- $\texttt{Free}(vtag)$ simply marks the handle $vtag$ unavailable such that no further references to it are valid.
- $\texttt{Launch}()$ starts a protocol instance at the reader side and a handle π (unique and never repeats) of this instance is returned.
- $\texttt{SendReader}(m, \pi)$ sends a message m to the reader for a specific instance determined by the handle π. A reply message m' from the reader may be returned depending on the protocol.
- $\texttt{SendTag}(m, vtag)$ sends a message m to the tag determined by the handle $vtag$. A reply message m' from this tag may be returned depending on the protocol.
- $\texttt{Result}(\pi)$ returns either 1 if the protocol instance π being queried completed with success (i.e. the protocol identifies a legitimate tag) or 0 otherwise.
- $\texttt{Corrupt}(vtag)$ returns all the internal secrets K_{vtag} [2] and S_{vtag} of the tag determined by the handle $vtag$.

[2] Originally in [93], K_{vtag} was not included in the description. They assume that K_{vtag} is always extractable from S_{vtag}. We add K_{vtag} here to make the description clearer.

The interface (the environment) that provides the access to these oracles for the adversary also maintains a hidden table $\mathcal{T}$, which is not available to the adversary until the last step of the privacy experiment (to be reviewed below). When `DrawTag`() is called, a new entry of the pair $(vtag, ID)$ is added into $\mathcal{T}$. When `Free`$(vtag)$ is called, the entry with the same $vtag$ handle will be marked unavailable. The true ID of the tag with handle $vtag$ is represented by $\mathcal{T}(vtag)$.

4.4.3 Privacy Experiment

The privacy experiment that runs on an RFID protocol is defined as a game to see whether the adversary outputs *True* or *False* after seeing the hidden table $\mathcal{T}$. At the beginning, the adversary is free to access any oracles within his allowed oracles collection (which defines different classes of adversary) according to his own attack strategy. Once the adversary finishes querying, the hidden table $\mathcal{T}$ will be released to him. The adversary will then analyze the table using the information obtained from the queries. If the adversary outputs *True*, then he wins the privacy experiment.

To measure the privacy level of an RFID protocol, a *blinder* is constructed to simulate `Launch`(), `SendReader`(m, π), `SendTag`$(m, vtag)$ and `Result`(π). If the adversary can still win with a similar probability in the above experiment even in the present of a blinder (hence the simulations do not affect the winning probability too much), then his attack strategy is considered to be trivial. i.e. either the simulations are perfect or the attack strategy does not exploit the simulated oracles. If for all the possible attack strategies from this adversary, we can construct a blinder (possibly different) for each of them such that they are all trivial attacks, then the RFID protocol being experimented is called P-private where P is the privacy class. Let $\mathcal{A}$ be the adversary and $\mathcal{A}^{\mathcal{B}}$ be the same adversary blinded by the blinder $\mathcal{B}$, then $|\Pr[\mathcal{A} \text{ wins}] - \Pr[\mathcal{A}^{\mathcal{B}} \text{ wins}]| = \epsilon$ can be used to express the above measurement where ϵ is a negligible value.

4.4.4 Privacy Classes

The eight privacy classes are distinguished by different oracles collections and different natures on accessing `Corrupt`$(vtag)$ according to the strategies of the adversary.

- *Weak* : A basic privacy class where access to all the oracles are allowed except `Corrupt`$(vtag)$.
- *Forward* : It is less restrictive than *Weak* where access to `Corrupt`$(vtag)$ is allowed under the condition that when it is accessed the first time, no other

$$\begin{array}{ccccccc} \textit{Strong} & \Rightarrow & \textit{Destructive} & \Rightarrow & \textit{Forward} & \Rightarrow & \textit{Weak} \\ \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow \\ \textit{Narrow-Strong} & \Rightarrow & \textit{Narrow-Dest.} & \Rightarrow & \textit{Narrow-Forward} & \Rightarrow & \textit{Narrow-Weak} \end{array}$$

Figure 4.2: Relationship of the 8 privacy classes in the Vaudenay model

types of oracle can be accessed subsequently except more $\texttt{Corrupt}(vtag)$ (can be on different handles).

- *Destructive* : It further relaxes the limitation on the adversary's strategies compares to *Forward* where there is no restriction on accessing other types of oracle after $\texttt{Corrupt}(vtag)$ under the condition that whenever $\texttt{Corrupt}(vtag)$ is accessed, such handle $vtag$ cannot be used again (i.e. virtually destroyed the tag).

- *Strong* : It is even more unrestrictive than *Destructive* where the condition for accessing $\texttt{Corrupt}(vtag)$ is removed. It is the strongest defined privacy class in Vaudenay's privacy model.

Each of these privacy classes also has their *Narrow* counterparts. Namely, *Narrow-Strong*, *Narrow-Destructive*, *Narrow-Forward* and *Narrow-Weak*. These classes share the same definitions of their counterparts only there is no access to $\texttt{Result}(\pi)$.

By relaxing the limitation on the adversary's attack strategies from *Weak* to *Strong*, the adversary becomes more powerful. One can see that the privacy level is increasing from *Weak* to *Strong* if the protocol is secure against the respective class of adversary. Hence, for an RFID protocol to be *Strong*-private, it must also be *Destructive*-private. Likewise, to be *Destructive*-private, it must also be *Forward*-private, and so on. And then for a P-private protocol, it must also be *Narrow-P*-private since the *Narrow* counterparts are more restrictive. With all these implications, the relations between the eight privacy classes are depicted in figure 4.2.

4.5 Our Refined Vaudenay Privacy Model

Having observed the classification of privacy presented in [93], we show that these eight privacy classes can be reduced to three privacy classes under appropriate assumptions. Hence, we provide a simplified privacy classification in RFID but by no means it is being crippled, rather, it is more handy to use. Based on our simplified classification,

we show that the strongest privacy level is indeed achievable, in contrast to the result presented in [93]. This is a positive result that supports the use of RFID in practice. We also answer the open question in [93] by pointing out the possibility to achieve forward privacy without PKC, both within the formal model and in practice.

4.5.1 Preliminaries

The following basic assumptions will be used in the following parts. We note that these assumptions have been used in the existing works as well, and hence, they are not unreasonable. We consider an RFID system with one reader and many tags, in which their relationship is always many to one, i.e. all the legitimate tags are identifiable by a legitimate reader and only this legitimate reader can identify them. During an execution of the RFID protocol, only a single tag will be involved in the communication with the reader in each instance, i.e. an appropriate and secure singulation protocol is always assumed. The reader may need a back-end database server to help with the identification process. The link that connects the reader and the back-end database server is assumed to be secure and always reliable and available. The reader can retrieve necessary data whenever required. Hence, we consider the reader has online connection to the server and together they can be regarded as a single entity. The back-end server is secure and cannot be attacked. Furthermore, due to this requirement, the reader is not corruptible and all the data stored in reader side are secure. Only the wireless link established between the reader and the involving tag during a protocol instance is insecure. Tags are not tamper-proofed. All the internal secrets stored, the memory contents written and the algorithms defined are assumed to be readily available to the adversary when a tag is corrupted. The reader will always initiate the protocol by sending out the first query message (may contain a challenge) as the tags are passive.

Notice that whenever we mention the data inside the reader, we do not necessarily mean they are stored in the memory of the reader. Rather, they may actually retrieved from the database server in reality. Data management is also another advanced research topic in RFID researches to solve access rights problem, provide support of multiple and off-line readers, etc., in particular for the supply chain environment like [61]. Although there is such a separation, it does not affect our discussion as we merely focus on the communication between tags and reader.

4.5.2 New Privacy Classification

In this section, we firstly comment on the privacy model defined in [93]. In particular, we comment that the separation of eight privacy classes is rather excessive and unnecessary for most of the RFID protocols under proper assumptions. Then, we provide our simplified privacy model that will merge some of the privacy classes into a single class. The main aim of this section is to prove the following proposition:

Proposition 1. *For protocols without correlated keys and do not produce false-negative results, the eight privacy classes can be reduced to three major privacy classes if the adversary only makes "wise" oracle access.*

The "no false-negative" assumption that we will incorporate also appear in Lemma 8 of [93] where narrow-forward and narrow-weak privacy classes are reduced to forward and weak privacy classes respectively (i.e. from eight classes to six classes). The lemma assumes that any legitimate tag will *always* be identifiable, which means *no* false-negative is possible. Hence, accessing the $\texttt{Result}(\pi)$ oracle is not significant as 1 will always be returned. As a result, the separation between *Forward* (*Weak*) and *Narrow-Forward* (*Narrow-Weak*) becomes unnecessary. We further extend this to the strong and destructive classes and consider also the false-positive case in the following proposition.

Proposition 2. *If the privacy model considers only RFID protocols that are correct and no false-negative is possible and we assume that the adversary $\mathcal{A}$ only makes "wise" oracle access whenever $\mathcal{A}$ has a non-trivial attack strategy, then the separation between narrow and non-narrow classes is unnecessary.*

The idea of proposition 2 is that if we can be sure and verify that the RFID protocol being examined will never give out false-negative, then we can examine the protocol only according to the definition of the privacy classes *Strong*, *Destructive*, *Forward* and *Weak* by assuming a "wise" adversary. This means that whether the $\texttt{Result}(\pi)$ oracle is accessed or not, it does not affect the privacy experiment results. We can remove the necessity of this oracle and reduce the eight privacy classes into four privacy classes.

Before proofing the proposition, we have to define what is "wise" oracle access and redefine what are trivial and non-trivial attacks. We also introduce perfect blinders and partial blinders.

Wise adversary

An adversary $\mathcal{A}$ who is "wise" on oracle access will not make any oracle access that is redundant, or in other words, brings no advantage to him in attacking privacy of the protocol. We argue that this definition can be defined naturally as although the oracle access is available to $\mathcal{A}$, if the access will not provide any help to $\mathcal{A}$ during the attack, then naturally $\mathcal{A}$ will not access it anyway. Simply speaking, $\mathcal{A}$ will not waste any oracle access. More formally, let S and S' denote two different attack strategies of $\mathcal{A}$ in the privacy experiment for the same privacy class. Let q and q' be the total number of oracle accesses after executing S and S' respectively. S defines a "wiser" oracle access strategy compares to S' if and only if $\Pr[\mathcal{A}_S \text{ wins}] = \Pr[\mathcal{A}_{S'} \text{ wins}]$ and $q < q'$. Overall, a "wise" adversary can be generally defined such that for all his attack strategies, the total numbers of oracle accesses are always minimal. Of course, such general definition of "wise" is not specific enough because q is not known before the end of attack. Specific rules are needed to keep q minimal. Consider the following as the special properties of our "wise" adversary:

- No access to the same oracle (if not probabilistic) with the same input twice.
- No access to oracles where the results can be precisely predicted.

Property 2 may be too general and should receive more justification. However, to serve our purpose in reducing the privacy classes, it is enough to focus on the $\texttt{Result}(\pi)$ oracle only, i.e. if a certain result is expected, the "wise" adversary will not access the $\texttt{Result}(\pi)$ oracle. Indeed, if the RFID protocol is *Correct*, then any legitimate or non-legitimate tag should be identified correctly, i.e. if the protocol instance π was completed for a legitimate tag, then $\texttt{Result}(\pi)$ should return 1; otherwise, 0 should be returned if it was a non-legitimate tag. This should be true as long as there are no adversarial attacks or the attacks are *insignificant*. We say that an attack is *significant* if and only if it causes the $\texttt{Result}(\pi)$ oracle to return an opposite result. This means that if there is a significant attack on a legitimate tag, then $\texttt{Result}(\pi)$ would return 0 instead of 1, and we have a *false-negative*; if there is a significant attack on a non-legitimate tag, then $\texttt{Result}(\pi)$ would return 1 instead of 0, and we have a *false-positive*. Notice that we do not need to consider incorrect identification here where a legitimate tag with ID a is identified as ID b because the $\texttt{Result}(\pi)$ oracle will only return 1 either way, making it indistinguishable by looking at the returned value only. After all, impersonation is not the goal of the privacy adversary.

Redefining trivial and non-trivial attacks

By definition, if there is a blinder $\mathcal{B}$ such that $|\Pr[\mathcal{A} \text{ wins}] - \Pr[\mathcal{A}^{\mathcal{B}} \text{ wins}]| = \epsilon$ where ϵ is a negligible value, then we say that the attack by $\mathcal{A}$ is trivial, otherwise if the value is non-negligible then the attack is non-trivial. It naturally follows that we can express this difference in the success probability of $\mathcal{A}$ under normal oracle access and simulated oracle access as the potential advantage loss of $\mathcal{A}$ because $\mathcal{A}$ has a different failure probability during the interactions with simulated oracles due to abortion of the blinder. We define this disadvantage as $\mathcal{D}^{\mathcal{B}}{}_{abort} = |\Pr[\mathcal{A} \text{ wins}] - \Pr[\mathcal{A}^{\mathcal{B}} \text{ wins}]| = |(1 - \Pr[\mathcal{A} \text{ fails}]) - (1 - \Pr[\mathcal{A}^{\mathcal{B}} \text{ fails}])| = |\Pr[\mathcal{A} \text{ fails}] - \Pr[\mathcal{A}^{\mathcal{B}} \text{ fails}]|$. Hence, $\mathcal{D}^{\mathcal{B}}{}_{abort}$ is the difference in the probability that $\mathcal{A}$ will fail after the introduction of $\mathcal{B}$ and if $\mathcal{D}^{\mathcal{B}}{}_{abort} = \epsilon$, then the attack by $\mathcal{A}$ is trivial; otherwise if $\mathcal{D}^{\mathcal{B}}{}_{abort} = \theta$ where θ is some non-negligible value, then the attack by $\mathcal{A}$ is non-trivial.

Perfect blinder

A perfect blinder $\bar{\mathcal{B}}$ is a blinder that can simulate all the four blinded oracles (`Launch`(), `SendReader`(m, π), `SendTag`$(m, vtag)$ and `Result`(π)) perfectly such that $\mathcal{D}^{\bar{\mathcal{B}}}{}_{abort} = \epsilon$.

Partial blinder

Similarly, a partial blinder $\dot{\mathcal{B}}$ is a blinder that has at least one of the four blinded oracles where the simulation is not perfect. i.e. $\dot{\mathcal{B}}$ will have a chance to abort if an imperfect simulated oracle is being accessed. Notice that we may or may not end up with $\mathcal{D}^{\dot{\mathcal{B}}}{}_{abort} = \theta$ because $\mathcal{A}$ may or may not have effectively exploited the imperfect simulated oracle(s), it depends on the attack strategy of $\mathcal{A}$.

We have the following lemma that changes a partial blinder to a perfect blinder.

Lemma 1. *A partial blinder can be viewed as a perfect blinder if and only if the adversary does not effectively exploit the imperfect simulated oracle(s).*

Proof. Let $\dot{\mathcal{B}}$ be the partial blinder where at least one of the four simulated oracles is imperfect. Let $\mathcal{O}$ denote the set of simulated oracles, then we have $\mathcal{O}_p$ be the set of perfect simulated oracles and $\mathcal{O}_p^c$ be the set of imperfect simulated oracles. $\mathcal{O}_p \cup \mathcal{O}_p^c = \mathcal{O}$ and $\mathcal{O}_p \cap \mathcal{O}_p^c = \emptyset$. Let $\mathcal{O}'$ be the set of non-simulated oracles and let E^* be the event that an abortion happens in oracle $*$. (if part) It is easy to justify that $\Pr[\mathcal{A}^{\dot{\mathcal{B}}} \text{ fails}] = \Pr[E^{\mathcal{O}}] + \Pr[E^{\mathcal{O}'}] = \Pr[E^{\mathcal{O}_p \cup \mathcal{O}_p^c}] + \Pr[E^{\mathcal{O}'}] = \Pr[E^{\mathcal{O}_p}] + \Pr[E^{\mathcal{O}_p^c}] + \Pr[E^{\mathcal{O}'}]$. Since

the adversary does not effectively exploit the imperfect simulated oracle, which means $\Pr[E^{\mathcal{O}_p^c}]$ is negligible. Note we also have $\Pr[\mathcal{A}^{\bar{\mathcal{B}}}$ fails$] = \Pr[E^{\mathcal{O}_p}] + \Pr[E^{\mathcal{O}'}]$, which is basically $\Pr[\mathcal{A}^{\dot{\mathcal{B}}}$ fails$]$ - $\Pr[E^{\mathcal{O}_p^c}]$. i.e. $|\Pr[\mathcal{A}^{\bar{\mathcal{B}}}$ fails$]$ - $\Pr[\mathcal{A}^{\dot{\mathcal{B}}}$ fails$]| = \epsilon$. (only if part) Suppose the adversary did effectively exploit the imperfect simulated oracle, then we have $\mathcal{D}^{\dot{\mathcal{B}}}{}_{abort} = \theta$, which can not be a perfect blinder for $\mathcal{D}^{\bar{\mathcal{B}}}{}_{abort} = \epsilon$ □

Corollary, we can divide the following similar lemma that changes a partial blinder of one privacy class to a perfect blinder of another privacy class using a similar proof.

Lemma 2. *A partial blinder of a stronger privacy class can be viewed as a perfect blinder of a weaker privacy class if and only if the imperfect simulated oracle is not available in the weaker privacy class.*

We do not repeat the proof here as it is very similar to the pervious proof. Clearly, not using effectively is an analogue to not available. These lemmas are general, which applies to any oracles and privacy classes. But since our goal is to show the relation between *Narrow* and *Non-narrow* classes where the `Result`(π) oracle is available only to non-narrow classes, without loss of generality we will specifically use the `Result`(π) oracle as an example in the following proof. Let $\mathcal{A}$ be the adversary who attacks any *Non-narrow* privacy classes and $\mathcal{A}_N$ be the same adversary who attacks the *Narrow* counterpart of the corresponding privacy classes. Our goal is to show that $|\Pr[\mathcal{A}$ wins$]$ - $\Pr[\mathcal{A}_N$ wins$]| = \epsilon$ in any situations provided that there is no false-negative and the adversaries are "wise". We are now ready to prove the proposition.

Proof. The significance of calling the `Result`(π) oracle is when there will be an opposite output, i.e. getting 1 when it supposes to be 0 or vice versa. This means that at least some of the attack sequences in the attack strategy have significant effect to the protocol, which makes the reader misidentify a legitimate tag as a non-legitimate one (false-negative) or a non-legitimate one as a legitimate one (false-positive). Otherwise, it would not be "wise" for the adversary to access `Result`(π) if he did not execute any significant attacks since either 1 or 0 will be the guaranteed output for legitimate or non-legitimate tag. Indeed, the adversary always knows this fact (whether a tag is legitimate or not) when he calls `DrawTag`() to obtain a handle to a tag where a bit b is also provided to indicate the legitimacy of that tag. According to the behaviour of the blinder $\mathcal{B}$ in simulating the `Result`(π) oracle, there can be different situations as described in table 4.3:

	True oracle	Perfect simulation	Imperfect simulation
Legitimate	1	$(vtag, 1) \leftarrow \texttt{DrawTag}()$	$(vtag, 1) \leftarrow \texttt{DrawTag}()$
Non-legitimate	0	$(vtag, 0) \leftarrow \texttt{DrawTag}()$	$(vtag, 0) \leftarrow \texttt{DrawTag}()$
False-negative	N/A	N/A	N/A
False-positive	1	$1 \leftarrow \texttt{Result}^{\mathcal{B}}(\pi)$	unknown

Table 4.3: Simulation results of the $\texttt{Result}(\pi)$ oracle

Since we have the hypothesis that there is no false-negative, we do not need to consider it in the proof. We now have four cases to consider: i) when the attack is trivial, ii) when the attack is non-trivial and there is/are imperfect simulated oracle(s) other than $\texttt{Result}^{\mathcal{B}}(\pi)$, iii) when $\texttt{Result}^{\mathcal{B}}(\pi)$ is the only imperfect simulated oracle but $\mathcal{A}$ does not make effective use of it, and iv) when $\texttt{Result}^{\mathcal{B}}(\pi)$ is the only imperfect simulated oracle and $\mathcal{A}$ exploited it effectively.

(Case i) Consider when the attack strategies of $\mathcal{A}$ are all trivial. Then, by definition, the four oracles $\texttt{Launch}()$, $\texttt{SendReader}(m, \pi)$, $\texttt{SendTag}(m, vtag)$ and $\texttt{Result}(\pi)$ must be simulated successfully without non-negligibly affecting the success probability of the blinded $\mathcal{A}$. Since the simulation is perfect, $\mathcal{A}$ should expect no advantage gained by accessing any one of these blinded oracles in compare to when they are not blinded, i.e. we always have a perfect blinder $\bar{\mathcal{B}}$ such that $\mathcal{D}^{\bar{\mathcal{B}}}{}_{abort} = \epsilon$ where ϵ is some negligible value. Hence the RFID protocol is secure in the an non-narrow class. As the narrow counterpart is a subset of the non-narrow class, the protocol is also secure in the corresponding narrow class. As a result, protocols are both secure in narrow and non-narrow classes if the adversary's attacks are all trivial, which makes the separation unnecessary.

(Case ii) We consider when $\mathcal{A}$ has a non-trivial attack strategy. This means that there is at least one of the four blinded oracles that failed to simulate the real oracle perfectly. Suppose that it is not the $\texttt{Result}^{\mathcal{B}}(\pi)$ oracle which is/are imperfect or if $\texttt{Result}^{\mathcal{B}}(\pi)$ is imperfect, there is/are other imperfect blinded oracle(s). Since the imperfect blinded oracle(s) other than $\texttt{Result}^{\mathcal{B}}(\pi)$ is/are available to both the narrow and non-narrow classes, which means $\mathcal{A}$ can always launch non-trivial attacks through them, i.e. the RFID protocol is not secure in both classes anyway, hence the separation is unnecessary.

(Case iii) Suppose that it is now only the $\texttt{Result}^{\mathcal{B}}(\pi)$ oracle which is imperfect. Then, we have a partial blinder $\dot{\mathcal{B}}$. Assume that $\mathcal{A}$ did not make effective use of $\texttt{Result}^{\mathcal{B}}(\pi)$

during his attack; then by lemma 1, the partial blinder $\dot{\mathcal{B}}$ of the non-narrow classes can be viewed as a perfect blinder $\bar{\mathcal{B}}$ for the same privacy classes. Also by lemma 2, $\dot{\mathcal{B}}$ of the non-narrow classes is also a perfect blinder of the narrow classes since $\texttt{Result}^{\mathcal{B}}(\pi)$ is not available in the narrow classes. Since the blinder is perfect in both classes, $\mathcal{A}$'s attacks can only be trivial and the RFID protocol is secure in both non-narrow and narrow classes. Hence, even if $\texttt{Result}^{\mathcal{B}}(\pi)$ can not be simulated perfectly, there is no difference in the privacy experiments for both classes if the imperfect $\texttt{Result}^{\mathcal{B}}(\pi)$ is not exploited effectively.

(Case iv) Now for $\mathcal{A}$ to exploit the imperfect $\texttt{Result}^{\mathcal{B}}(\pi)$ effectively, $\mathcal{A}$ must cause an opposite output to happen when accessing $\texttt{Result}^{\mathcal{B}}(\pi)$. Since false-negative is not possible as it is the hypothesis, we only need to look at false-positive, i.e. getting 1 instead of 0. False-positive happens when a non-legitimate tag is wrongly identified by the reader as a legitimate tag. Let us denote this event as E. Assume that $\mathcal{A}$ is "wise" enough not to waste any oracle accesses. When E occurs, $\mathcal{A}$ must have done some significant attacks to a non-legitimate tag or else the protocol is simply incorrect. In order to attack the tag, $\mathcal{A}$ must have obtained a handle $vtag$ to this tag, which means $\mathcal{A}$ must have called the $\texttt{DrawTag}()$ oracle. Recall that $\texttt{DrawTag}()$ returns $vtag$ and a bit b indicating whether $vtag$ is legitimate or not. Since $vtag$ is non-legitimate, we have $b = 0$. Recall that $\texttt{DrawTag}()$ is not simulated by the blinder $\mathcal{B}$, $\mathcal{B}$ can also observe the returned pair $(vtag, 0)$ when $\texttt{DrawTag}()$ is accessed by $\mathcal{A}$, hence $\mathcal{B}$ must also know $vtag$ is a non-legitimate tag. Since $\mathcal{B}$ does not know K_S, $\mathcal{B}$ has no way to tell if the reader will accept $vtag$ or not for $\mathcal{A}$ may have attacked $vtag$ at any moment, hence $\mathcal{B}$ may not be able to output the same value as the real $\texttt{Result}(\pi)$ oracle. $\mathcal{B}$ can only hope that whenever $\mathcal{A}$ accesses the $\texttt{Result}^{\mathcal{B}}(\pi)$ oracle, $\mathcal{A}$ must have already attacked $vtag$ successfully, hence $\mathcal{B}$ can be constructed to simulate $\texttt{Result}^{\mathcal{B}}(\pi)$ by returning 1 if π is the protocol instance with $vtag$ where $(vtag, 0)$ is observed when $\texttt{DrawTag}()$ is accessed. The simulation is perfect as long as $\mathcal{A}$ performs significant attacks to $vtag$, which causes the results change from 0 to 1. The simulation will fail when $\mathcal{A}$ makes the $\texttt{Result}^{\mathcal{B}}(\pi)$ query for the protocol instance where $vtag$ is not being attacked. In that case, $\mathcal{B}$ should return 0 instead of 1. However, this should not happen because this contradicts the second property of the "wise" $\mathcal{A}$ who will not waste any oracle accesses as he knows that the reader must be able to identify a non-legitimate tag (i.e. returning 0) if it has not been attacked. Hence $\mathcal{A}$ would not have called $\texttt{Result}^{\mathcal{B}}(\pi)$ for the protocol instance with $vtag$ when $\mathcal{A}$ did not perform any significant attacks

to *vtag*. At the end, $\mathcal{B}$ can simulate the oracles perfectly in front of the "wise" $\mathcal{A}$ and hence $\mathcal{D}^{\bar{\mathcal{B}}}{}_{abort} = \epsilon$, making $\mathcal{A}$'s strategy trivial, which contradicts that $\mathcal{A}$ has a non-trivial attack strategy. Hence $\mathcal{A}$ would not have let E occur, which becomes case iii. □

This proof shows that the $\texttt{Result}(\pi)$ oracle will never help the adversary if the RFID protocol being examined renders no false-negative. Furthermore, the adversary should not waste time on causing a false-positive since the attack should be on privacy and not on impersonation nor unauthorised access. In other words, from all the possible attack strategies of $\mathcal{A}$, there will be no $\texttt{Result}(\pi)$ queries if the RFID protocol being attacked does not give out false-negative. One can also extend proposition 2 to include RFID protocols where false-negative occurs with negligible if not zero probability with the same proof. Now, we have obtained the result that a P-private adversary's strategy performs as best as a *Narrow-P*-private adversary's strategy under proposition 2. Hence, we have reduced eight classes to four classes, as follows.

$$\textit{Strong} \quad \Rightarrow \quad \textit{Destructive} \quad \Rightarrow \quad \textit{Forward} \quad \Rightarrow \quad \textit{Weak}$$

Next, we analyse the usefulness of the destructive class. In fact, it is also mentioned in [93] that the purpose of separating the strong and destructive classes is unclear. The destructive class is a rarely happen privacy level. Perhaps, this is the reason why there is no example provided, which is secure for this class in [93][3]. Therefore, we come up with the following proposition.

Proposition 3. *If the privacy model considers only RFID protocols that use no correlated keys among tags, then it is unnecessary to consider the destructive classes (both narrow and non-narrow).*

In other words, the destructive class is only useful to examine RFID protocols where the tags share some correlated secrets. Such type of protocols is not common in RFID. We only came across three constructions in [8, 21, 69]. The motivation behind these protocols by providing correlated key protocols is to reduce the workload and time required to lookup a matching key to verify the tag in the reader side. In most of the proposed RFID protocols under symmetric key settings [10, 16, 24, 38, 54, 74], it is unavoidably to engage in an exhaustive key search process in the reader side in order

[3]Notice that the example provided in [93] for the narrow-destructive class that use independent keys is no different from a protocol for the narrow-forward class, while the example given that uses dependent keys is insecure in the narrow-destructive class.

to compute and match the response of any tag from all the possible keys stored inside the database. Attempts to solve this problem by providing some means to keep tags and the reader synchronized on the next expected key to be used [37, 41, 85, 86] are found to have security loopholes [6, 49]. Furthermore, Juels exploited this to attack various protocols by constructing side-channel attacks thank to the obvious different key lookup time detectable from each protocol session in [49]. A recent attempt to provide a constant lookup time [15] turns out to use a one-way trapdoor function, which is considered as one of the public key settings. Hence, there is still no efficient protocol known to solve this issue under symmetric key settings.

Additionally, correlated keys protocol under symmetric key settings can reduce the number of keys search to a logarithmic scale but with a sacrifice on strong privacy [69]: any corruption of the tag will degrade the privacy level because a tag stores not only its secret keys but also keys that share with other tags. One typical example of a correlated key protocol can be constructed using $\log_2 n$ keys for n tags. Suppose there are 8 tags in the system. One can generate only 6 keys, namely: $K_0^a, K_0^b, K_0^c, K_1^a, K_1^b, K_1^c$. Each tag is equipped with a unique set of keys, i.e. $Tag_1 \leftarrow \{K_0^a, K_0^b, K_0^c\}$, $Tag_2 \leftarrow \{K_0^a, K_0^b, K_1^c\}$, ... $Tag_n \leftarrow \{K_1^a, K_1^b, K_1^c\}$. It is easy to verify that these tags can be uniquely identified by checking at most 6 instead of 8 keys as in the independent key protocols by the reader. However, corrupting any one of these tags provides the adversary with a full potential to distinguish each of these tags responses. Damgård [21] provided a result on the tradeoff between the number of correlated keys and the number of corrupted tags as:

$$\frac{ctu}{v} + \frac{ctu}{v-u}$$

where c is the number of keys stored in each tag, v is the number of different keys per column, t is the number of tags queried by the adversary and u is the number of corrupted tag. As long as the result of the formula is negligible, the protocol is secure. In our example given above, $c = 3, v = 2$ and $u = 1$, hence we have $\frac{3t}{2} + 3t \leq \epsilon$, which means $t < 1$, i.e. the protocol is secure only if the adversary does not query any tag at all, or simply the protocol is never secure against tag corruptions.

From the above discussion, it is clear that correlated key protocols are extremely weak against tag corruption. One can only expect the protocol to be secure if $t, u << v << n << v^c$. In the model of [93], since there is no limitation on either t or u, there can be no correlated key protocols that is secure in both strong and destructive privacy classes. The proof of proposition 3 follows.

Proof. Recall the destructive class definition, after calling `Corrupt`($vtag$), the same tag handle $vtag$ is not allowed to be used anymore. It is clear to see from the definition that this destructive corruption cannot provide the adversary any additional advantage in winning the privacy game if each of the tags is independent to each other. In order for the `Corrupt`($vtag$) oracle to become significant under the destructive class definition, the corrupted internal secrets K_{vtag} and S_{vtag} have to be useful in some following oracle accesses (if it is useful to the results obtained from some pervious oracles, then we have gone backward to the forward class). Since the corrupted tag of handle $vtag$ cannot be accessed again, the secrets must only be used on some other tags. If the tags are independent to each other, K_{vtag} and S_{vtag} would have revealed no information about any other tags. As there is no effect on other tags, the simulation of the blinder can be easily constructed, making the adversary's strategy trivial and hence the attack is insignificant. □

Combining the above results, the destructive class is rather not very meaningful. It is only useful to examine protocols that use correlated keys while these protocols can never achieve strong and destructive privacy classes under the model in [93]. Together with proposition 2, we have successfully reduced the eight privacy classes into three major classes, as follows.

$$\textit{Strong} \quad \Rightarrow \quad \textit{Forward} \quad \Rightarrow \quad \textit{Weak}$$

Our result simplifies the previous privacy classification due to Vaudenay [93]. Furthermore, in contrast to Vaudenay's result, we shall show that strong privacy is indeed possible, and hence this result will indeed make RFID protocols more useful in its real applications.

4.6 Our Different Results From the Vaudenay Model

In this section, we will present our new results in privacy model in RFID. In particular, we shall show that strong privacy is indeed possible (cf. [93]) and we shall present our affirmative answer to the open problem posed in [93] in regards to the construction of RFID scheme with forward privacy without requiring the public key cryptography (PKC).

Tag$\{K_P, ID, K_{ID}\}$		Reader$\{K_S, K_M\}$
	$\xleftarrow{\quad a \quad}$	pick $a \in \{0,1\}^s$ randomly
$c = \texttt{Enc}(K_{ID}\|\|ID\|\|a, K_P)$	$\xrightarrow{\quad c \quad}$	$\texttt{Dec}(c, K_S) = K_{ID}\|\|ID\|\|a'$
		if $a' = a$, verifies $K_{ID} = F_{K_M}(ID)$

Figure 4.3: An example PKC RFID protocol

4.6.1 Strong Privacy is Possible

One of the results in [93] is that strong privacy is impossible. This is supported by a theorem that *a Destructive-private RFID protocol is not Narrow-Strong-private.* Since *Strong*-private (**S**) implies both *Destructive*-private (**D**) and *Narrow-Strong*-private (**NS**) by definition[4], we have $\mathbf{S} \subseteq \mathbf{D}$ and $\mathbf{S} \subseteq \mathbf{NS}$. i.e. $\exists p \in \mathbf{S}$ s.t. $p \in \mathbf{D}$ and $p \in \mathbf{NS}$ where p is an RFID protocol. However, we would like to use our results in Section 3 to show that strong privacy is actually possible. We consider the same example of PKC-based RFID protocol provided in Section 4.3 of [93], which is *Narrow-Strong*-private. By applying proposition 2, we show that it is also *Strong*-private.

We look at the following example PKC protocol where $\texttt{Enc}()$ is IND-CPA secure and (K_P, K_S) is the public and private keys pair. For completeness, we present the protocol in figure 4.3.

To apply proposition 2, we have to observe whether false-negative could be generated. Since c is the only message received by the reader, a false-negative can only happen if c is malicious (i.e. ID and K_{ID} are replaced), or c happens to be the same encrypted value c' where $c' = \texttt{Enc}(ID'||K_{ID'}||a, K_P)$. The former is safe guarded by the IND-CPA secure property of the PKC algorithm, which states that it is infeasible for any computationally bounded adversary to retrieve the private key by looking at the ciphertexts of arbitrarily chosen plaintexts only. That means, the only possible option is to guess the private key, which happens with negligible probability. The latter will not happen as decryption is unique, otherwise both c and c' will be decrypted to a same value. Therefore, we can apply proposition 2 and the PKC protocol is also *Strong*-private if it is *Narrow-Strong*-private. This gives us the result $\mathbf{S} = \mathbf{NS}$. Since $\mathbf{S} \subseteq \mathbf{D}$, we also have $\mathbf{NS} \subseteq \mathbf{D}$. Together with the theorem in [93], we conclude with $\mathbf{NS} \subset \mathbf{D}$.

[4]This is easy to verify. As *Narrow-Strong* is *Strong* without the $\texttt{Result}(\pi)$ oracle access. *Destructive* is *Strong* with additional limitation on accessing the $\texttt{Corrupt}(vtag)$ oracle. Both are more restrictive (i.e. the adversary is less powerful) than *Strong*. A protocol secure in *Strong* must also be secure in *Destructive* and *Narrow-Strong*.

Tag$\{K_P, ID, K_{ID}\}$		Reader$\{K_S, K_M\}$
pick $r \in \{0,1\}^s$ randomly	$\xleftarrow{\quad a \quad}$	pick $a \in \{0,1\}^s$ randomly
$c = \texttt{Enc}(K_{ID}\|\|ID\|\|a\|\|r, K_P)$	$\xrightarrow{\quad c \quad}$	$\texttt{Dec}(c, K_S) = K_{ID}\|\|ID\|\|a'\|\|r$
		if $a' = a$, verifies $K_{ID} = F_{K_M}(ID)$

Figure 4.4: An example PKC RFID protocol providing strong privacy

4.6.2 Truly Random Source is Required

Let us observe the PKC protocol in figure 4.3 again. The protocol assumes that the underlying encryption algorithm is IND-CPA. Due to the randomness of the IND-CPA property, which is needed to provide indistinguishability, c is different every time even if the same a is received by the same tag, i.e. $c = \texttt{Enc}(K_{ID}||ID||a, K_P)$ in protocol instance π is not equal to $\tilde{c} = \texttt{Enc}(K_{ID}||ID||a, K_P)$ in another protocol instance $\tilde{\pi}$. This randomness is implicitly included in the IND-CPA assumption. We can change the notation a little bit to reveal this hidden randomness. We rewrite the PKC protocol in figure 4.4.

In fact, even under the IND-CPA assumption, the tag still needs to pick a random value r for every encryption (e.g. using the ElGamal scheme). This is just abstracted in [93]. Notice that ElGamal is not IND-CCA2 secure. If we further consider a decryption oracle is available, then we cannot instantiate the encryption with ElGamal. Hence we will have to require the PKC example to be IND-CCA2 in this case. With the new notation, we can now consider the following question: *If a tag is corrupted, will the algorithm to generate future random values be revealed as well?* If PRNG is implemented in the tag to generate random values, the answer to this question should be 'yes'. It is easy to see that if the PRNG algorithm is revealed after corrupting the tag, the adversary can easily trace the tag by computing $r = \texttt{PRNG}(S)$ (S is the memory state of the tag) and then verifies that if $c = \texttt{Enc}(K_{ID}||ID||a||r, K_P)$ where K_{ID}, ID, S_{ID}, and $\texttt{PRNG}()$ are all revealed after tag corruption. Since c is unique, the adversary must be able to trace the tag. However, if the tag has a truly random source (e.g. another module attached to the tag), this can be modelled as a random oracle and the answer should be 'no'. We conclude that a truly random source (under the random oracle model) is required for the PKC protocol to be *Strong*-private, which was missing in the definition provided in [93].

Tag$\{K_{ID}\}$		Reader$\{(ID_1, K_1), (ID_2, K_2), ..., (ID_n, K_n)\}$
	$\xleftarrow{\quad a \quad}$	pick $a \in \{0,1\}^s$ randomly
$c = F(K_{ID}, a)$	$\xrightarrow{\quad c \quad}$	for $j \in \{1, n\}$ and $i \in \{0, t-1\}$
set $K_{ID} = G(K_{ID})$		find (ID_j, K_j) s.t. $c = F(G^i(K_j), a)$
		set $K_j = G^{i+1}(K_j)$

Figure 4.5: An OSK protocol variant

4.6.3 Forward Privacy Without PKC

Besides claiming strong privacy is impossible, Vaudenay [93] also posed an open research question asking whether forward privacy without PKC is possible. We answer the open question here. Consider a variant of the OSK protocol [74] that appeared in [93] in figure 4.5.

This protocol is proven to be *Narrow-Destructive*-private in [93]. Recall that *Narrow-Destructive* $\Rightarrow$ *Narrow-Forward*, this protocol is also *Narrow-Forward*-private. We note that our proposition 2 *cannot* be applied to this protocol because false-negative can happen when a legitimate tag has been queried for t times by an adversary before it is queried by the reader again. Since K_{ID} would become $G^t(K_{ID})$ by then and the reader will only try $0 \leq i \leq t-1$ different $G^i(K_j)$ values per (ID_j, K_j) pair to find a matching $F(G^i(K_j), a)$ for c, that legitimate tag will not be identified successfully by the reader, hence a false-negative occurs. In other words, calling `Result`(π) in this case helps the adversary to gain advantage in winning the privacy experiment, which causes this protocol to be *Narrow-Forward*-private only but not *Forward*-private. Hence, leaving the question *"whether Forward-private without PKC is possible"* open.

Here, we would like to apply proposition 2, so that *Forward* is not different from *Narrow-Forward*, and the OSK variant protocol will become also *Forward*-private, and hence, it will answer the open problem. First of all, we notice that the reason why there can be false-negative is due to $i \leq t-1$[5]. Next, we consider the number of queries to a tag the adversary can make be q and we assume that $q \leq t$. In other words, the adversary can never query any particular tag for more than t times and the reader is now always able to identify any legitimate tag, which also means there will not be any false-negative. This implies that proposition 2 can be applied and we have the OSK variant protocol become *Forward*-private.

[5]In the original OSK paper [74], this limitation does not exist in the protocol description, which is why Avoine showed that this protocol is secure in his paper [6], but later on Juels and Weis disagreed in [49] when this limitation was considered.

The only thing that is arguable is whether the assumption ($q \leq t$) makes any sense or not. Clearly, one can also argue that when $q > t$, then the privacy will not be satisfied any longer. Hence, the problem has turned to a a scalability issue: *"Can we always have a more resourceful reader compared to an adversary?"* In fact, the ability of an adversary can be limited by different means in reality. Limited tag queries due to the mobility of tags and throttling [43] are some realistic examples to support the assumption. In particular, for the low-cost RFID tag environment, it is more appropriate to consider a less almighty adversary model. Furthermore, seeking strong privacy in front of a powerful adversary for RFID that is known by its limited resources characteristic seems to be impractical.

4.7 Conclusion

When we first started our research, compare to the more than hundreds published RFID papers, RFID modellings received less attentions, not more than ten we could identified [5, 6, 17, 20, 43, 49, 50, 97]. But after Vaudenay published his strong privacy RFID model in [93], it seems that many researchers have picked up this research area since then. We could easily find out nearly 20 piece of works [2, 3, 7, 9, 18, 21, 22, 23, 34, 39, 51, 58, 65, 70, 77, 88, 91, 92, 99], including two of our papers [70, 71] are also inspired by the Vaudenay model. We were the first to find out the issue of the impossibility results in the Vaudenay model and provide a fix to that. Some later works [2, 3, 7, 22] also show supportive results to our findings. In this chapter, we examined the Vaudenay model in a great detail and presented some new results. Firstly, we examined the eight different classes presented in [93] and applied some reasonable assumptions to simplify the classification. Then, we presented a counter argument to [93] by stating that strong privacy in RFID is indeed achievable. In summary, to achieve strong privacy, tags are required to perform not only public key cryptography, but also require an additional reliable random source, which was missing from the description provided in [93]. Nonetheless, this results in a high manufacturing cost for RFID tags. However, in contrast to Vaudenay's result, we have shown that strong privacy is indeed achievable. Furthermore, we believe that in the future development of RFID, privacy will have to be sacrificed to keep the cost low. Hence, it is worthwhile to reconsider whether RFID should face such a strong adversary model. Due to the short communication range and infrequent access properties of RFID tags, we believe it is not necessary to assume the presence of powerful adversaries. Henceforth, an adequate

and appropriate privacy model, which takes into account the constraints of RFID is still missing.

Chapter 5

Family of Synchronised Authentication Protocols in RFID

Chapter Overview

In the previous chapter, we presented our works on RFID privacy modellings. We critically reviewed some of the early works on this field. We showed their inconsistencies and suggested a strong privacy model by Vaudenay. Based on his model, we proposed our refined version and proved our different findings from his results. This gives us new tool to evaluate the performance of RFID protocols efficiently. In this chapter, we will look at a new RFID protocol classification proposed by us, namely the family of *Synchronised Authentication Protocols* (SAP) in RFID. First we will define what they are and we will use our refined version of the Vaudenay model to evaluate their privacy protection performances. Then we will suggest a new RFID protocol construction method, which is a variant of the family of SAP that has a better privacy protection performance then the rest. Part of the results in this chapter has been published in [71] at ESORICS 2009. We have the following contributions in this chapter:

- We create a new RFID protocol classification that captures most of the RFID protocols proposed nowadays
- We evaluate the performance of the RFID protocols within this new classification using our refined Vaudenay model
- We propose a new RFID protocol construction method that can provide a better privacy protection
- We suggest some application scenarios for this new protocol

5.1 Synchronised Authentication Protocols in RFID

Many RFID authentication protocols with randomised tag response have been proposed to avoid simple tag tracing. These protocols are symmetric in common due to the lack of computational power to perform expensive asymmetric cryptography calculations in low-cost tags. Protocols with constantly changing tag key have also been proposed to avoid more advanced tag tracing attacks. With both the symmetric and constant-changing properties, tag and reader re-synchronisation is unavoidable as the key of a tag can be made desynchronised with the reader due to off-line attacks or incomplete protocol runs.

We are going to classify these synchronised RFID authentication protocols (SAPs) into different types and then examine their highest achievable levels of privacy protections using the refined Vaudenay model we have discussed in the previous chapter. We are the first to provide classification for synchronised RFID authentication protocols based on their construction methods, their structures and prove their limitations on privacy protections.

First, we will look into the general constructions of symmetric key RFID authentication protocols. Both tag-to-reader and mutual (i.e. tag and reader) authentication protocols are examined. We deduce that all of these protocols unavoidably require *tag key update* in the tag side and *tag key synchronisation* between tag and reader at some point of the protocol in order to provide better untraceability against stronger attacks. Then we classify these protocols into four main construction types based on when the *tag key update* and *tag key synchronisation* operations are carried out. We adopt the refined privacy model from chapter 4 to prove the highest privacy levels that can be attained in these protocols for each construction type. We do this by combining the results of [93] and [77] and constructing a universal generic attack for each construction type targeting a higher privacy level. Notice that our attacks are purely taking advantages of the adversary model defined in [93] but not exploiting various flaws in protocol designs. Our new privacy results show the separation between *weak privacy* and *narrow-forward privacy* in these protocols, which effectively fills the missing relationship of these two privacy levels in Vaudenay's paper and answer the question raised by Paise and Vaudenay in ASIACCS 2008 [77] on why they cannot find a candidate protocol that can achieve both privacy levels at the same time. We also show that *forward privacy* is impossible with these synchronised protocols.

5.1.1 Privacy Experiment

Let us recall the privacy experiment defined in [93]. The setup of privacy experiment requires a hidden table $\mathcal{T}$ to be maintained whenever the oracles `DrawTag`() and `Free`($vtag$) are called. This hidden table is not available to the adversary until the last step of the privacy experiment (to be reviewed below). When `DrawTag`() is called, a new entry of the pair $(vtag, ID)$ is to be added into $\mathcal{T}$. When `Free`($vtag$) is called, the entry with the same $vtag$ handle is to be marked unavailable. The true ID of the tag with handle $vtag$ is represented by $\mathcal{T}(vtag)$.

The privacy experiment that runs on an RFID protocol is defined as a game to see whether the adversary outputs *True* or *False* after seeing the hidden table $\mathcal{T}$. At the beginning, the adversary is free to access any oracles within his oracle collection according to his own attack strategy (which defines the maximum targeting privacy level to attack). Once the adversary finishes querying, the hidden table $\mathcal{T}$ will be released to him. The adversary will then analyse the $(vtag, ID)$ entries in the table using the information obtained before from the queries. If the adversary finally outputs *True* for the question whether $\mathcal{T}(vtag) = ID$ in a non-trivial sense (i.e. not blindly outputs *True* because $\mathcal{T}(vtag) = ID$ as listed in the table), then he has successfully traced a victim tag of identity ID and won the privacy experiment. We say that the RFID protocol being experimented is not L-private where L is the highest privacy level achievable from the oracle collection of the adversary.

5.1.2 Achievable Privacy Levels

As pointed out in [93] and by us in [70], (narrow-)strong privacy for tag authentication protocols is only achievable with PKC under the asymmetric key setting. The same result is supported by [77] for mutual authentication protocols. From the results we obtained, which will be presented below, we also agree to this impossibility result for RFID protocols under symmetric key setting. Hence, this will leave us with these six privacy levels:

$$\begin{array}{ccccc} \textit{Destructive} & \Rightarrow & \textit{Forward} & \Rightarrow & \textit{Weak} \\ \Downarrow & & \Downarrow & & \Downarrow \\ \textit{Narrow-Destructive} & \Rightarrow & \textit{Narrow-Forward} & \Rightarrow & \textit{Narrow-Weak} \end{array}$$

We have also proved in [70] that the destructive levels are only distinguishable from the forward levels as long as the RFID protocols share correlated secrets (e.g. global key, partial group key, etc.) among tags. Corrupting one tag in these protocols will also

reveal (partial) secrets of related tags. The majority of RFID protocols do not belong to this special protocol category. Hence we will only focus on RFID protocols where each tag is independent from each other and does not store any correlated secrets. This leaves us with four main privacy levels to be examined:

$$\begin{array}{ccc} \textit{Forward} & \Rightarrow & \textit{Weak} \\ \Downarrow & & \Downarrow \\ \textit{Narrow-Forward} & \Rightarrow & \textit{Narrow-Weak} \end{array}$$

5.1.3 Protocol Constructions

We look at different constructions of RFID authentication protocols (both tag-to-reader and mutual) under the symmetric key setting with or without *tag key update* and *tag key synchronisation*. We show the limitation of each of the constructions on achieving a certain privacy level in tag tracing. Before we define our protocol construction classifications, we have these notations:

- $\mathcal{O}^{Tag}(), \mathcal{O}^{Reader}()$: A collection of operations denoted as an oracle following the protocol specification carried out on the tag and reader sides respectively.
- K^i_{ID} : The tag key at instance i where the initial key is K^0_{ID}.
- S^i_{ID} : The tag state at instance i denoted as an encapsulation of the tag key K^i_{ID} and other per instance generated and received values. If S^i_{ID} is updated to S^{i+1}_{ID}, K^i_{ID} is updated to K^{i+1}_{ID} as well.
- $\mathcal{O}^{Update}(S^i_{ID})$: A tag key update oracle performed on the tag side which takes S^i_{ID} as input and outputs an updated K^{i+1}_{ID}.
- $\mathcal{O}^{Sync}(S^i_{ID})$: A tag key synchronisation oracle performed on the reader side which takes S^i_{ID} as input and outputs a synchronised K^d_{ID}. It is a recursive function which has an upper bound n where $n + i \geq d > i$ or $d = i - 1$. The upper bound is added to reflect the side-channel attack effect described in [49].

It is important for us to state that we do not concern about how RFID authentication protocols are implemented. Some may use simple bitwise operations like XOR, some may use hashing functions, some may even use symmetric encryption/decryption. We only classify them based on how and when $\mathcal{O}^{Update}(S^i_{ID})$ is executed. For an RFID authentication protocol to fall into one of the following construction types, the bottom line is that the protocol has to be at least correct (i.e. when the protocol is started

with $\pi \leftarrow \texttt{Launch}()$, then by calling $\texttt{Result}(\pi)$, it should output 1, with overwhelming probability, for legitimate tags and 0 otherwise). Protocols that fail this basic requirement should not be defined as authentication protocol at all. We classify RFID authentication protocols into the following four construction types:

- **Type 0** : Protocols that are correct and lack tag key update mechanisms or equivalently even with $\mathcal{O}^{Update}(S^i_{ID})$ implemented it can not be executed properly as if it is not there, which causes K^i_{ID} remains static at the end of the protocol [1].
- **Type 1** : Protocols that are correct and $\mathcal{O}^{Update}(S^i_{ID})$ can be executed properly, which causes K^i_{ID} to change every time the protocol is executed.
- **Type 2a** : Mutual authentication protocols that are correct and $\mathcal{O}^{Update}(S^i_{ID})$ is executed properly *after* the final reader authentication message is received, which causes K^i_{ID} to change after the reader is authenticated.
- **Type 2b** : Mutual authentication protocols that are correct and $\mathcal{O}^{Update}(S^i_{ID})$ is executed properly *before* the final reader authentication message is received, which causes K^i_{ID} to change before the reader is authenticated.

5.2 Our New Privacy Results on SAPs

We can now formally analyse the four symmetric RFID protocol construction types. For each of them, we will prove the impossibility for it to achieve a certain privacy level with a universal attack. It is important to note that these attacks are *generic* and *universal* as they are only constructed using the oracles defined in previous section. We do not need to exploit any design flaw in the protocols in order to make the attacks success. Hence the attacks are valid as long as the same adversary model is applied.

Also, as our results are about the highest achievable privacy levels, not the lowest, there can be some protocols of the same construction type that only achieve a weaker privacy level. For protocols that do not provide privacy protection at all, we represent them with a special class *Nil*. Since we are not claiming the lowest achievable privacy level for the protocols, we do not consider the separation between any weaker privacy

[1] Some protocols, for example the YA-TRAP [85], although they have some tag key update mechanisms, they are known to have design flaws that effectively render their key update mechanisms useless (i.e. as if the tag key is never updated), we do not classify these protocols to have tag key update. Readers can refer to [6, 49, 90] for more specific attacks on existing protocols based on their design flaws.

Tag$\{K_{ID}\}$		Reader$\{ID, K_{ID}\}$
v: random value	$\xleftarrow{\quad Query, c \quad}$	*c: random challenge*
$S_{ID} : \{K_{ID}, c, v\}$		
$Response \leftarrow \mathcal{O}^{Tag}(S_{ID})$	$\xrightarrow{\quad Response \quad}$	*r: Response*
		$\forall i \in \{ID\}, S_i : \{K_i, r, c\}$
		$Verify\ if\ r = \tilde{r} \leftarrow \mathcal{O}^{Reader}(S_i)$
		$if\ FOUND,\ set$ `Result`$(.) = 1$
		$else\ set$ `Result`$(.) = 0$

Figure 5.1: **Type 0** SAP construction

levels weaker than *Weak privacy* as defined in [93] and just group them all into the special class *Nil*.

For each of the construction types, we abstract the common form of that type of protocols in a figure for illustration purpose. There can be variations on how the reader verifies legitimate tags responses and how the messages flow. But what in common is whether there is tag key update or not and if there is, when is it executed? Again, our universal attacks do not concern the implementation details of these protocols, hence they are universal.

5.2.1 Type 0 Protocols Can Never Achieve Forward Privacy Levels

Construction

Type 0 represents the most basic form of an RFID authentication protocol that uses symmetric key without tag key update. Protocols in [19, 54, 55, 63, 64, 69, 79, 85, 94] are some examples. It should be trivial for most readers that forward privacy is impossible in this type of construction, since tag corruption will reveal the static tag key. It still serves as a base in our classifications because we will reduce some other construction types to this type in the following sections. Here we look at the common construction of this type of protocols in figure 5.1.

Since there is no $\mathcal{O}^{Update}(S^i_{ID})$, both tag and reader keep the same K_{ID} value through out the life time of the tag. Without tag key update, protocols with this construction can never achieve forward privacy and narrow-forward privacy. Because forward privacy is harder than narrow-forward privacy, we only need to show that narrow-forward privacy is not achievable. Consider the following attack:

1. $\texttt{CreateTag}^1(ID_0)$, $\texttt{CreateTag}^1(ID_1)$
2. $vtag \leftarrow \texttt{DrawTag}()$
3. $\pi \leftarrow \texttt{Launch}()$
4. $c \leftarrow \texttt{SendReader}(\pi, Init)$
5. $r : Response \leftarrow \texttt{SendTag}(vtag, c)$
6. (Forward r to reader to close π) $null \leftarrow \texttt{SendReader}(\pi, r)$
7. $\texttt{Free}(vtag)$
8. $vtag' \leftarrow \texttt{DrawTag}()$
9. $K_{ID_x} \leftarrow \texttt{Corrupt}(vtag')$
10. Queries ended, receive $\mathcal{T}(vtag) = ID_b$
11. Let $S_{ID_x} : \{K_{ID_x}, r, c\}$, if $r = \tilde{r} \leftarrow \mathcal{O}^{Reader}(S_{ID_x})$ then $x = b$. Otherwise $x = |1-b|$
12. Output whether $\mathcal{T}(vtag') = ID_x$

The idea of the attack is to record a protocol instance between a legitimate tag and a reader. A random tag is then corrupted and its tag key is exposed. By simulating a protocol run using the exposed tag key, if the result is the same as the recorded one, then the same tag is found with high confident. An adversary running the attack above will only fail (i.e. $\mathcal{T}(vtag') \neq ID_x$) if $\mathcal{O}^{Reader}(S_{ID_0}) = \mathcal{O}^{Reader}(S_{ID_1})$. This should only happen with a negligible probability, otherwise the protocol is simply incorrect, which produces wrong identification. Hence the adversary will succeed with overwhelming probability. Since there is no further oracle access after $\texttt{Corrupt}(vtag')$ and no $\texttt{Result}(\pi)$ in the attack, this is a significant narrow-forward privacy level attack. We have shown that RFID protocols without tag key update is not narrow-forward private and hence not forward private.

Remark 1.

A **Type 0** construction RFID protocol presented in [93] using pseudorandom function (PRF) has been proved to provide weak privacy. Hence it is the highest privacy level that can be attained by RFID protocols with **Type 0** construction. Our conclusion for **Type 0** construction summarised in table 5.1.

Type 0	*Forward levels*	*Weak levels*	*Nil*
Non-narrow levels	-	✓	✓
Narrow levels	-	✓	

Table 5.1: Achievable privacy levels of **Type 0** construction

Tag$\{K^i_{ID}\}$		Reader$\{ID, K^i_{ID}\}$
v: random value	$\xleftarrow{\textit{Query}, c}$	*c: random challenge*
$S^i_{ID} : \{K^i_{ID}, c, v\}$ $Response \leftarrow \mathcal{O}^{Tag}(S^i_{ID})$ $K^{i+1}_{ID} \leftarrow \mathcal{O}^{Update}(S^i_{ID})$ $i = i + 1$	$\xrightarrow{\textit{Response}}$	*r: Response*, $\forall j \in \{ID\}$ $K^d_j \leftarrow \mathcal{O}^{Sync}(S^i_j)$, $S^d_j : \{K^d_j, r, c\}$ *Verify if* $r = \tilde{r} \leftarrow \mathcal{O}^{Reader}(S^d_j)$ *if* $FOUND$, *set* `Result`$(.) = 1$, $K^i_j = K^d_j$; *else set* `Result`$(.) = 0$

Figure 5.2: **Type 1** SAP construction

5.2.2 Type 1 Protocols Can Never Achieve Non-Narrow Privacy Levels

Since the static tag key has limited the highest achievable privacy level of **Type 0** protocols to weak privacy only, tag key update is incorporated in the construction of protocols to help rising the privacy level. Protocols in [10, 37, 74, 75] are some examples. **Type 1** protocols are **Type 0** protocols with tag key update and tag key synchronisation. See figure 5.2 for their construction.

Since $\mathcal{O}^{Update}(S^i_{ID})$ is executed every time on the tag side, the stored K_{ID} inside the tag is always changing [2]. Although now there is tag key update, an adversary can cause desynchronisation between tag and reader so that protocols with this construction can never achieve forward privacy and weak privacy. Because forward privacy is harder than weak privacy, we only need to show that weak privacy is not achievable. Consider the following attack:

1. `CreateTag`$^1(ID_0)$, `CreateTag`$^1(ID_1)$

[2]Notice that $\mathcal{O}^{Update}(S^i_{ID})$ is executed before the tag response is sent out. Although updating the key after response does not change the protocol result, this is a good practice to avoid tag corruption by an adversary at the moment right after the response is captured but before $\mathcal{O}^{Update}(S^i_{ID})$ is executed (i.e. keeping the old tag key in the memory).

2. $vtag \leftarrow \texttt{DrawTag}()$
3. $\pi \leftarrow \texttt{Launch}()$
4. $c \leftarrow \texttt{SendReader}(\pi, Init)$
5. $r : Response \leftarrow \texttt{SendTag}(vtag, c)$
6. (Forward r to reader to close π) $null \leftarrow \texttt{SendReader}(\pi, r)$
7. (Use the same c to query $vtag$) Repeat n times:
8. $\quad r : Response \leftarrow \texttt{SendTag}(vtag, c)$
9. $\texttt{Free}(vtag)$
10. $vtag' \leftarrow \texttt{DrawTag}()$
11. $\pi' \leftarrow \texttt{Launch}()$
12. $c' \leftarrow \texttt{SendReader}(\pi', Init)$
13. $r' : Response \leftarrow \texttt{SendTag}(vtag', c')$
14. $null \leftarrow \texttt{SendReader}(\pi', r')$
15. $z \leftarrow \texttt{Result}(\pi')$
16. Queries ended, receive $\mathcal{T}(vtag) = ID_b$
17. If $z = 0$ then $x = b$. Otherwise $x = |1 - b|$
18. Output whether $\mathcal{T}(vtag') = ID_x$

An adversary running the attack above makes use of the maximum desynchronised key states n such that K_{ID}^i becomes K_{ID}^{n+1+i}. The desynchronised tag will not be recognised by the reader anymore because $\mathcal{O}^{Sync}(S_{ID}^i)$ will not run recursively beyond n (or even if n is infinity, desynchronised tag can be distinguished with a side-channel attack on the time taken for the reader to recognise that tag as described in [49]). The adversary will only fail if $\texttt{Result}(\pi')$ still outputs 1 for the desynchronised-beyond-n-tag (i.e. the tag is still authenticated). This means $K_{ID}^{n+1+i} = K_j^m$ for some $j \in \{ID\}$ and $0 \leq m \leq n$ (i.e. a duplicate tag key), which should only happen with negligible probability. Hence the adversary will succeed with overwhelming probability. Since there is no $\texttt{Corrupt}(vtag')$ in the attack, this is a significant weak privacy level attack.

Type 1	*Forward levels*	*Weak levels*	*Nil*
Non-narrow levels	-	-	✓
Narrow levels	✓	✓	

Table 5.2: Achievable privacy levels of **Type 1** construction

We have shown that RFID protocols with tag key update is not forward private and not weak private.

Remark 2.

A **Type 1** protocol presented in [93] using random oracle model has been proved to provide narrow-destructive privacy, which is equivalent to narrow-forward privacy since the protocol does not have correlated secrets among tags. Hence the highest privacy level that can be attained by **Type 1** protocols is narrow-forward. We conclude the result of **Type 1** construction in table 5.2.

Remark 3.

Another interesting remark is the separation result of the weak privacy level and the narrow-forward privacy level, which was not obtained in [93] and it was asked in [77] if achieving both privacy levels with symmetric key only is feasible or not. Clearly, there are only protocols that either do not update the tag key (**Type 0**) or protocols that update it (**Type 1**). They span the whole protocol set and we do not have overlapping between weak privacy level and narrow-forward privacy level according to our results in 4.3 and 4.4. Hence we have shown the separation here and answered the question.

Remark 4.

As pointed out in [70], let q be the number of queries in the above attack and assume that $q \leq n$, then there can be protocols, using symmetric key only, that achieve forward privacy level. This is the highest privacy level for symmetric key protocols. However, we do not consider that assumption here.

5.2.3 Type 2a Protocols Can be Reduced to Type 0 Protocols

Without reader authentication, any adversary can keep querying a tag with any compatible reader until it is desynchronised with legitimate reader. Mutual authentication

Tag$\{K^i_{ID}\}$		Reader$\{ID, K^i_{ID}\}$
v: *random value*	$\xleftarrow{\textit{Query}, c}$	c: *random challenge*
$S^i_{ID} : \{K^i_{ID}, c, v\}$		
$Response \leftarrow \mathcal{O}_1^{Tag}(S^i_{ID})$	$\xrightarrow{Response}$	r: $Response$, $\forall j \in \{ID\}$
		$K^d_j \leftarrow \mathcal{O}^{Sync}(S^i_j)$, $S^d_j : \{K^d_j, r, c\}$
		$Verify\ if\ r = \tilde{r} \leftarrow \mathcal{O}_1^{Reader}(S^d_j)$
$a : Auth$, $Verify\ if$	$\xleftarrow{Auth}$	$if\ FOUND$, set `Result`$(.) = 1$,
$a = \tilde{a} \leftarrow \mathcal{O}_2^{Tag}(S^i_{ID})$		$K^i_j = K^d_j$, $Auth \leftarrow \mathcal{O}_2^{Reader}(S^d_j)$;
$if\ MATCHED$,		$else\ set$ `Result`$(.) = 0$
$K^{i+1}_{ID} \leftarrow \mathcal{O}^{Update}(S^i_{ID})$,		
$i = i + 1$		

Figure 5.3: **Type 2a** SAP construction

protocols add an additional authentication message for the reader in the protocol construction to safeguard the query is in fact coming from a legitimate reader. **Type 2a** protocols update the tag key after such reader authentication message is received. Protocols in [24, 38, 41, 52, 60, 61, 76, 82, 95] are some examples. Their construction can be represented in figure 5.3.

With tag key update after reader authentication, it protects the protocol from the desynchronised-beyond-n attack discussed before because each update must now come with a valid reader authentication message, which can be hard to forge. As a result, the tag key can only be desynchronised within one update. If the reader stores both the updated tag key value and the previous tag key value, in case the tag fails to update its tag key (most likely because of adversarial attacks), the reader can still authenticate the victim tag using the previous tag key in the next protocol instance. This measure is enough to provide weak privacy to this type of protocol construction.

However, imagine an offline attack to tag where invalid reader authentication message is sent. This has the same effect as if the valid reader authentication message is blocked or intercepted in an online attack but of course the former one is easier to launch. These kinds of attacks cause the tag fail to execute $\mathcal{O}^{Update}(S^i_{ID})$ because the reader is never authenticated. It is not hard to see that the protocol is now reduced to **Type 0** protocol as if there is never an $\mathcal{O}^{Update}(S^i_{ID})$ oracle being implemented in the protocol construction. As inherited from **Type 0** protocol, forward privacy levels cannot be achieved. A formal description of the attack is presented below:

1. $\texttt{CreateTag}^1(ID_0)$, $\texttt{CreateTag}^1(ID_1)$

Type 2a	*Forward levels*	*Weak levels*	*Nil*
Non-narrow levels	-	✓	✓
Narrow levels	-	✓	

Table 5.3: Achievable privacy levels of **Type 2a** construction

2. $vtag \leftarrow \texttt{DrawTag}()$
3. $\pi \leftarrow \texttt{Launch}()$
4. $c \leftarrow \texttt{SendReader}(\pi, Init)$
5. $r : Response \leftarrow \texttt{SendTag}(vtag, c)$
6. (Forward r to reader to close π) $Auth \leftarrow \texttt{SendReader}(\pi, r)$
7. (Replace $Auth$ with a random value $a \neq Auth$)
8. $null \leftarrow \texttt{SendTag}(vtag, a)$
9. (No $\mathcal{O}^{Update}(.)$ is executed) $\texttt{Free}(vtag)$
10. $vtag' \leftarrow \texttt{DrawTag}()$
11. $K_{ID_x} \leftarrow \texttt{Corrupt}(vtag')$
12. Queries ended, receive $\mathcal{T}(vtag) = ID_b$
13. Let $S_{ID_x} : \{K_{ID_x}, r, c\}$, if $r = \tilde{r} \leftarrow \mathcal{O}^{Reader}(S_{ID_x})$ then $x = b$. Otherwise $x = |1-b|$
14. Output whether $\mathcal{T}(vtag') = ID_x$

Other than the negligible case where $\mathcal{O}^{Reader}(S_{ID_0}) = \mathcal{O}^{Reader}(S_{ID_1})$, the above attack will only fail if the random value a is accepted by the tag such that $\mathcal{O}^{Update}(.)$ is executed to update the tag key. This should also happen with negligible probability, otherwise the reader authentication message can be easily forged. Hence the adversary will succeed with overwhelming probability. Since there is no further oracle access after $\texttt{Corrupt}(vtag')$ and no $\texttt{Result}(\pi)$ in the attack, this is a significant narrow-forward privacy level attack. We have shown that RFID protocols with tag key update after the reader is authenticated work as best as the **Type 0** protocols. We conclude the result of **Type 2a** construction in table 5.3.

Tag$\{K^i_{ID}\}$		Reader$\{ID, K^i_{ID}\}$
v: random value	$\xleftarrow{\textit{Query}, c}$	*c: random challenge*
$S^i_{ID} : \{K^i_{ID}, c, v\}$		
$Response \leftarrow \mathcal{O}^{Tag}_1(S^i_{ID})$	$\xrightarrow{Response}$	*r: Response*, $\forall j \in \{ID\}$
$K^{i+1}_{ID} \leftarrow \mathcal{O}^{Update}(S^i_{ID})$		$K^d_j \leftarrow \mathcal{O}^{Sync}(S^i_j)$, $S^d_j : \{K^d_j, r, c\}$
$i = i + 1$		*Verify if* $r = \tilde{r} \leftarrow \mathcal{O}^{Reader}(S^d_j)$
		if $FOUND$, *set* `Result`$(.) = 1$,
a : *Auth, Verify if*	$\xleftarrow{Auth}$	$K^i_j = K^d_j$, $Auth \leftarrow \mathcal{O}^{Reader}(S^d_j)$
$a = \tilde{a} \leftarrow \mathcal{O}^{Tag}_2(S^i_{ID})$		*else set* `Result`$(.) = 0$

Figure 5.4: **Type 2b** SAP construction

5.2.4 Type 2b Protocols Can be Reduced to Type 0 or Type 1 Protocols

Type 2b protocols update the tag key before the reader authentication message is received. Examples are in [59, 77]. We acknowledge that the reduction from this construction type to **Type 1** is simple: an adversary just needs to block the last reader authentication message and the protocol is identical to a **Type 1** protocol. In fact, it is very uncommon to see protocols with such construction. It is only included in here for completeness. The construction can be represented by figure 5.4.

With tag key update before reader authentication, it makes sure that the tag key is changed even if the reader authentication message is blocked or incorrect, such that when facing a (narrow) forward privacy adversary, the corrupted tag key cannot be used to relate to any previous protocol instance. However, this is true only if tags update their keys regardless of the correctness of the reader authentication result. This means that the tag key is updated as if there is no reader authentication or a failed reader authentication does not affect the next protocol instance (e.g. a stateless RFID tag). An adversary can launch a desynchronisation attack to these protocols because they do not take advantage of reader authentication. Clearly, this performs as best as **Type 1** protocols (an example in [77]). The only exception we can think of is when the tag takes the reader authentication result into account (e.g. rewinds back to the previous tag key if the reader authentication is failed) or the result will affect the next protocol instance (e.g. a stateful RFID tag). However, an adversary can still use the same attack described in section 5.2.3 to freeze the tag key or tag state and the protocol is reduced into a **Type 2a** protocol. We do not repeat the same attack here but conclude

Type 2b	*Forward levels*	*Weak levels*	*Nil*
Non-narrow levels	-	✓ (stateful tag)	✓
Narrow levels	✓ (stateless tag)	✓	

Table 5.4: Achievable privacy levels of **Type 2b** construction

the result of **Type 2b** construction in table 5.4

5.3 Seeking for a Better Solution

We defined four RFID authentication protocol constructions and investigated on their highest achievable privacy levels. From the results we obtained, forward privacy cannot be achieved by any type of synchronised symmetric protocol constructions. Furthermore, there is no privacy improvements at all with an extra reader authentication message. After all, under the symmetric key setting, RFID authentication protocols have limited privacy protections against tag tracing and a candidate that provides both weak privacy and narrow-forward privacy protections does not exist. This provides us a potential answer to the open question in [93], which is, forward privacy without PKC is impossible. This claim remains valid until some special symmetric protocols that do not fall into one of our four constructions types can be found, then we need another examination. However, it is important for us to make ourselves clear that we do not claim our results on all the symmetric RFID protocols, instead, all our findings are bounded by the current adversary model defined in [93], [70] and [77]. This leaves the possibility that there may exist some symmetric RFID protocols not included in or well described by the Vaudenay's model where our results do not apply on them. Hence, one may be able to find alternative ways to overcome the limitations of RFID protocols by choosing more expensive cryptographic primitives in the design of RFID protocols or tweaking the privacy model where different assumptions are used in order to reflect some other RFID applications or scenarios. With this in mind, our results are still valid as long as the RFID protocol being examined has the same settings and assumptions as stated in this chapter.

We have shown there are two powerful attacks that make tag tracing possible, even when the tag secrets are constantly changing. Ironically, it is the changes that aided the attacks. There were no official names for these two attacks. Based on

their nature, we will refer them as the *querying-attack* (by a *querying-attacker*) and the *blocking-attack* (by a *blocking-attacker*), both of these attacks can be classified as some denial-of-service (DOS) attacks. The idea is to make the reader and the victim tag desynchronise (i.e. adding a "mark" onto the victim tag) such that the attacker can spot the abnormal behaviour of the reader to a suppose-to-be legitimate tag. Such subtle difference reflects the fact that the victim tag is there and hence being traced again. As their names suggested, querying-attack is an attack that tries to desynchronise reader and tags by issuing queries repeatedly. Since a tag will change its secrets after each query, there will be a point that the reader cannot catch up with the changes. For example, the reader may store or configured to search up to next n tag secrets per tag due to resource, performance and user experience issues, a querying-attacker can successfully desynchronise the reader and tags by sending out $n+1$ queries. Even if the reader has infinite resource, the extraordinary time required to process the much advanced tag compares to other ordinary tags provides a side-channel information to the attacker [49] and the result remains the same. On the other hand, blocking-attacker tries to desynchronise reader and tags by blocking the reader-to-tag authentication messages. Without the reader authentication message to confirm the reception of the tag response, the tag may choose to keep the current tag secrets without changing them in order to avoid desynchronisation. Combining such blocking-attack with a forward-attacker (i.e. a blocking-forward-attacker), the same old tag secrets can be used to decode the communication after the last successful query.

To protect RFID applications from *forward-attack*, RFID tag is made to change its symmetric tag key constantly to provide unlinkability. However, this introduced *blocking-attack* and *querying-attack*, which remain to be the strongest and unsolvable attacks. We notice that the main weakness in SAPs is that RFID tags can be easily made desynchronised with the reader. This is mainly because the tags cannot authenticate the reader first at the very beginning when the reader sends out the query. As RFID tags are mostly passive, they are powered up by the reader query signal and hence the reader will always be the first to send out the message in an RFID protocol. As a result, RFID tags will response to every reader query, no matter the reader is legitimate or not. Consider this, if the query message from the reader contains a verifiable value such that the tag can use it to authenticate the reader, knowing that this query message is originated from a legitimate reader, then the tag will not be made desynchronised because the attacker lacks the knowledge to create valid queries to launch the querying-attack. Of course the solution is not as simple as this. Next

we need to make sure a valid query cannot be replayed, otherwise the attacker can simply record a valid query and launch *querying-attack* with it. This can be done in two directions: first, update the tag key after each reader query. Second, embed a random nonce in each reader query. Now, since the reader authentication is sent first, before a tag gives out valid response, blocking-attack will not be success as a tag will take no action if the reader message is blocked. A blocking-attacker may now turn to block the reception of the tag-to-reader response, in order to desynchronise the reader. This can be easily fixed by storing both the previous and the current tag secrets.

This twist, moving the reader authentication message to the beginning, seems to be a promising anti-tag tracing measurement, but it is not as easy to carry out as it seems. The reader now needs to authenticate itself in the first message to the tags. If all the tags can verify such message, that means a common long-term secret is shared among all the tags. This is dangerous in a symmetric key system: once a tag is compromised, the revealed common secret can be used to jeopardise the whole system. Hence the authentication message has to be unique per tag (ideally to be unique per tag per query). If it is unique per tag, that means the message must contain a value computed using also the secrets of the receiving tag such that only the receiving tag can verify it. As the tag changes its secrets after each query, the reader authentication message can be made unique per query. There is just one major question remains: *"How do we let the reader chooses which tag secrets to use to authenticate itself before any tag has identified and authenticated itself to the reader?"*. As a common saying, this is the chicken or the egg dilemma and we are going to investigate on this.

In order to strengthen the privacy protection in RFID applications, provided that the previous SAP constructions have their weaknesses and limitations, we propose a new SAP construction method. The new construction differs from the others in a way that we consider RFID applications using SAP with the protocol structure where the reader authentication is done before the tag responses, in contrast to SAPs where there is no reader authentication (cf. **Type 0**, **Type 1** protocols in [71]) or the reader authentication is done after the tag has authenticated itself (cf. **Type 2a**, **Type 2b** protocols in [71]). We ask a simple question : "If the reader can authenticate itself to the tags first, does that help strengthen the privacy of the application?" and we show the answer in the following sections. Our findings turned out to be positive. For applications where reader authentication is done first, it can protect the RFID tags from being traced even in front of strong and powerful adversaries like forward-attackers, querying-attackers, blocking-attackers and a hybrid of them. We give formal

proves to these claims. Instead of only staying on the theoretical level, we also provide practical example applications where our new SAP construction can be applied.

5.4 Our New RFID System Model

We define our RFID system model in a traditional way like many others have commonly defined. There are three entities in the system: a back-end server, a reader and a collection of tags. Our main focus is on the wireless RFID communication between the reader and the tags, so we have the following assumptions on our system model when we do privacy evaluation to simplify its complexity.

5.4.1 Core Assumptions

An RFID system can be composed of multiple readers, each with its own back-end server. Since we are only interested in the performance of tag tracing protections of the SAP, without loss of generality, we assume there is only one legitimate system reader in the system and every RFID tag will be communicating with it. A legitimate system reader is the only reader that can access (and hence connected to) the system back-end server. The access to the server is done through a secure channel as such, no attack to the back-end server is possible. Compromising the reader is not possible either as the back-end server is separated from the reader in practice. No secrets will be revealed from the reader too thanks to its tamper-proof protection. In other words, all the secrets stored in the back-end server remain unaccessible from the adversaries. Since the reader and the back-end server are linked up through out the lifetime of the system, we refer to them as a single entity denoted as $\mathcal{R}$ from now on. RFID tags, on the other hand, are much more vulnerable. We assume there is no tamper-proof protection installed on RFID tags: when a tag is captured by the adversaries, all the tag secrets can be extracted promptly. Hence the security of the system cannot be based on the secrecy of the tag secrets. The adversaries, however, can use the compromised tag secrets to create a clone of the compromised tag to spoof the system. Whether these clone tags can be caught is beyond the scope of this work, after all we only focus on tag tracing protections. Each tag $\mathcal{T}$ has its own unique ID, which is only known to $\mathcal{R}$. We will use the notion $\mathcal{T}_i$ to refer to the tag with ID i. What is stored in a tag is some tag secrets, denoted as K_i, that provide enough information for $\mathcal{R}$ to find the true ID of the tag where the tuple $\{i, K_i\}$ is stored and maintained inside $\mathcal{R}$. We assume there are no false positive (i.e. misidentification) on fake tags nor false negative (i.e.

Reader $\{K_{i,i=1\ldots n}\}$		Tag $\{K_{ID}\}$
Picks a random *nonce*	$\xrightarrow{\textbf{Query,Challenge}}$	Generates a random value *rand*
$K_{ID} \leftarrow \texttt{Auth}^1_{\mathcal{R}}(K_{i,i=1\ldots n}, nonce, resp)$	$\xleftarrow{\textbf{Response}}$	$resp \leftarrow \texttt{Auth}^1_{\mathcal{T}}(K_{ID}, nonce, rand)$
$ver \leftarrow \texttt{Auth}^2_{\mathcal{R}}(K_{ID}, nonce)$	$\xrightarrow{\textbf{Verification}}$	$Pass \leftarrow \texttt{Auth}^2_{\mathcal{T}}(K_{ID}, nonce, ver)$
$K'_{ID} \leftarrow \texttt{Auth}^3_{\mathcal{R}}(K_{ID})$		$K'_{ID} \leftarrow \texttt{Auth}^3_{\mathcal{T}}(K_{ID})$

Figure 5.5: Traditional three-round RFID mutual authentication protocol construction

unidentifiable) on legitimate tags to happen in the system under normal circumstances unless it was the work of the adversaries.

5.4.2 System Setup

At the beginning of the system start up, $\texttt{SetupReader}(1^s)$ is called to properly setup $\mathcal{R}$. It generates the required public system parameters *param* and the reader secrets O with the inputting security parameter s. *param* is available to anyone including the adversaries to properly setup their own reader to be ready to communicate with the system tags. Each system tag is empty in the beginning and is being setup with $\texttt{SetupTag}(ID, param, b)$. It generates per tag unique secrets K_{ID} for tag $\mathcal{T}_{ID}$ using a unique ID and the public parameters *param* as input. Adversaries can also call this to setup their own tags. b is used to indicate whether this tag will be recognised by the system or not. If $b = 1$, the tag $\mathcal{T}_{ID}$ will be created as a legitimate tag and its ID and K_{ID} will be stored in $\mathcal{R}$; otherwise it will not be registered with the system (i.e. a fake tag) and is only used by the adversaries to communicate with $\mathcal{R}$. The underlying RFID authentication protocol $\texttt{Auth}()$ is prepared during system setup as well. We use $\texttt{Auth}^i_{\mathcal{T}}(.)$ to indicate the i-th operation of the protocol on the tag side and $\texttt{Auth}^i_{\mathcal{R}}(.)$ for the reader side.

5.4.3 RFID Authentication Protocol Construction

Since the RFID tags are passive devices, which are powered up with the reader query signal, the reader will always be the first to initiate a communication. Figure 5.5 shows the traditional three-round RFID mutual authentication protocol construction.

This construction is classified as a **Type 2a** SAP in [71] and is shown to fail forward privacy in front of a blocking-forward-attacker. In this type of SAP, the reader first picks a random challenge and broadcasts it together with the query message. Using its own secret key K_{ID}, the tag generates a response with the received challenge. The

reader then search through its list of K_i to look for a matching tag key that will compute the same response with the challenge. Once the matching tag key is found, the reader can generate a verification for that particular tag using the found tag key. If the verification is correct, the tag knows that it has been communicating with a legitimate reader and the tag key will be updated to a new value. A blocking-forward-attacker can trace the tags in this type of SAP by first recording the challenge from the reader and the response of the target tag and then block the tag from receiving the verification message, such that the tag will keep the same tag key unchanged. In the next round, a tag is compromised to see if the stored tag key can generate the same recorded response with the recorded challenge. In case there are only two tags left, the adversary can always tell if the compromised tag or the other tag was the one being queried previously and hence, the adversary has successfully traced the target tag.

We now propose an alternate SAP construction where the reader authenticates itself in the first message together with the query. For now, we just assume that the reader has already picked a tag key to use. We will discuss how can this becomes practical in later section. Notice that we do NOT assume the tag key that the reader picked is ALWAYS CORRECT and we think it is too strong an assumption for a practical privacy model. A trivial example is that since the reader is broadcasting the query, some of the receiving tags must have a different tag key than the one the reader picked, which is an incorrect tag key for these tags. We also allow the situation that the reader and the receiving tag may be desynchronised and possess different states of the tag key even though the receiving tag in fact has a matching ID on the reader side database. i.e. what we assume is merely the action that a tag key is picked by the reader, there is no guarantee that the tag key is correct, hence it is totally possible that $\texttt{Auth}^1_{\mathcal{T}}(.)$ will output $\perp$ instead of $Pass$ with the received message. This is very important as we do not want to make our model unrealistic with such a strong assumption and turns out to be impractical. The alternate SAP construction is presented in figure 5.6. Following the tradition, we will refer this as a **Type R** SAP since the authentication order is now *reversed*.

5.5 Our New RFID Adversary Model

We adapt the flexible adversary model proposed by Vaudenay in [93]. Since we have a new system model, the original adversary model is properly modified to reflect the changes. From now on, we will denote the adversary as $\mathcal{A}$.

Reader $\{K_{ID}\}$		Tag $\{K_{ID}\}$
Picks a random *nonce*		
$ver \leftarrow \texttt{Auth}^1_{\mathcal{R}}(K_{ID}, nonce)$	$\xrightarrow{\textbf{Query,Challenge, Verification}}$	$Pass \leftarrow \texttt{Auth}^1_{\mathcal{T}}(K_{ID}, nonce, ver)$
		Generates a random value *rand*
$Pass \leftarrow \texttt{Auth}^2_{\mathcal{R}}(K_{ID}, nonce, resp)$	$\xleftarrow{\textbf{Response}}$	$resp \leftarrow \texttt{Auth}^2_{\mathcal{T}}(K_{ID}, nonce, rand)$
$K'_{ID} \leftarrow \texttt{Auth}^3_{\mathcal{R}}(K_{ID})$		$K'_{ID} \leftarrow \texttt{Auth}^3_{\mathcal{T}}(K_{ID})$

Figure 5.6: **Type R** SAP construction

5.5.1 Adversary Abilities

The abilities of the adversaries are modelled as oracle accesses. Based on the different attack strategies of the adversaries and which oracles are accessed, we can classify them into different types of adversaries. The following are the oracles provided in our adversary model:

- `CreateReader`($param$) equips $\mathcal{A}$ with a non-system reader $\bar{\mathcal{R}}$ ready to communicate with the system tags. $param$ is the public parameters obtained from the system during system start up, which is the output of `SetupReader`(1^s).

- `CreateTag`(ID) equips $\mathcal{A}$ with a non-system tag $\bar{\mathcal{T}}_{ID}$ on $\mathcal{A}$'s choice of ID ready to communicate with the system reader. `SetupTag`($ID, param, 0$) is called to properly setup the tag and returns the tag key $\bar{K}_{ID}$ to $\mathcal{A}$. Note that $b = 0$ and hence $\bar{\mathcal{T}}_{ID}$ is not registered with the system. It is assumed that the IDs of all the non-system tags created with this oracle are not the same as any of the system tags.

- `DrawTag`() returns a virtual reference $vtag$ to one of the tags randomly picked from the system. This oracle models the scenario when $\mathcal{A}$ needs to get access to a tag that is anonymous to him. Otherwise $\mathcal{A}$ can always request access to a known ID tag using $\mathcal{T}_{ID}$ as the reference.

- `Free`($vtag$) invalidates the virtual reference to one of the system tags. The next call to `DrawTag`() may or may not return a virtual reference to the same tag again. This models the scenario when $\mathcal{A}$ loses contact with the accessing tag.

- `Exec`($vtag$) initialises the reader ready for a new protocol instance and returns π as the handle to it. Then it runs the full protocol faithfully until the end without any interruption with the tag $vtag$. The full transcript containing all the messages exchanged are given to $\mathcal{A}$. Notice that `Exec`($vtag$) may or may not be

executing a query with the correct ID of the $vtag$. i.e. the reader may issue a query for $\mathcal{T}_{ID}$ while the true ID of $vtag$ is ID', resulting a failed authentication. This models an adversary who eavesdrops on the communication between reader and tags of unknown IDs (the tag ID is part of the tag secrets).

- `Launch`() initialises the reader ready for a new protocol instance and returns π as the handle to it. The initial broadcast message m_π generated by the reader is given to $\mathcal{A}$. Calling `Launch`() does not guarantee the completion of the whole protocol if there is no subsequence calls to `SendTag`(.) and `SendReader`(.) to complete the protocol. This models an active adversary who records a valid reader verification message (in which the adversary cannot create by himself since he lacks the knowledge of the tag secrets) and possibly replays it at a later time using the following oracle. Again $\mathcal{A}$ has no information about which m_π is for which tag ID.

- `SendTag`$(vtag, m)$ sends $vtag$ a message m. The tag response m' (if there is any) is returned to $\mathcal{A}$. $vtag$ can be replaced by $\mathcal{T}_{ID}$ in case it is known to $\mathcal{A}$. Notice that the oracle does not require π to indicate which protocol instance as a tag can only handle one instance at a time.

- `SendReader`(π, m) sends the reader a message m for the protocol instance π. We allow the reader to have incomplete protocol instances withholding in its memory. i.e. the reader can handle multiple protocol instances at the same time without requiring any previously launched protocol instances to complete first. e.g. $\mathcal{A}$ can call `Launch`() twice to obtain both m_π and $m_{\pi+1}$. $\mathcal{A}$ can call `SendReader`(π, m) later to indicate m is for instance π even though the instance $\pi + 1$ has begun already. Since the reader will not give out any response in our **Type R** protocol structure, there will be no output for this oracle.

- `Result`(π) provides $\mathcal{A}$ with the authentication result of the protocol instance π. It outputs 1 if it was a success (a legitimate tag has been authenticated) or 0 if the authentication failed. If `SendReader`(π, m) is not called before `Result`(π), the output is $undefined$. Notice that in our protocol structure, the authentication result is not released and is known to the reader only. Hence this oracle models the real situations where the result can be obtained as side channel information, e.g. an electric door lock opened.

- `Corrupt`$(vtag)$ outputs the current tag key K_{vtag} of $vtag$ to $\mathcal{A}$. $vtag$ can be

replaced by $\mathcal{T}_{ID}$ in case it is known to $\mathcal{A}$. Notice that $\texttt{Corrupt}(vtag)$ can only be called before or after the protocol in our **Type R** protocol structure. i.e. corrupting a tag in between the first and second messages is not allowed [3]. Some may argue that if the tag corruption is done right after the tag has received the first message, not only the tag key but also all the other intermediate values can be revealed. Such data can be used to simulate the full memory state of a tag, which may or may not jeopardise the security of the system. We believe such powerful corruption is impractical to pull off because i.) it is difficult to predict what the current tag state is when the tag is performing computations. ii.) the tag will give out the response (second message) within a fraction of second, it is hard to freeze the tag within that short period of time. iii.) RFID tags are powered by the reader signal and the intermediate values are not stored in the permanent memory section like the tag key. Once the power source is lost, these values will be vanished. It will be very difficult for an adversary to maintain the power source while corrupting the tag.

The strongest adversary is defined to be the one who has no limitation on accessing all of these oracles except $\texttt{Corrupt}(vtag)$, where this restriction is imposed: $\texttt{Corrupt}(vtag)$ has to be the last oracle to access or in other words, once $\texttt{Corrupt}(vtag)$ is called, there can be no other subsequence oracle accesses [4]. This resembles the strongest adversary, the *wide-forward-attacker*, defined in [93] for RFID protocols using symmetric key only. According to [71], all the four construction types of SAP are not strong enough to stand against this adversary in the tag tracing attack. We are going to show that our **Type R** protocol construction remains secure in front of this attacker.

5.5.2 Privacy Experiment

Before analysing the security of our new protocol construction, we have to define the tag tracing privacy game first. Before the game starts, the system is properly setup by running $\texttt{SetupReader}(1^s)$ once and n times the $\texttt{SetupTag}(i, param, 1)$ where $|n|$ is the number of legitimate tags in the system. The tags are collected in a list $\mathcal{L}$, i.e. $\mathcal{T}_i \in \mathcal{L}$

[3] Corrupting a tag in between the protocol is allowed in other protocol structures like **Type 2a** and **Type 2b** because the reader verification (the third message) can be blocked such that the corruption made between the second and the third message is done while the protocol is not completed yet.

[4] Originally in [93], more calls to $\texttt{Corrupt}(.)$ are allowed as they consider also the RFID systems where tags have correlated secrets. We assume every tag has a unique tag key in our system model hence we think that more calls to $\texttt{Corrupt}(.)$ are redundant.

for $i = 1 \ldots n$.

The game is hosted by a challenger and runs in two stages. During the first stage, $\mathcal{A}$ is given access to $\mathcal{L}$ using any adversary oracle that is allowed within the ability of $\mathcal{A}$. Once $\mathcal{A}$ is satisfied interacting with the tags in $\mathcal{L}$ to obtain enough information about the tags, the first stage concludes. This is called the learning stage. $\mathcal{A}$ is now requested to submit an uncorrupted tag as the challenge tag $vtag_C$ to the challenger. $vtag_C$ is added into a separate list $\mathcal{C}$. Notice that in order for $\mathcal{A}$ to gain the most information, $\mathcal{A}$ will most likely not access $\texttt{Corrupt}(vtag_C)$ during this stage as there can be no other oracle access for $\mathcal{A}$ in the next stage once $\texttt{Corrupt}(vtag_C)$ is accessed.

In the second stage, $k \leq n-1$ uncorrupted tags are randomly picked from $\mathcal{L} - vtag_C$ and added into $\mathcal{C}$ as well. The $k+1$ tags in $\mathcal{C}$ are relabelled as $\mathcal{T}_{C_0}, \mathcal{T}_{C_1}, \ldots, \mathcal{T}_{C_k}$. Now $\mathcal{A}$ is allowed to make more oracle accesses to the tags in $\mathcal{C}$. Once $\mathcal{A}$ is satisfied interacting with these tags, the second stage concludes. This is called the guessing stage. $\mathcal{A}$ is now required to output a tag $\mathcal{T}_{C_b}$ where $b \in \{0 \ldots k\}$ that he thinks was $vtag_C$. The winning probability of a blind guessing $\mathcal{A}$ is $\frac{1}{k+1}$. Hence a secure protocol requires for any $\mathcal{A}$ to have gained only negligible advantage ϵ from the interaction with the system such that $\mathcal{A}$'s winning probability to the tag tracing game $\frac{1}{k+1} + \epsilon$ is only negligibly better than a blind guess.

5.6 Security Analysis of the Type R Construction

In this section, we will look at the improvements of the **Type R** protocol construction compares to the other four construction types. There are several trivial security improvements from the construction and we will first list them here.

5.6.1 Type R Protocols Can Resist Querying-Attack

We have this proposition: *If there exists no $\mathcal{A}$ who can forge a valid verification message, then a* **Type R** *protocol is safe from querying-attack.*

In order to launch a querying-attack, the adversary requires to send queries to a target tag repetitively such that the tag will keep changing its tag key to a new value until a point that the reader can no longer authenticate the tag. For $\mathcal{A}$ to successfully launch this attack on **Type R** protocols, he is required to generate valid verification messages per each query such that only when the message is verified will the tag changes its tag key. Since $\mathcal{A}$ lacks the knowledge of the tag key, which is needed to compute a correct verification message, the tag will not be made desynchronised in this way. This

is based on the assumption that $\mathcal{A}$ cannot forge a valid verification message.

Another way for $\mathcal{A}$ to launch a querying-attack is to get a valid verification message by calling `Launch`(), which returns the first message m_π of the reader to $\mathcal{A}$. Since m_π is a valid message of an unknown tag, $\mathcal{A}$ has to call `SendTag`$(vtag, m_\pi)$ for all the uncorrupted tags such that the matching tag $\mathcal{T}_i$ will change its tag key to a new value. To avoid synchronisation on the reader side, `SendReader`(π, m') is not called. Now to carry on the attack, $\mathcal{A}$ is tempted to resend m_π to $\mathcal{T}_i$. But since m_π was computed using the previous tag key of $\mathcal{T}_i$, $\mathcal{T}$ will not verify it and no tag key updates anymore. $\mathcal{A}$ may try to call `Launch`() again to get $m_{\pi+1}$. But as the previous tag response was not sent to $\mathcal{R}$, even $\mathcal{R}$ does not know $\mathcal{T}_i$ has updated its key and hence $m_{\pi+1}$ is computed using the previous tag key of $\mathcal{T}_i$ too, which will fail the verification. This implies that $\mathcal{A}$ can at most desynchronise the tag and reader one key state ahead on the tag side. Resynchronisation is easy, the reader just need to send two consecutive queries using the current and the next tag key to compute the verification messages when the tag wearer finds out the tag does not authenticate recently.

5.6.2 Type R Protocols Can Resist Blocking-Attack

A blocking-attack tries to desynchronise a tag and the reader by blocking the reception of the other party's response such that the uninformed party may not update its tag key. In the tradition three-round mutual authentication protocol, this attack can be easily launched. If the second message is always blocked, a **Type 2b** protocol will have the tag keeps changing its tag key while the reader is lagged behind. This will not happen in **Type R** protocols. Of course, it does not make sense to block the first message. So if the second message is blocked, as we have discussed above, it can at most desynchronise the tag and the reader one tag key state only.

In fact, even though $\mathcal{A}$ can desynchronise a tag one step ahead, $\mathcal{A}$ has no way to tell which tag has been desynchronised. Notice that the tag IDs are unknown to $\mathcal{A}$, so as the intended recipient of the first message m_π obtained from `Launch`() is also unknown to $\mathcal{A}$. If $\mathcal{A}$ broadcasts m_π to all the tags and forwards their responses to $\mathcal{R}$, the intended recipient $\mathcal{T}_i$ can be spotted by calling `Result`(π) as 1 will return. However, this means that $\mathcal{T}_i$ and $\mathcal{R}$ are synchronised. If the second message is blocked (i.e. `SendReader`(π, m') is not called), $\mathcal{A}$ cannot call `Result`(π) to test for the authentication result and $\mathcal{A}$ cannot identify the intended recipient $\mathcal{T}_i$.

Reader $\{K_{ID}\}$		Tag $\{\bar{K}_{ID}\}$
Picks a random c $v \leftarrow \mathcal{H}(K_{ID}\|\|c)$	$\xrightarrow{v,c}$	If $\mathcal{H}(\bar{K}_{ID}\|\|c) = v$ then picks a random n, $r \leftarrow \mathcal{H}(\bar{K}_{ID}\|\|c\|\|n)$
If $\mathcal{H}(K_{ID}\|\|c\|\|n) = r$ then $K'_{ID} \leftarrow \mathcal{H}(K_{ID})$	$\xleftarrow{r,n}$	else picks a random r $K'_{ID} \leftarrow \mathcal{H}(\bar{K}_{ID})$

Figure 5.7: A simple **Type R** RFID mutual authentication protocol

5.6.3 An Example Protocol

Let us give a simple example protocol that use the **Type R** protocol construction in figure 5.7.

Here $\mathcal{H}()$ is a one-way and collision free hash function and $||$ means concatenation. We have the following claim: *This simple protocol is strong against querying-attack, blocking-attack, forward-attack and a combination of them.*

Proof. First we show that the verification message v, c is unforgeable. Notice that v is computed from $\mathcal{H}(K_{ID}||c)$ and after v is verified, the tag key is updated to $K'_{ID} = \mathcal{H}(K_{ID})$. In order to forge a v', c' such that $v' = \mathcal{H}(K'_{ID}||c')$, either $\mathcal{A}$ has guessed the hash value $\mathcal{H}(K_{ID})$, which is negligible, or $\mathcal{A}$ has found a collision of $\mathcal{H}()$, which violates the assumption. Without a valid verification message, $\mathcal{A}$ cannot desynchronise the tag using querying-attack. Next, the resistance to blocking-attack is automatically provided by the protocol construction where desynchronisation stops right after the blocking of the second message, no subsequence blockings can cause the tag to update its key further.

Now, for the protection against forward-attack, the one-wayness property of $\mathcal{H}()$ provides unlinkability among changing tag keys in the hash-chain. On the other hand, this attack takes advantage from the static tag key by using it to decode any previously recorded communication between the tag and the reader. The only case where the tag remains with the same tag key is when the verification to v, c has failed. This can happen if v, c is a failed forging attempt of $\mathcal{A}$ or v, c is not intended for $\bar{K}_{ID}$. Either case, the tag will only output a random response, which cannot be used to related to any tag key. The last possibility is when the reader has been "desynchronised" after a blocking-attack (we are reluctant to call this a desynchronisation as the reader is only one tag state behind). As discussed in 5.6.2, even after $\mathcal{A}$ has desynchronised a victim tag, $\mathcal{A}$ has no way to tell which tag has been desynchronised. Hence there does

not exist two tag responses r, r' such that they are computed from the same tag key. Calling $\texttt{Corrupt}(.)$ does not help tracing the tag because there is no tag response that was computed using the updated tag key K'_{ID} for reference. □

Next, we have to recall the tag tracing game to proof the security of **Type R** protocol against wide-adversary, which is the adversary who can access the $\texttt{Result}(\pi)$ oracle to obtain the side-channel information of the authentication result of a tag [93]. We are aware that this extra piece of information may help the adversary to trace a tag. We have the following claim: *This simple protocol is not secure against wide-adversary in our tag tracing game for $k = 0$ and $k = 1$. The attack becomes infeasible when k is sufficiently large.*

Proof. It is trivial when $k = 0$, i.e. only the challenge tag $vtag_C$ is presented for $\mathcal{A}$ to guess. The mere fact that when $\mathcal{T}_{C_0}$ gives a response reveals its present and the winning probability of $\mathcal{A}$ is $\frac{1}{0+1}$. When $k = 1$, there are two tags presented for $\mathcal{A}$ to guess: the challenge tag and another random tag. It first seems that each of their response is independent to each other and appears to be random, the winning probability of $\mathcal{A}$ is no better than $\frac{1}{2}$. We proof with the following attack strategy where $\mathcal{A}$ will win with slightly better probability.

1. System starts up: $param \leftarrow \texttt{SetupReader}(1^s)$, $\mathcal{T}_0 \leftarrow \texttt{SetupTag}(0, param, 1)$, $\mathcal{T}_1 \leftarrow \texttt{SetupTag}(1, param, 1)$
2. $vtag \leftarrow \texttt{DrawTag}()$
3. $\{\pi, m_\pi\} \leftarrow \texttt{Exec}(vtag)$
4. $\theta \leftarrow \texttt{Result}(\pi)$
5. If $\theta = 1$ then quit; otherwise proceeds (m_π is a valid query for tag $\mathcal{T}_i \in \mathcal{L} - vtag$)
6. Concludes stage 1. Submit $vtag$ as the challenge tag
7. $\texttt{Free}(vtag)$
8. $vtag_1 \leftarrow \texttt{DrawTag}()$
9. $vtag_2 \leftarrow \texttt{DrawTag}()$
10. $m'_1 \leftarrow \texttt{SendTag}(vtag_1, m_\pi)$
11. $m'_2 \leftarrow \texttt{SendTag}(vtag_2, m_\pi)$

12. $\texttt{Free}(vtag_1)$
13. $\texttt{Free}(vtag_2)$
14. $\widehat{vtag} \leftarrow \texttt{DrawTag}()$
15. $\{\pi + 1, m_{\pi+1}\} \leftarrow \texttt{Exec}(\widehat{vtag})$
16. $\theta \leftarrow \texttt{Result}(\pi + 1)$
17. If $\theta = 0$ then goto 15; otherwise $\widehat{vtag}$ was $vtag$

The attack take advantage of the authentication result: if $\mathcal{T}_1$ was not authenticated using m_π, then m_π must be a valid query message for $\mathcal{T}_0$. Since $\mathcal{T}_0$ did not received m_π in the first stage, forwarding m_π to it in the second stage will desynchronise it. If the authentication result is 1 in the second stage, the querying tag must be $\mathcal{T}_1$, otherwise $\mathcal{A}$ has to call $\texttt{Exec}(\widehat{vtag})$ repeatedly until 1 is obtained. The winning probability of $\mathcal{A}$ is $\frac{1}{2} + \frac{1}{n \cdot k^2}$. Hence this tag tracing attack becomes infeasible when k is sufficiently large.

5.7 Example Applications for Type R Protocols

In contrast to the other SAP constructions where the reader is required to undergo full key space search to look for the matching tag key (and hence the tag ID), **Type R** protocol assumes the tag ID (and hence the corresponding tag key) is already determined at the beginning. When we have this assumption, there are a few identified applicable areas where this assumption is practical, otherwise this new protocol construction will not be of much use. We consider three different applications where this is possible.

Ownership transfer. When a buyer and a seller come to an agreement to transfer the ownership of an RFID tagged product, they must have verified and examined the product so that it is at its good state. In order to do so, they have to obtain the tag information before the ownership transfer is carried out, which also provide them with the tag ID and the current tag key of the tagged product. In that sense, the tag ID is known already at the beginning. **Type R** protocol can be used here to aid the ownership transfer by authenticating the validity of the tagged product, e.g. it is of the claimed tag ID associated with the claimed tag information and the seller actually knows the current tag key so that it does not appear to be a stolen product.

A practical use of **Type R** protocol for RFID ownership transfer is presented in [72].

Tag search. Instead of asking which tags are within the signal range of the reader, there can be other use of a tag scan, like asking if a particular tag is within the signal range. i.e. instead of issuing a query of "Who are there", the reader issues a query "Are tag_a, tag_b, tag_c, ... here?". In this case, the tag key of those particular tags are picked from the database and a specific tag search query is broadcasted to look for these tags. This is particularly useful in a warehouse to locate specific RFID tagged items when some signaling devices like alarms and buzz lights are attached to the source.

Tag query for small and large group. If there are only a small number of tags within the system, we can identify all of these tags by issuing them each a tag search query. The effect is the same as a broadcasted tag query message but provide better security protections as we have discussed. This is good for systems where there are not many legitimate tags but a higher standard of privacy protection is needed. On the other hand, **Type R** protocol provides better side-channel attack protection in a large group while it remains an open problem in other SAP constructions.

5.8 Conclusion

We have classified five types of SAPs in this chapter. Out of the five, four of them do not perform very well on privacy protection. We have shown that the fifth type of authentication protocol structure has its strength over other traditional construction types. It provides better protections against querying-attack, blocking-attack, forward-attack and any combination of them. The side-channel attack problem is also mitigated under this construction type. We also identified several potential application areas where they can benefit from this new tool.

Chapter 6

Ownership Transfer Protocol in RFID

Chapter Overview

In the previous chapter, we proposed a new classification for RFID protocols, called SAP. Since SAPs do not perform well in privacy protection, we suggested a new RFID protocols construction method and we proved its privacy protection abilities. In this chapter, we will introduce a new area in RFID protocols. We have seen mostly authentications and now we will look at ownership transfer. First we give a brief introduction to this topic, then we will go onto present our ownership transfer protocol. Our protocol is based on the new construction method we suggested in the previous chapter. Part of the results in this chapter has been published in [72] at RFIDsec Asia 2010 and in [73]. We have the following contributions in this chapter:

- We review some recently proposed works on RFID ownership transfer and identify their common security properties
- We introduce four new security properties for RFID ownership transfer
- We propose our new RFID ownership transfer protocol using our new construction method

6.1 RFID Ownership Transfer

Another aspect to look at the secret key issue (so as the privacy issue) of RFID is the possession of the key: i.e. who should know the secret key. The most used RFID system model consist of three components: a centralised trusted back-end database server, the RFID reader and the RFID tags. We refer to it the *centralised server model*. Under

this model, all the tag secret keys are stored and maintained in the back-end database server. Every RFID reader is assumed to have a secure connection to this server in order to access the tag secret keys. Every time a query is broadcast by the reader and some tag response is received, matching secret key will be fetched to properly resolve the response. If constantly changing secret key method is used, this may follow up with key update and synchronisation between server and the tag. This model implies that only the centralised server should possess all the secret keys. Of course, most likely the owner of the server would be the product manufacturer or the underlying company who runs the RFID application. This would not be a great privacy issue as long as the tags are still the company's assets.

In the future, smart home appliances will become more and more common. Personal RFID readers are expected to go into everyone's pocket and RFID tagged products will be all over the places. To be able to resolve the tag responses at home, personal RFID reader should also gain access to the matching secret key of the corresponding RFID tagged product. It is by no means impossible to require personal readers to connect to the product company server every time a query is needed, only less convenient, not to mention the heavy burden of the server to handle all the incoming requests. Also, there exists one major problem now: the trust issue. As the company is fully capable to read and scan all of its RFID tagged products because it possesses all the secret keys, it is not hard to see the consequence of using this centralised server model. Privacy of the product buyer (it is rather inappropriate to call him the product owner for now as we have not define what is an owner), especially his whereabouts, his trail, his usual places, etc. can be profiled by the company easily (refer to the battlefield example above). If we define the ownership of an RFID tag as the one who knows the tag secret key (we have a different definition in the context), then we see the need of *ownership transfer* in this scenario. It is not only an issue between the product company and the buyer, but also between buyers when the product changes hands.

6.2 Motivation

The aim of this work is to provide a practical and secure RFID ownership transfer scheme. Most of the previous RFID ownership transfer schemes that we reviewed in the literature are designed for a single-mined purpose: transfer the tag key from one owner to another owner securely. These limited schemes are not suitable in practice. In this work, we look into the privacy needs of every involving entities in an ownership

transfer: the company, the buyer, the previous owner and the new owner. To make our scheme practical, we also consider what will happen in front of a cheating seller in order to provide protections to the potential buyer. Or if the previous or new owner is later found out to be cheated, our scheme can also provides some protections to the victim. Besides ownership transfer, a subclass of it called *ownership delegation* is also considered in this work. Different from ownership transfer, where the ownership is fully transferred from one entity to another, delegation only transfer part of the ownership (this is essentially the right to read the tag) from the owner to a delegate. This delegated partial ownership will expire by itself or it can be explicitly taken back by the owner.

We will first have a brief review on what have been done in those previously proposed RFID ownership transfer schemes. Seeing what are missing in the literature, we show our contributions in the following subsection.

6.2.1 Previous Works on RFID Ownership Transfer

Compare to RFID authentication protocols, RFID ownership transfer schemes have received less attention in the literature. During our work, we can easily find a lot of works about the former, while only around ten pieces of work, to the best of our knowledge, are related to or have mentioned about the latter. We give a brief review of these works in this section.

Molnar et al. [67, 68] are the first to discuss RFID ownership transfer and ownership delegation explicitly along with their pseudonym RFID authentication protocol in their papers. *Ownership transfer* and *controlled delegation* are the new security properties they introduced for RFID applications. In their scheme, a trusted centre (TC) manages all the tag secrets in a tree structure. Each tag has one unique key and multiple shared keys with other tags to aid faster tag lookup. Pseudonyms are generated per each query using these keys such that only the TC can disambiguate tag responses and identify each tag. Controlled delegation is done by giving authorised reader a derived key, obtained by running a pseudo-random generator on input the unique key of a tag. The tag will use also the derived key in generating the next q pseudonyms as controlled by an internal non-volatile counter. Delegation expires automatically after q queries. Ownership transfer in fact is done with two controlled delegations. When a tag changes hands, the new owner requests delegation from the TC and asks for the remaining number of delegated tag queries of the previous owner (say p). The new owner then repeatedly queries the tag p times or send a new counter value to the tag

that is greater than the current counter value plus p. This prevents ownership overlap between the new and the previous owner. Fouladgar and Afifi use a similar setting as Molnar et al. in [29, 30, 31] where the role of TC is replaced by a centralised database (CDB). Each tag has an internal counter that increases per each query. Once this counter reaches its fixed maximum value, the current tag key will expire and the CDB must be contacted to renew the tag key. Delegation is done by releasing the current tag key to an verified user by the CDB. Ownership transfer is done by setting the tag counter to its maximum value first (to invalidate any delegation) and then renew the tag key, followed by a delegation to the new owner.

Since the TC or the CDB still holds all the tag secrets, tag queries made by future owners could still be monitored, which violates their privacy. Lim and Kwon [62] only consider these centralised management methods as temporary ownership transfer schemes and proposed *"perfect" ownership transfer*, which requires the previous owner to transfer all the tag secrets to the new owner and allow the new owner to secretly update them so that *new owner privacy* is preserved. Saito et al. have a similar idea in [80], however, the security of their scheme is only based on the short read range of the backward channel (tag to reader communication) by assuming that it is hard for adversaries to eavesdrop on this channel.

Instead of using a centralised server, Soppera and Burbridge [84] adopt the scheme of Molnar et al. by replacing the centralised TC with some distributed local devices called RFID acceptor tag such that delegations are done with them instead of the TC. Koralalage et al. [56] also suggest to use some key card reading devices to aid customers to directly overwrite the stored tag secret by swiping an universal customer card and inputting a PIN as the new tag secret. Both of these systems require the distribution of external devices, which adds extra cost and introduces new trust issues.

Previous owner privacy is another important security property in ownership transfer but it has not been addressed properly until Osaka et al. [76] proposed their scheme. In their scheme, both the previous and the new owner's privacy are preserved by allowing the previous owner to change the tag key first, then send this new (temporary) tag key to the new owner via a secure channel, and finally let the new owner to change this (temporary) tag key privately. This message flow pattern if designed correctly can protect both owners' privacy. However, a flaw in their ownership transfer protocol allows an attacker to break previous owner privacy if the tag is compromised, hence their scheme failed *forward security*. We adopt this message flow pattern in our scheme and at the same time we provide forward security.

Song [83] introduced a new property called *authorisation recovery*. In situations like after sales services or warranty purposes, a tag may be required to send back to its previous owner. For example a factory needs to access the tag to verify the product before providing any repairing service under warranty. This property ensures that ownership recovery is possible and does not involve another instance of ownership transfer between the current owner and the previous owner. The idea in [83] is fairly simple. The new owner just needs to record the (temporary) tag key given by the previous owner when ownership transfer was carried out. At times when authorisation recovery is needed, the current owner executes the key change protocol with the recorded tag key as input rather than using a random value. As the previous owner knew and recorded such key also, his/her authorisation to the tag is recovered. However, we see the way the author achieved this property as a side effect of running the key change protocol only. As a matter of fact, changing the tag key to some already known value only means sharing the ownership (if ownership is defined as possession of the tag key) with someone else. Although this authorisation recovery method saves the owners from needing to run the ownership transfer protocol (which is however a more proper way to do in our opinion), there is an unwanted effect that comes with it: the ownership to the tag becomes unclear now. If later on the tag is to be returned to the original owner, who is sharing the ownership with the previous (now current) owner, a new instance of ownership transfer protocol must be executed in order to fix the now unclear ownership. Dimitriou [25] also proposed a similar property called tag release where the current owner can issue a special command to let the tag restores back to its factory default key, which is always stored in the tag memory, allowing the manufacturer to gain back the access to the tag. But then again, to regain the authorisation, the original owner requires the manufacturer to delegate the updated tag key to him/her followed by a new instance of ownership transfer. One may think that a controlled delegation would probably fix this dilemma. But delegation requires a secure channel to deliver the delegated key, which does not enjoy the advantage of knowing a previously known shared secret (the temporary tag key used in previous ownership transfer). Hence we suggested a modified property called *temporary authorisation recovery*, which is a combination of controlled delegation and authorisation recovery that provides instant authorisation recovery to the previous owner and still maintains the full ownership to the tag by the original owner at the same time.

Recently, Deursen et al. [89] presented a formal model for RFID ownership transfer. They defined secure ownership and exclusive ownership where the former states that

the tag holder must be the tag owner and the latter states that there cannot be other tag owners beside the tag holder. However, they did not consider controlled delegation nor authorisation recovery where a tag holder may not be a tag owner and hence their model cannot be applied in our scheme as we provide both of these properties.

6.2.2 Our Contributions

In addition to the previously discussed RFID ownership transfer security properties: *controlled delegation, previous owner privacy, current owner privacy and temporary authorisation recovery*, there are some new security properties that are firstly introduced by us. We consider these new security properties as some practical needs for users during everyday RFID ownership transfer. To help illustrate our ideas, we have the following scenario in mind when we construct our scheme:

> *"Bob would like to buy an RFID tagged item currently owned by Alice. After agreeing on the price, they are about to begin the ownership transfer procedures. Alice first scans and authenticates the target item among all of her other RFID tagged products. The item is now taken out and isolated from the others. Alice changes the tag key of the item to a temporary value. Bob is now given the ID of the tag along with the item description and the temporary tag key. Although the item has been authenticated by Alice, from Bob's point of view, he cannot be sure about this, as the ID, the item description and the key are all provided by Alice. So instead of jumping right into the ownership transfer process, Bob may want to check on the item himself first. Bob may be able to verify that the tag and the item are not fabricated, but he cannot be sure if the ID truly belongs to the tag (i.e. same brand, same product, different item). Hence what he needs is tag assurance, to guarantee the tag is the same one as the one Alice has described. If Bob is satisfied, they can carry on the ownership transfer procedures. Now the money is paid and the ownership is transferred, but Bob immediately found out that the item is defective. Bob requests a refund but Alice now denies ever selling Bob such item. What Bob needs the most now is undeniable ownership transfer, where it provides a mean to prove Alice was the previous owner of the item. With such proof, Bob shows to the authorities that Alice was actually the one who sold him the item. Now Alice cannot deny the fact that she last owned the item, instead, now she*

claims that Bob stole the item from her. What can clear Bob's name is a current ownership proof to the item. Bob presented such proof and the case is adjourned."

With the scenario above, we propose a new RFID ownership transfer scheme that has all the security properties defined according to and adapted from previous schemes, these including : *controlled delegation*, *previous owner privacy*, *new owner privacy* and *temporary authorisation recovery*. Also, we introduce four new security properties for RFID ownership transfer. Some of them have been mentioned in the scenario, which are *tag assurance*, *undeniable ownership transfer* and *current ownership proof*. Furthermore, we provide *owner initiation* to guarantee all the reader-to-tag commands are executed only by the owner and never the delegate nor adversaries.

6.3 Preliminaries

In this section, we outline the models, assumptions, security definitions and the building blocks that are required to construct our scheme. We stress that readers should follow closely to our models and assumptions in this section before jumping to section 6.4, where our final scheme is presented.

6.3.1 System Model

We do not use the centralised server model in our scheme. Instead, we allow each user to have their own personal reader, which has their own personal database either connected wirelessly or built internally in the reader. This model removes the need of the trusted centner (TC) that is required in the centralised server model to maintain the current (sometimes also previous) ownership of each tag. From now on, we will simply refer to the combination of the reader and the connected database as *the reader*, since their connection is always considered to be secure in our system model. It is also assumed that the reader cannot be compromised (or in other words there is no advantage gained by compromising the reader) because in fact the database can be separated and located in a different physical location from the reader.

The manufacturer

To begin with, there is a special system role called *the manufacturer*, who is responsible to initialise the system, create and setup the tags. It is equipped with the following

functions:

- `SetupReader`() – initialises the system by inputting the security parameter 1^k and prepares the pre-defined tag authentication protocol `Auth`()
- `SetupTag`() – creates and setups a tag by inputting an unique ID along with the tag descriptions $Info_{ID}$. This outputs the corresponding tag secret K_{ID}
- `Auth`() – allows tag authentication to carry out between reader and tags

The manufacturer first executes `SetupReader`() with a security parameter 1^k to properly setup the reader and initialise the system to use `Auth`(). We define `Auth`() as one of those constantly changing symmetric key authentication protocols discussed above. Since the use of `Auth`() is not necessary in our scheme, we will skip the details about it, it is only mentioned for a complete description of the system model. After the setup of the reader, the manufacturer further creates and setups the tags by running `SetupTag`() with an unique ID together with some axillary tag related information $Info_{ID}$ (e.g. product description, origin, manufacture date, etc.) for each tag as input. This function outputs an unique tag secret K_{ID}, which is used as the initial tag key.

The RFID communication

Whenever a reader requires to authenticate a tag, it will execute the tag authentication protocol `Auth`() by first sending out a query and then relay the tag response to the database via a secure channel. After the database has processed the response, it will send back the result to the reader. Any user (including attackers) with a compatible reader can also setup their own reader by running `SetupReader`() using the public security parameters and start interacting with the tags, however, the user cannot access the other's database. Likewise, any user (mainly attackers) with a compatible tag can setup their own custom tag by running `SetupTag`() with some random or chosen ID, $Info_{ID}$ and start interacting with the other readers.

6.3.2 Ownership Transfer Model

In case of ownership transfer, there are new roles we refer to *the previous owner*, *the current owner*, *the potential owner* and *the delegate*. Every potential owner is equipped

with his/her own system compatible reader, together with his/her own database connected via a secure channel (personal readers may even have it installed internally). Each role have their own power to execute a certain functions as detailed below:

The current owner

The current owner is the basic role in the ownership transfer model. Originally, *the manufacturer* is the first and current owner of every RFID tag. Basically, the current owner is the one who has all the control over his/her own tag and is equipped with the following functions:

- `Auth`() – authenticates the owned tag
- `OwnerTrans`() – transfers the ownership to a new owner
- `KeyChange`() – changes the current tag key explicitly
- `Delegate`() – delegates the tag access right to a delegate
- `AuthRecover`() – temporary reverts the ownership to the previous owner
- `PreOwner`() – proves the ownership of the previous owner to the tag so that the previous owner cannot deny ever owning (selling) the tag
- `CurOwner`() – proves the ownership of the current owner to the tag so that any third party can be convinced that the tag was not stolen but actually owned by the current owner

`Auth`() is always used by the owner to access the tag, only the one who knows the current tag key can successfully execute this function. When an ownership transfer is required, the current owner and a potential owner will run our ownership transfer scheme `OwnerTrans`(). If it is a success, the roles will change: all the tag related secret and other information will be passed along to the potential owner, who becomes *the current owner*; the original owner now becomes *the previous owner*. `KeyChange`() can be run at any time to refresh the current tag key to some random value, which is useful to guarantee owner privacy. When it is needed, the owner can allow a delegate to gain temporary tag access right by executing `Delegate`(). Temporary authorisation recovery by running `AuthRecover`() is an added feature to the application at times like product maintenance to allow the previous owner to gain back the tag access right temporarily. Finally, `PreOwner`() and `CurOwner`() are the new features firstly provided by us and we will provide more details when we present our ownership transfer scheme.

The delegate

If the current owner executes the controlled delegation protocol `Delegate`(), there is an additional role called *the delegate*, who will receive the tag key and as a result, gains the access right to the tag. However, differ from the current owner, who has full control over the tag, the delegate can only authenticate the tag under a pre-defined number of times using the supplied tag key given by the current owner. For example, a shop keeper may delegate a worker to help stock taking, where tag reading is already enough to fulfill the task. Hence, the delegate can only execute the `Auth`() function temporarily:

- `Auth`() – authenticates the delegated tag

6.3.3 Basic Assumptions

As there can be different settings in the same system model, we have the following basic assumptions to characterise our model from the others. Our scheme and security proofs are also built upon these assumptions.

Capability assumption

We consider RFID tags as very constrained devices. They can at most perform some light-weight cryptographic hashing functions; on the contrary, readers are much more capable to perform more expensive cryptographic operations like asymmetric encryption and decryption, signing and signature verification.

Memory assumption

Tags are vulnerable to key compromise attack. We always assume all the internal secrets stored in tag memory are also available to competent adversaries. The base requirement of RFID tags is some *incorruptible memory* or *delicate memory*, i.e. adversaries can read the memory by compromising the tag but they lack the ability/tool to corrupt the memory or write back some chosen value. Even better is that once the tag is compromised, it will not be functioning anymore. The best they can do is to use the compromised memory content to create a clone by simulating the responses of the compromised tag. Whether this simulation or cloned tag can be caught is beyond the scope of our work. Hence we generalise this to an assumption *"once a tag is compromised, its memory can only be read and the tag no longer responses to*

other commands". This resemble the *forward attacker* as defined in [93], which is the strongest adversary definition for non-PKC capable RFID tags. On the contrary, tags are built with memory update mechanism but it only functions when the pre-defined protocols implemented in the tags are executed and followed faithfully.

Fixed target assumption

Unlike tag authentication protocols where the reader needs to search for the correct tag ID from its database by matching the tag response generated by the corresponding tag secret, we assume that in our ownership transfer scheme, there is always a target tag, which has been authenticated already, such that the reader knows exactly the ID of the tag and which tag secret K_{ID} to use to communicate with it. This assumption makes sense as both the seller and the buyer are trading a particular item they are both interested and selected. For this assumption to be applicable, we require the trading item to be authenticated first by `Auth`() and then separated from other RFID items so that it will be the sole item involved in the ownership transfer scheme before the scheme can be carried out.

Communication assumption

For the communication between reader and tag, we always assume that all the reader to tag messages can be delivered although these messages can still be eavesdropped, recorded and replayed by adversaries but are never blocked (notice that this does not mean all the reader to tag messages are originated from an honest reader, they can come from the adversaries or replays too). This assumption is logical since the reader always broadcasts strong wireless signals, which is hard to block. Also, due to the previous assumption, the intended recipient tag is always participating in the scheme, which eliminates the situation that the reader is broadcasting valid commands to a fake tag ONLY and resulted in simple record and replay (or relay) attack later on. On the other hand, this assumption can be easily removed if we require the tag to generate a random nonce for the reader first, and embed this nonce in the reader to tag message. Then the tag can verify the freshness of the message using the embedded nonce. Since this assumption is not too strong, we just leave it here to keep our scheme simple and avoid the necessity of adding a random number generator in a tag.

6.3.4 Adversary Model

We adopt the adversary model proposed by Vaudenay in [93] and simplify it with the following adversary abilities:

- `SetupReader`() – allows the creation of a fake reader to interact with other tags
- `SetupTag`() – allows the creation of a fake tag to interact with the reader
- `SendReader`() – sends a message to the reader. A reply message from the reader may be returned depending on the protocol
- `SendTag`() – sends a message to a tag. A reply message from this tag may be returned depending on the protocol
- `Corrupt`() – returns all the internal secrets stored inside the tag and virtually removes the tag from the application

We do not assume users are honest in our system, hence it is possible that either the previous owner, the current owner, the potential owner or the delegate is cheating in the scheme. However it is not realistic to consider when both sides are cheating (i.e. at most one adversary during any transaction), otherwise both can simply collogue and there can be no security property enforceable.

6.3.5 Security Properties

We identify the following security properties from previous RFID ownership transfer schemes:

- Previous owner privacy - At the completion of the ownership transfer scheme, the privacy of the previous owner is preserved. Meaning that no future owners can relate or trace back any previous communication between the previous owner and the RFID tag even though a full history of transmitted messages is eavesdropped and recorded.
- New owner privacy - At the completion of the ownership transfer scheme, the privacy of the new owner is preserved. Meaning that no previous owners can relate or track any current communication between the new owner and the RFID tag even though all the transmission is being eavesdropped.

- Controlled delegation - The current owner of the RFID tag has the authority to execute a delegation protocol, which temporary delegates the access right of the tag to anyone without forfeiting the ownership to the tag. The delegate cannot overtake the ownership while the owner can cancel this delegation at anytime. Moreover, the delegation will automatically expires once a pre-determined number of queries value is reached.

- Temporary authorisation recovery - The current owner of the RFID tag can allow the previous owner to gain back the access to the RFID tag without going through another instance of the ownership transfer protocol. At the same time, the current owner can cancel the recovered authorisation at anytime without the help from the previous owner.

We further introduce four new security properties firstly proposed in this work:

- Tag assurance - During the ownership transfer scheme, the buyer can be assured that the RFID tag undergoes the ownership transfer is the tag claimed by the current owner and requested by the buyer. This property guarantees that the current owner cannot randomly pick any tagged product he/she owns and sells it to the buyer. Together with the assumption 6.3.3, we provide in our scheme a way for the buyer to verify the ID of the tag.

- Current ownership proof - The current owner can prove to any third party that he/she is the current owner of the RFID tagged item.

- Undeniable ownership transfer - The current owner can prove to any third party that the RFID tagged item was owned by a previous owner and the previous owner cannot deny ever owning the tag.

- Owner initiation - The current owner and only the current owner can initiate an ownership transfer, key change and delegation. Unlike most of the other ownership transfer schemes where anyone who holds the current tag key can initiate an ownership transfer, we explicitly limit this to the current owner only (i.e. the delegate is excluded).

6.3.6 Building Blocks

To build our proposed scheme, we assume there exists a cryptographic hash function $\mathcal{H}() : \{0,1\}^* \rightarrow \{0,1\}^k$ that has the following properties:

- One-wayness - The computation of the hash value is efficient while it is hard to find the pre-image.
- Collision resistance - Given any hash value, it is hard to find another message not equal to the pre-image but gives the same hash value.

We also assume that there exists a public key cryptosystem (PKC) for the users to create publicly verifiable digital signatures such that for any given message m, a public key PK and a corresponding private key SK, we have

$$\sigma = \texttt{Sig}(m, SK) \text{ and } OK \leftarrow \texttt{Ver}(m, \sigma, PK)$$

where $\texttt{Sig}()$ is the signing operation that hash the input message m into proper length and outputs the signature σ signed with the private key SK on the hash of message. $\texttt{Ver}()$ is the signature verification operation that outputs OK if the signature is truly signed with the corresponding private key of the public key PK on m and outputs $\perp$ otherwise. We require that the signatures generated are unforgeable. As one may expected, the signature (together with the assumption 6.3.3) is used to provide current and undeniable ownership proofs.

The PKC is also capable to generate encrypted message from any given message m by an encryption function $\texttt{Enc}()$ using the public key PK and decrypt encrypted message by an decryption function $\texttt{Dec}()$ using the corresponding private key SK. i.e. we have

$$c = \texttt{Enc}(m, PK) \text{ and } m = \texttt{Dec}(c, SK)$$

These functions are only used to establish a secure channel to safely transfer the current tag key from the owner to the buyer. If there exists other form of secure channel (i.e. direct linkage between the readers of the owner and the buyer), these encryption and decryption functions are unnecessary.

Finally, as we mentioned in the assumption 6.3.3, there is a secure RFID authentication protocol $\texttt{Auth}()$ such that after its execution, it outputs $True$ if and only if the tag response r matches with the result generated using K_{ID}, otherwise it outputs $\perp$. Afterward, the real ID of the tag can be looked up by the reader using K_{ID} as the reference key from the database.

6.4 Our Ownership Transfer Scheme

We use the building blocks described in section 6.3.6 to construct our ownership transfer scheme. Our scheme composes of a setup and four protocols: key change protocol,

controlled delegation protocol, ownership transfer protocol and temporary authorisation recovery protocol. Each protocol has its own security goal to achieve. Notice that during the protocols, some messages are intended for the tag only (e.g. the commands) but we still use message flow arrows between the current owner and the potential owner/the delegate to indicate that such messages can always be overheard by the participating parties. We give details of our scheme below.

6.4.1 Setup

Before anyone can apply our scheme to aid RFID ownership transfer, users (including the manufacturer) are required to obtain their own public key PK and private key SK of the PKC. The manufacturer chooses a security parameter 1^k and runs `SetupReader`() to setup the reader and prepares the authentication protocol `Auth`() and the hash function $\mathcal{H}()$. The output bits of $\mathcal{H}()$ is set to $k-bits$. The manufacturer then chooses an unique ID for each tag and runs `SetupTag`(), which outputs a $k-bits$ random number K_{ID} as the initial tag key. For each of the tag entries, the reader records and maintains the following values:

- ID : The ID of the tag.
- $Info_{ID}$: The information about the tag.
- K_{ID} : The current tag key.
- $K_{H_0} = K_{ID}$: The tag session key used in generation of $\mathcal{O}$.
- $\sigma_0 \leftarrow \texttt{Sig}(V_{S_0}, SK_M)$: The signature of the manufacturer (first owner) for a tag signed using its private key SK_M. $V_{S_0} \leftarrow \mathcal{H}(ID||Info_{ID})$.

Each tag is then assigned the following values:

- K_{ID} : The symmetric key of the tag shared with its current owner.
- $V_{S_0} \leftarrow \mathcal{H}(ID||Info_{ID})$: The hash (chain) value of the tag ID and its information used in signature generation.
- $\mathcal{O} \leftarrow \mathcal{H}(\sigma_0||K_{H_0})$: The hash value of the current owner's signature.

Owner $\{K_{ID}, K_{H_i}, \sigma_i\}$		Tag $\{K_{ID}, \mathcal{O}\}$
$r \xleftarrow{R} \{0,1\}^k$, $T \leftarrow \mathcal{H}(r\|\|K_{ID}\|\|\mathbf{KC})$, $\mathcal{O} \leftarrow \mathcal{H}(\sigma_i\|\|K_{H_i})$, $u = \mathcal{O} \oplus T$, $K_{ID} = \mathcal{H}(r \oplus K_{ID})$	$\xrightarrow{\mathbf{KC},r,u}$	If $u \oplus \mathcal{H}(r\|\|K_{ID}\|\|\mathbf{KC}) \neq \mathcal{O}$, then $Fail$; Otherwise $K_{ID} = \mathcal{H}(r \oplus K_{ID})$

Figure 6.1: Key change protocol

6.4.2 Key Change Protocol

First of all, we present our key change protocol in figure 6.4.2. There are two main instances where this protocol should be executed, one before and one after the ownership transfer protocol. Changing the current tag key before the ownership transfer protocol can eliminate all the linkage of the previous communications between the current owner and the tag when the current tag key was used. This effectively provides *previous owner privacy*. Later when the ownership transfer protocol is completed, the new owner must change the tag key again such that the current tag key obtained from the previous owner can be overwritten with a fresh new key unknown to him/her. Since there is no secret shared between the tag and the new owner yet, it is unavoidable to preform such key change in a private environment free from the interception of the previous owner. This private key change effectively provides *new owner privacy*. The protocol is presented in figure 6.4.2. (notice that assumption 6.3.3 applies here). We will violate the notation a bit from now on and use bold letters to indicate the type of command being sent in the protocol. Here we have **KC** to indicate the command "Key Change".

6.4.3 Controlled Delegation Protocol

Next, we present our controlled delegation protocol. Using a similar idea in [29], a counter c is kept in the tag memory if the tag received a delegation command. Each time when the tag is queried the value will increase by 1. Once c reaches c_{max}, the delegation is automatically expired and the delegated key will be replaced with the original tag key that was backed up at the start of the delegation. There is also a delegation cancel protocol, which invalidates the delegated key despite the current value of c and restores the backed up key as the current tag key. This effectively provides *controlled delegation*. To complete the protocol, the current owner has to send the delegated key to the delegate via a secure channel. In our setting, the public key of the delegate can be used to encrypt the key in a secure manner thanks to the

Controlled delegation:		
OWNER $\{K_{ID}, K_{H_i}, \sigma_i, PK_D\}$	TAG $\{K_{ID}, \mathcal{O}\}$	DELEGATE $\{SK_D\}$
$r \xleftarrow{R} \{0,1\}^k$, pick max, $T \leftarrow \mathcal{H}(r\|\|K_{ID}\|\|max\|\|\mathbf{CD})$, $\mathcal{O} \leftarrow \mathcal{H}(\sigma_i\|\|K_{H_i})$, $u = \mathcal{O} \oplus T$, $K_D \leftarrow \mathcal{H}(r \oplus K_{ID})$, $e \leftarrow \mathtt{Enc}(K_D, PK_D)$	$\xrightarrow{\mathbf{CD}, max, r, u, e}$ $m = r\|\|K_{ID}\|\|max\|\|\mathbf{CD}$, if $u \oplus \mathcal{H}(m) \neq \mathcal{O}$, then $Fail$; Otherwise $K_B = K_{ID}$, $c_{max} = max$, $c = 0$, $K_{ID} = \mathcal{H}(r \oplus K_{ID})$	$K_D \leftarrow \mathtt{Dec}(e, SK_D)$, $c_{max} = max$, $c = 0$
Subsequent tag queries:		
DELEGATE $\{K_D, c, c_{max}\}$		TAG $\{K_{ID}, c, c_{max}, K_B\}$
If $c = c_{max}$, then $Fail$; Otherwise $c = c + 1$, executes $\mathtt{Auth}(K_D)$	$\xrightarrow{Query}$ $\xleftarrow{Response}$	Executes $\mathtt{Auth}(K_{ID})$, if $c < c_{max}$, then $c = c + 1$, if $c = c_{max}$, then $K_{ID} = K_B$, $K_B = 0^k$
Delegation cancel:		
OWNER $\{K_{ID}, K_{H_i}, \sigma_i\}$		TAG $\{K_{ID}, \mathcal{O}, c, c_{max}, K_B\}$
$r \xleftarrow{R} \{0,1\}^k$, $T \leftarrow \mathcal{H}(r\|\|K_{ID}\|\|\mathbf{DC})$, $\mathcal{O} \leftarrow \mathcal{H}(\sigma_i\|\|K_{H_i})$, $u = \mathcal{O} \oplus T$	$\xrightarrow{\mathbf{DC}, r, u}$	If $u \oplus \mathcal{H}(r\|\|K_B\|\|\mathbf{DC}) \neq \mathcal{O}$, then $Fail$; Otherwise $K_{ID} = K_B$, $K_B = 0^k$

Figure 6.2: Controlled delegation protocol

PKC. The protocol is presented in figure 6.2. Notice that when the delegated key is replaced by the backed up key at the end of delegation, the backed up key is zeroed out with k 0-bits to clear any possible trace of the old tag key (in case the tag is compromised).

6.4.4 Ownership Transfer Protocol

Following the assumption in 6.3.3, an intended RFID item has already been authenticated using `Auth`() and singulated from other RFID items. Its ID and $Info_{ID}$ are obtained and its corresponding tag key K_{ID} is selected. Before the protocol begins, the owner will forward the ID and $Info_{ID}$ to the buyer (notice that the buyer can only verify the validity of ID and $Info_{ID}$ until phase 5.). They also exchange their public keys PK_O and PK_B, allowing the other party to verify the validity of the public key with the PKC before actually starting the ownership transfer protocol. Our ownership transfer protocol contains several phases. One nice feature of this is that users can cancel the ownership transfer at any phase without sabotaging the security of the whole system. The first three phases are in fact the key change protocol, controlled delegation protocol and an execution of `Auth`(). At the end of the protocol, the new owner should execute the key change protocol in a private environment. The protocol

is presented in figure 6.3.

6.4.5 Temporary Authorisation Recovery Protocol

This protocol is very similar to the controlled delegate protocol as the previous owner can be viewed as a delegate. Instead of using a new random tag key, the current owner can take advantage by making use the previously known secret shared among him/her and the previous owner, which was the temporary tag key K_P being used in the last ownership transfer protocol. This way the current owner is saved from contacting the previous owner to execute an ownership transfer protocol or from sending the previous owner the new delegated tag key via a secure channel in order to recover the previous owner's authorisation to the tag. Comparing to [83] and [25] where the ownership will be taken by the previous owner once authorisation recovery is executed, our scheme allows the current owner to regain the ownership by executing the delegation cancel protocol without going through another ownership transfer instance with the previous owner. We present our protocol in figure 6.4.

6.5 Security Analysis

6.5.1 Previous Owner Privacy and New Owner Privacy

We have already mentioned about the security properties *previous owner privacy* and *new owner privacy*, which are achieved by the key change protocol described in section 6.4.2. By running the key change protocol before and (secretly) at the end of the ownership transfer protocol, any trace of the previous tag key is eliminated thanks to the one-wayness property of the hash function $\mathcal{H}()$. We prove this by contradiction: suppose there is an attacker who can output the previous tag key $K_{ID_{i-1}}$ given the current tag key K_{ID_i} as input (i.e. it is a forward security attacker who compromises the memory of the tag to extract the current tag key), one can use this attacker to find the pre-image of K_{ID_i} in $\mathcal{H}()$ by computing $r \oplus K_{ID_{i-1}}$, where r was the random number used in the last instance of the key change protocol sent in plaintext. This contradicts the assumption that finding the pre-image of a hash value is hard under the one-wayness property. Hence either the output of the previous tag key $K_{ID_{i-1}}$ is only a blind guess (which only has negligible probability 2^{-k} to be a correct guess) or the attacker knows the previous tag key from other source. There are two cases for the attacker to obtain the previous tag key: i.) by compromising the tag before the key

Owner	Tag	Buyer
$\{PK_O, SK_O, PK_B, K_{ID}, K_{H_i}, \sigma_i\}$	$\{K_{ID}, V_{S_i}, \mathcal{O}\}$	$\{PK_B, SK_B, PK_O, ID, Info_{ID}\}$
Phase 1. Key change		
$r \xleftarrow{R} \{0,1\}^k$, $T \leftarrow \mathcal{H}(r\|\|K_{ID}\|\|\mathbf{KC})$, $\mathcal{O} \leftarrow \mathcal{H}(\sigma_i\|\|K_{H_i})$, $u = \mathcal{O} \oplus T$, $K_{ID} = \mathcal{H}(r \oplus K_{ID})$	$\xrightarrow{\mathbf{KC}, r, u}$ $m = r\|\|K_{ID}\|\|\mathbf{KC}$, If $u \oplus \mathcal{H}(m) \neq \mathcal{O}$, then *Quit*; Otherwise $K_{ID} = \mathcal{H}(r \oplus K_{ID})$	
Phase 2. Delegation		
$r \xleftarrow{R} \{0,1\}^k$, $max = 1$, $T \leftarrow \mathcal{H}(r\|\|K_{ID}\|\|1\|\|\mathbf{CD})$, $\mathcal{O} \leftarrow \mathcal{H}(\sigma_i\|\|K_{H_i})$, $u = \mathcal{O} \oplus T$, $K_{H_{i+1}} \leftarrow \mathcal{H}(r \oplus K_{ID})$, $e \leftarrow \mathtt{Enc}(K_{H_{i+1}}, PK_B)$	$\xrightarrow{\mathbf{CD}, 1, r, u, e}$ $m = r\|\|K_{ID}\|\|1\|\|\mathbf{CD}$, if $u \oplus \mathcal{H}(m) \neq \mathcal{O}$, then *Quit*; Otherwise $K_B = K_{ID}$, $c_{max} = 1$, $c = 0$, $K_{ID} = \mathcal{H}(r \oplus K_{ID})$	$K_{H_{i+1}} \leftarrow \mathtt{Dec}(e, SK_B)$
Phase 3. Authentication		
	$\xleftarrow{Query}$ $\xrightarrow{Response}$ Executes $\mathtt{Auth}(K_{ID})$, $K_{ID} = K_B$	Executes $\mathtt{Auth}(K_{H_{i+1}})$, if returns $\perp$, then *Quit*; Otherwise proceed
Phase 4. Ownership transfer starts		
$r \xleftarrow{R} \{0,1\}^k$, $T \leftarrow \mathcal{H}(r\|\|K_{ID}\|\|\mathbf{TS})$, $\mathcal{O} \leftarrow \mathcal{H}(\sigma_i\|\|K_{H_i})$, $u = \mathcal{O} \oplus T$	$\xrightarrow{\mathbf{TS}, r, u}$ $m = r\|\|K_{ID}\|\|\mathbf{TS}$, if $u \oplus \mathcal{H}(m) \neq \mathcal{O}$, then *Quit*; Otherwise proceed	
Phase 5. Tag assurance		
$V_{S_{i+1}} \leftarrow \mathcal{H}(V_{S_i})$	$\xleftrightarrow{\mathbf{TA}, V_{S_i}}$	Let $V_{S_0} = \mathcal{H}(ID\|\|Info_{ID})$, for $j = 0 \ldots n$, $V_{S_{j+1}} \leftarrow \mathcal{H}(V_{S_j})$, until $V_{S_j} = V_{S_i}$ If not found, then *Quit*; Otherwise proceed
Phase 6. Buyer signature verification		
If $\mathtt{Ver}(V_{S_{i+1}}, \sigma_{i+1}, PK_B)$ returns $\perp$, then *Quit*; Otherwise proceed	$\xleftarrow{\mathbf{VR}, \sigma_{i+1}}$ Stores σ_{i+1}	$\sigma_{i+1} \leftarrow \mathtt{Sig}(V_{S_{i+1}}, SK_B)$
Phase 7. Ownership transfer ends		
$K = K_{ID} \oplus K_{H_{i+1}}$, $r \xleftarrow{R} \{0,1\}^k$, $T \leftarrow \mathcal{H}(r\|\|K_{ID}\|\|\mathbf{TE})$, $\mathcal{O} \leftarrow \mathcal{H}(\sigma_i\|\|K_{H_i})$, $u = \mathcal{O} \oplus T$, records K_{ID} as K_P	$\xrightarrow{\mathbf{TE}, r, u, \sigma_i, K}$ $m = r\|\|K_{ID}\|\|\mathbf{TE}$, if $u \oplus \mathcal{H}(m) \neq \mathcal{O}$, then *Fail*; Otherwise $V_{S_i} = \mathcal{H}(V_{S_i})$, $\mathcal{O} = \mathcal{H}(\sigma_{i+1}\|\|K_{H_{i+1}})$	If $\mathtt{Ver}(V_{S_i}, \sigma_i, PK_O)$ returns $\perp$, then *Fail*; Otherwise $K_{ID} = K \oplus K_{H_{i+1}}$, records $\sigma_{i+1}, K_{H_{i+1}}$, K_{ID} as K_P

Figure 6.3: Ownership transfer protocol

Temporary authorisation recovery:		
OWNER $\{K_{ID}, K_{H_i}, \sigma_i, K_P\}$	TAG $\{K_{ID}, \mathcal{O}\}$	PREVIOUS OWNER $\{K_P\}$
$r \xleftarrow{R} \{0,1\}^k$, $T \leftarrow \mathcal{H}(r\|\|K_{ID}\|\|\mathbf{AR})$, $\mathcal{O} \leftarrow \mathcal{H}(\sigma_i\|\|K_{H_i})$, $u = \mathcal{O} \oplus T$, $K_D \leftarrow \mathcal{H}(r \oplus K_{ID})$, $e = K_D \oplus K_P$	$\xrightarrow{\mathbf{AR},r,u,e}$ $m = r\|\|K_{ID}\|\|\mathbf{AR}$, if $u \oplus \mathcal{H}(m) \neq \mathcal{O}$, then $Fail$; Otherwise $K_B = K_{ID}$, $K_{ID} = \mathcal{H}(r \oplus K_{ID}) \oplus e$	$K_{ID} = K_P$
Authorisation taken back (same as delegation cancel):		
OWNER $\{K_{ID}, K_{H_i}, \sigma_i\}$		TAG $\{K_{ID}, \mathcal{O}, c, c_{max}, K_B\}$
$r \xleftarrow{R} \{0,1\}^k$, $T \leftarrow \mathcal{H}(r\|\|K_{ID}\|\|\mathbf{DC})$, $\mathcal{O} \leftarrow \mathcal{H}(\sigma_i\|\|K_{H_i})$, $u = \mathcal{O} \oplus T$	$\xrightarrow{\mathbf{DC},r,u}$	If $u \oplus \mathcal{H}(r\|\|K_B\|\|\mathbf{DC}) \neq \mathcal{O}$, then $Fail$; Otherwise $K_{ID} = K_B$, $K_B = 0^k$

Figure 6.4: Temporary authorisation recovery protocol

exchange protocol was carried out. However, this violates the assumption 6.3.3 that once a tag is compromised, it is not functioning anymore and would not have completed the key exchange protocol. ii.) the attacker is the previous owner who always know the previous tag key. Since the previous owner will not attack his own privacy, he will only target on attacking the new owner privacy. Hence the only fix to this is to require the new owner to carry out the key change protocol in a private environment away from the interception of the previous owner, such that the random number r becomes a secret added into the computation of the new tag key. Guessing r would take the same effort as guessing K_{ID_i} as they are both k-bits.

6.5.2 Controlled Delegation and Temporary Authorisation Recovery

Since these two properties are more like security features rather than security protections, it is trivial enough to verify their correctness from the protocol description. The only thing to keep in mind is that the delegated key computation is the same as the new tag key computation (i.e. $\mathcal{H}(r \oplus K_{ID})$), one should not reuse the same random number r for the key change protocol after the controlled delegation protocol. Otherwise the delegate can instantly obtain the new tag key, which was in fact the delegated key he received before. Also, notice that as long as the current tag key does not change, the delegation message $\mathbf{CD}, max, r, u, e$ and the delegation cancel message $\mathbf{DC}, r, u$ can be replayed. e.g. the delegate may want to gain additional access to the tag after the first controlled delegation has expired. Hence one may want to execute the key change

protocol to renew the tag key after a delegation has expired.

6.5.3 Tag Assurance

Tag assurance is guaranteed in phase 3. and 5. of the ownership transfer protocol. In most of the previous ownership transfer schemes, the buyer can only choose to believe the RFID tagged item presented by the current owner is the item he/she wants and not something else (consider a cheating owner who swapped the trading item with something else that looks similar to the original item but at a lower quality). In phase 3. of our protocol, it allows the buyer to make sure the owner actually knows the tag key of the trading item. This avoids someone trying to sell stolen goods. Next in phase 5. by verifying the hash chain value V_{S_i} generated from $ID, Info_{ID}$ gives the buyer confidence on the true identity of the tag (under the assumption 6.3.3). Together they guarantee to the buyer that the owner owns the item and the information $ID, Info_{ID}$ supplied by the owner is the correct description of the item. Thanks to the collision resistance property of $\mathcal{H}()$, it is hard for the owner to find another message/pre-image $ID', Info'_{ID'}$ (to replace the description of the swapped lower quality item with some exaggerated information) such that it gives the same hash chain value V_{S_i} after hashing it several times with $\mathcal{H}()$ provided that n (the maximum acceptable number of previous owners/number of hash chains) is reasonably small. Again, we prove this by contradiction: suppose there is an attacker who can output a fake description $ID', Info'_{ID'}$ of the trading item by inputting a hash chain value V_{S_i}, where $i <= n$. One can use this attacker to find a collision in $\mathcal{H}()$. Let $ID, Info_{ID}$ be the original message and $V_{S_j} = V_{S_i}$ is the hash chain value of it under $\mathcal{H}()$ where $j <= n$, then the collision is $V_{S_{j-1}}$ and $V_{S_{i-1}}$. This contradicts the assumption that finding a collision in a hash function is hard under the collision resistance property. Hence either $ID', Info'_{ID'}$ is in fact the correct description of the item (i.e. $ID, Info_{ID}$) or V_{S_i} must be fake as well. There can be two cases: i.) the attacker has overwritten the hash chain value stored in the tag with V_{S_i}. However, this violates the assumption 6.3.3 that the tag has incorruptible memory. ii.) the whole tag is a fake tag created by the attacker by running `SetupTag`(ID'). Whether a fake tag can be spotted or not is beyond our scope.

6.5.4 Current Ownership Proof and Undeniable Ownership Transfer

Tag ownership cannot be defined simply as the one who holds the tag or someone who knows the tag key, especially when delegation is implemented. Our scheme requires a tag to store the value $\mathcal{O}$, which is the hash value of the owner's signature. Hence ownership in our scheme is defined as someone who knows both the pre-image of this hash value and the current tag key. Since the pre-image is a signature, it can be tightly bound to the owner as he/she is the only one who can generate such signature. To bind the owner to the tag, the message signed is the hash chain value V_{S_i}. To prove previous ownership and current ownership, it is suffices (together with the assumption 6.3.3) to present V_{S_i}, σ_i and $V_{S_{i+1}}, \sigma_{i+1}$ to any third party. One cannot deny ever created the signatures and hence they become the evidence of ownership transfer and the proof of current ownership.

6.5.5 Owner Initiation

Since both the current owner and the delegate may hold the current tag key, the owner must possess some additional secret to distinguish the owner's role from the delegate's role, so that only the owner can issue commands to the tag but not the delegate. We use the hash value of the owner's signature $\mathcal{O}$ as the additional key to initiate tag commands. Notice that in each of the commands of the three protocols in our scheme, the owner is required to compute $\mathcal{O} \leftarrow \mathcal{H}(\sigma_i||K_{H_i})$ and $T \leftarrow \mathcal{H}(r||K_{ID}||\mathbf{COMMAND})$. $\mathcal{O}$ remains the same throughout the ownership of the same owner, while T changes every time when the current tag key changes and its freshness is guaranteed by the random number r. As σ_i is sent in plaintext in the ownership transfer protocol, the secrecy of $\mathcal{O}$ is protected by K_{H_i}, the delegated key sent via a secure channel to the current owner when ownership is transferred. Hence to break owner initiation, one must obtain K_{H_i}, which is impossible because it is sent in a secure channel, or $\mathcal{O}$. To obtain $\mathcal{O}$, one may guess on the value of K_{H_i}, which has negligible success probability. One may also compromise the tag as the tag stores $\mathcal{O}$. But once the tag is compromised, it will be virtually dead and rendered the acquisition of $\mathcal{O}$ useless. Otherwise one may try to compute $\mathcal{O}$ from $\mathcal{O} = u \oplus T$. But to compute T, the knowledge of the current tag key K_{ID} is required. At the end, we have guaranteed owner initiation.

6.6 Conclusions

We presented a new RFID ownership transfer scheme in this chapter. Our scheme consists of four protocols: key change protocol, delegation protocol, ownership transfer protocol and temporary authorisation recovery protocol. Our scheme combines these four protocols to provide a secure method for users to transfer their RFID tags to new hands. We also considered some practical needs users may request in ownership transfer. For example, we have tag assurance to deal with cheating sellers. Current ownership proof creates a tight binding between the current owner and the tag. Undeniable ownership transfer is aimed to handle dispute that may occur when the previous owner denies selling a faulty item to the current owner. Owner initiation guarantees only the current owner can give various commands to the tag. We believe this will open up new research directions in this area and allow more new ideas to come and strengthen the development of RFID applications.

Chapter 7

Conclusion

In this whole thesis, we have studied the privacy issues surrounding low-cost RFID tag systems. From RFID authentication protocols to privacy modellings and then from protocol construction classifications to practical ownership transfer protocol. This whole RFID privacy journey give us the impression that we cannot expect a overly constrained device to attain a high level of privacy standard like other more decent devices. Although through model tweaking, one can prove a better system on its privacy performance, but this either has additional assumptions on the system or the adversary is being restricted. We have been using the Vaudenay model in most of our works and we believe this is a very good model to be used on RFID. However, the impossible results tell us that RFID has its limitations at its current form and hence, we have found our solution by proposing the **Type R** construction. This is a way out because it changes the current form of RFID protocols. With the advance in RFID technology, we hope that someday these tiny RFID devices can find its position in our everyday life, without any privacy concerns.

Bibliography

[1] Ross Anderson and Markus Kuhn. Low Cost Attacks on Tamper Resistant Devices. In Bruce Christianson, Bruno Crispo, Mark Lomas, and Michael Roe, editors, *The Fifth International Workshop on Security Protocols – IWSP 1997*, volume 1361 of *Lecture Notes in Computer Science*, pages 125–136, Paris, France, April 1997. Springer.

[2] Frederik Armknecht, Ahmad-Reza Sadeghi, Alessandra Scafuro, Ivan Visconti, and Christian Wachsmann. Impossibility Results for RFID Privacy Notions. In Marina L. Gavrilova, C. J. Kenneth Tan, and Edward David Moreno, editors, *Transaction on Computational Science XI – Special Issue on Security in Computing, Part II*, volume 6480 of *Lecture Notes in Computer Science*, pages 39–63. Springer, October 2010.

[3] Frederik Armknecht, Ahmad-Reza Sadeghi, Ivan Visconti, and Christian Wachsmann. On RFID Privacy with Mutual Authentication and Tag Corruption. In Jianying Zhou and Moti Yung, editors, *The Eighth International Conference on Applied Cryptography and Network Security – ACNS 2010*, volume 6123 of *Lecture Notes in Computer Science*, pages 493–510, Beijing, China, June 2010. International Communications and Information Security Association (ICISA), Springer.

[4] Gildas Avoine. Privacy Issues in RFID Banknote Protection Schemes. In Jean-Jacques Quisquater, Pierre Paradinas, Yves Deswarte, and Anas Abou El Kalam, editors, *The Sixth International Conference on Smart Card Research and Advanced Applications – CARDIS 2004*, volume 153 of *Advances in Information and Communication Technology (AICT)*, pages 33–48, Toulouse, France, August 2004. International Federation for Information Processing (IFIP), Kluwer Academic Publishers.

[5] Gildas Avoine. Adversarial Model for Radio Frequency Identification. Cryptology ePrint Archive, Report 2005/049, February 2005.

[6] Gildas Avoine. Radio Frequency Identification: Adversary Model and Attacks on Existing Protocols. Technical Report LASEC-REPORT-2005-001, Swiss Federal Institute of Technology (EPFL), Security and Cryptography Laboratory (LASEC), Lausanne, Switzerland, September 2005.

[7] Gildas Avoine, Iwen Coisel, and Tania Martin. Time Measurement Threatens Privacy-Friendly RFID Authentication Protocols. In Siddika Berna Ors Yalcin, editor, *The Sixth International Workshop on Radio Frequency Identification: Security and Privacy Issues – RFIDsec 2010*, volume 6370 of *Lecture Notes in Computer Science*, pages 138–157, Istanbul, Turkey, June 2010. Springer.

[8] Gildas Avoine, Etienne Dysli, and Philippe Oechslin. Reducing Time Complexity in RFID Systems. In Bart Preneel and Stafford Tavares, editors, *The Twelfth International Workshop on Selected Areas in Cryptography – SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 291–306, Kingston, Ontario, Canada, August 2005. Springer.

[9] Gildas Avoine, Cédric Lauradoux, and Tania Martin. When Compromised Readers Meet RFID. In Heung Youl Youm and Moti Yung, editors, *The Tenth International Workshop on Information Security Applications – WISA 2009*, volume 5932 of *Lecture Notes in Computer Science*, pages 36–50, Busan, Korea, August 2009. Korea Institute of Information Security and Cryptology (KIISC), Springer.

[10] Gildas Avoine and Philippe Oechslin. A Scalable and Provably Secure Hash-Based RFID Protocol. In *The Second IEEE International Workshop on Pervasive Computing and Communication Security – PerSec 2005*, pages 110–114, Kauai island, Hawaii, USA, March 2005. Institute of Electrical and Electronics Engineers (IEEE), IEEE Computer Society Press.

[11] Gildas Avoine and Philippe Oechslin. RFID Traceability: A Multilayer Problem. In Andrew S. Patrick and Moti Yung, editors, *The Ninth International Conference on Financial Cryptography and Data Security – FC 2005*, volume 3570 of *Lecture Notes in Computer Science*, pages 125–140, Roseau, The Commonwealth Of Dominica, February–March 2005. International Financial Cryptography Association (IFCA), Springer.

[12] Leonid Bolotnyy and Gabriel Robins. Physically Unclonable Function-Based Security and Privacy in RFID Systems. In *The Fifth Annual IEEE International*

Conference on Pervasive Computing and Communications – PerCom 2007, pages 211–220, White Plains, New York, USA, March 2007. Institute of Electrical and Electronics Engineers (IEEE), IEEE Computer Society.

[13] Stefan Brands and David Chaum. Distance-Bounding Protocols (Extended Abstract). In Tor Helleseth, editor, *The Twelfth Annual International Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology – EUROCRYPT 1993*, volume 765 of *Lecture Notes in Computer Science*, pages 344–359, Lofthus, Norway, May 1993. International Association for Cryptologic Research (IACR), Springer.

[14] Mike Burmester and Breno de Medeiros. RFID Security: Attacks, Countermeasures and Challenges. In *The Fifth RFID Academic Convocation – RFID Journal LIVE! 2007*, Orlando, Florida, USA, April–May 2007. RFID Journal.

[15] Mike Burmester, Breno de Medeiros, and Rossana Motta. Robust, Anonymous RFID Authentication with Constant Key-Lookup. In Masayuki Abe and Virgil D. Gligor, editors, *The Third ACM Symposium on Information, Computer and Communications Security – ASIACCS 2008*, pages 283–291, Tokyo, Japan, March 2008. Association for Computing Machinery (ACM), ACM Press.

[16] Mike Burmester, Tri Van Le, Breno de Medeiros, and Gene Tsudik. Universally Composable RFID Identification and Authentication Protocols. *ACM Transactions on Information and System Security – TISSEC 2009*, 12(4):21:1–21:33, April 2009.

[17] Mike Burmester, Tri van Le, and Breno de Medeiros. Provably Secure Ubiquitous Systems: Universally Composable RFID Authentication Protocols. In *The Second International Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm 2006*, Baltimore, Maryland, USA, August–September 2006. IEEE Communications Society/Create-Net.

[18] Sébastien Canard and Iwen Coisel. Data Synchronization in Privacy-Preserving RFID Authentication Schemes. In *The Fourth International Workshop on Radio Frequency Identification: Security and Privacy Issues – RFIDsec 2008*, Budapest, Hungary, July 2008. Institute for Applied Information Processing and Communications (IAIK) TU Graz.

[19] Hung-Yu Chien and Chen-Wei Huang. A Lightweight RFID Protocol Using Substring. In Tei-Wei Kuo, Edwin Hsing-Mean Sha, Minyi Guo, Laurence Tianruo Yang, and Zili Shao, editors, *The Fifth International Conference on Embedded and Ubiquitous Computing – EUC 2007*, volume 4808 of *Lecture Notes in Computer Science*, pages 422–431, Taipei, Taiwan, December 2007. International Federation for Information Processing (IFIP), Springer.

[20] Ivan Damgård and Michael Østergaard Pedersen. RFID Security: Tradeoffs between Security and Efficiency. Cryptology ePrint Archive, Report 2006/234, July 2006.

[21] Ivan Damgård and Michael Østergaard Pedersen. RFID Security: Tradeoffs between Security and Efficiency. In Tal Malkin, editor, *The Cryptographers' Track at the RSA Conference on Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 318–332, San Francisco, California, USA, April 2008. RSA, Springer.

[22] Gerhard de Koning Gans and Flavio D. Garcia. Towards a Practical Solution to the RFID Desynchronization Problem. In Siddika Berna Ors Yalcin, editor, *The Sixth International Workshop on Radio Frequency Identification: Security and Privacy Issues – RFIDsec 2010*, volume 6370 of *Lecture Notes in Computer Science*, pages 203–219, Istanbul, Turkey, June 2010. Springer.

[23] Robert Huijie Deng, Yingjiu Li, Moti Yung, and Yunlei Zhao. A New Framework for RFID Privacy. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *The Fifteenth European Symposium on Research in Computer Security – ESORICS 2010*, volume 6345 of *Lecture Notes in Computer Science*, pages 1–18, Athens, Greece, September 2010. Springer.

[24] Tassos Dimitriou. A Lightweight RFID Protocol to protect against Traceability and Cloning attacks. In *The First International Conference on Security and Privacy for Emerging Areas in Communications Networks – SecureComm 2005*, pages 59–66, Athens, Greece, September 2005. IEEE Communications Society/CreateNet, IEEE Computer Society Press.

[25] Tassos Dimitriou. rfidDOT: RFID Delegation and Ownership Transfer made simple. In *The Fourth International Conference on Security and Privacy in Communication Netowrks – SecureComm 2008*, Istanbul, Turkey, September 2008. ACM Press.

[26] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong Authentication for RFID Systems Using the AES Algorithm. In Marc Joye and Jean-Jacques Quisquater, editors, *The Sixth International Workshop on Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370, Cambridge, Massachusetts, USA, August 2004. Springer.

[27] Martin Feldhofer and Christian Rechberger. A Case Against Currently Used Hash Functions in RFID Protocols. In Mário Freire, Sim ao Melo de Sousa, and Vitor Santos, editors, *The First International Workshop on Information Security – IS 2006*, volume 4277 of *Lecture Notes in Computer Science*, pages 372–381, Montpellier, France, October–November 2006. Springer.

[28] Martin Feldhofer, Johannes Wolkerstorfer, and Vincent Rijmen. AES Implementation on a Grain of Sand. *IEE Proceedings on Information Security*, 152(1):13–20, October 2005.

[29] Sepideh Fouladgar and Hossam Afifi. A Simple Delegation Scheme for RFID Systems (SiDeS). In *IEEE International Conference on RFID*, Grapevine, Texas, USA, March 2007. IEEE Communications Society.

[30] Sepideh Fouladgar and Hossam Afifi. A Simple Privacy Protecting Scheme Enabling Delegation and Ownership Transfer for RFID Tags. *Journal of Communications*, 2(6):6–13, November 2007.

[31] Sepideh Fouladgar and Hossam Afifi. An Efficient Delegation and Transfer of Ownership Protocol for RFID tags. In *The First International EURASIP Workshop on RFID Technology*, Vienna, Austria, September 2007. European Association for Signal Processing (EURASIP).

[32] Keith B. Frikken, Marina Blanton, and Mikhail J. Atallah. Robust Authentication Using Physically Unclonable Functions. In Pierangela Samarati, Moti Yung, Fabio Martinelli, and Claudio A. Ardagna, editors, *The Twelfth International Information Security Conference – ISC 2009*, volume 5735 of *Lecture Notes in Computer Science*, pages 262–277, Pisa, Italy, September 2009. Springer.

[33] Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. Dismantling MIFARE Classic. In Sushil Jajodia and Javier Lopez, editors, *The Thirteenth European*

Symposium on Research in Computer Security – ESORICS 2008, volume 5283 of *Lecture Notes in Computer Science*, pages 97–114, Málaga, Spain, October 2008. Springer.

[34] Flavio D. Garcia and Peter van Rossum. Modeling Privacy for Off-Line RFID Systems. In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *The Ninth International Conference on Smart Card Research and Advanced Applications – CARDIS 2010*, volume 6035 of *Lecture Notes in Computer Science*, pages 194–208, Passau, Germany, April 2010. International Federation for Information Processing (IFIP), Springer.

[35] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. Good Variants of HB^+ Are Hard to Find. In Gene Tsudik, editor, *The Twelfth International Conference on Financial Cryptography and Data Security – FC 2008*, volume 5143 of *Lecture Notes in Computer Science*, pages 156–170, Cozumel, Mexico, January 2008. International Financial Cryptography Association (IFCA), Springer.

[36] Henri Gilbert, Matthew J. B. Robshaw, and Hervé Sibert. Active attack against HB^+: a provably secure lightweight authentication protocol. *IEEE IET Electronics Letters*, 41(21):1169–1170, October 2005.

[37] Philippe Golle, Markus Jakobsson, Ari Juels, and Paul F. Syverson. Universal Re-Encryption for Mixnets. In Tatsuaki Okamoto, editor, *The Cryptographers' Track at the RSA Conference on Topics in Cryptology – CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 163–178, San Francisco, California, USA, February 2004. RSA, Springer.

[38] JaeCheol Ha, SangJae Moon, Juan Manuel Gonzalez Nieto, and Colin Boyd. Low-Cost and Strong-Security RFID Authentication Protocol. In Mieso K. Denko, Chi-Sheng Shih, Kuan-Ching Li, Shiao-Li Tsao, Qing-An Zeng, Soo-Hyun Park, Young-Bae Ko, Shih-Hao Hung, and Jong Hyuk Park, editors, *The Third International Symposium on Security in Ubiquitous Computing – SecUbiq 2007*, volume 4809 of *Lecture Notes in Computer Science*, pages 795–807, Taipei, Taiwan, December 2007. Springer.

[39] JungHoon Ha, SangJae Moon, Jianying Zhou, and JaeCheol Ha. A New Formal Proof Model for RFID Location Privacy. In Sushil Jajodia and Javier Lopez, editors, *The Thirteenth European Symposium on Research in Computer Security*

– *ESORICS 2008*, volume 5283 of *Lecture Notes in Computer Science*, pages 267–281, Málaga, Spain, October 2008. Springer.

[40] Gerhard P. Hancke and Markus G. Kuhn. An RFID Distance Bounding Protocol. In *The First International Conference on Security and Privacy for Emerging Areas in Communications Networks – SecureComm 2005*, pages 67–73, Athens, Greece, September 2005. IEEE Communications Society/Create-Net, IEEE Computer Society Press.

[41] Dirk Henrici and Paul Müller. Hash-based Enhancement of Location Privacy for Radio-Frequency Identification Devices using Varying Identifiers. In *The First IEEE International Workshop on Pervasive Computing and Communication Security – PerSec 2004*, pages 149–153, Orlando, Florida, USA, March 2004. Institute of Electrical and Electronics Engineers (IEEE), IEEE Computer Society Press.

[42] Nicholas J. Hopper and Manuel Blum. A Secure Human-Computer Authentication Scheme. CMU Technical Report CMU-CS-00-139, May 2000.

[43] Ari Juels. Minimalist Cryptography for Low-Cost RFID Tags (Extended Abstract). In Carlo Blundo and Stelvio Cimato, editors, *The Fourth International Conference on Security in Communication Networks – SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 149–164, Amalfi, Italy, September 2004. Università di Salerno, Springer.

[44] Ari Juels. RFID Security and Privacy: A Research Survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, February 2006.

[45] Ari Juels, David Molnar, and David Wagner. Security and Privacy Issues in E-passports. In *The First International Conference on Security and Privacy for Emerging Areas in Communications Networks – SecureComm 2005*, pages 74–88, Athens, Greece, September 2005. IEEE Communications Society/Create-Net, IEEE Computer Society Press.

[46] Ari Juels and Ravikanth Pappu. Squealing Euros: Privacy Protection in RFID-Enabled Banknotes. In Rebecca N. Wright, editor, *The Seventh International Conference on Financial Cryptography – FC 2003*, volume 2742 of *Lecture Notes in Computer Science*, pages 103–121, Le Gosier, Guadeloupe, French West Indies, January 2003. International Financial Cryptography Association (IFCA), Springer.

[47] Ari Juels, Ronald L. Rivest, and Michael Szydlo. The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy. In Vijayalakshmi Atluri and Peng Liu, editors, *The Tenth ACM Conference on Computer and Communications Security – CCS 2003*, pages 103–111, Washington, D.C., USA, October 2003. Association for Computing Machinery (ACM), ACM Press.

[48] Ari Juels and Stephen A. Weis. Authenticating Pervasive Devices with Human Protocols. In Victor Shoup, editor, *The Twenty-Fifth Annual International Cryptology Conference on Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 293–308, Santa Barbara, California, USA, August 2005. International Association for Cryptologic Research (IACR), Springer.

[49] Ari Juels and Stephen A. Weis. Defining Strong Privacy for RFID. Cryptology ePrint Archive, Report 2006/137, April 2006.

[50] Ari Juels and Stephen A. Weis. Defining Strong Privacy for RFID (Extended Abstract). In *The First International Workshop on Pervasive RFID/NFC Technology and Applications – PerTec 2007*, pages 342–347, White Plains, New York, USA, March 2007. Institute of Electrical and Electronics Engineers (IEEE), IEEE Computer Society Press.

[51] Ari Juels and Stephen A. Weis. Defining Strong Privacy for RFID. *ACM Transactions on Information and System Security – TISSEC 2009*, 13(1):7:1–7:23, October 2009.

[52] Jeonil Kang and DaeHun Nyang. RFID Authentication Protocol with Strong Resistance Against Traceability and Denial of Service Attacks. In Refik Molva, Gene Tsudik, and Dirk Westhoff, editors, *The Second European Workshop on Security and Privacy in Ad-hoc and Sensor Networks – ESAS 2005*, volume 3813 of *Lecture Notes in Computer Science*, pages 164–175, Visegrad, Hungary, July 2005. Springer.

[53] Ziv Kfir and Avishai Wool. Picking Virtual Pockets using Relay Attacks on Contactless Smartcard Systems. In *The First International Conference on Security and Privacy for Emerging Areas in Communications Networks – SecureComm 2005*, pages 47–58, Athens, Greece, September 2005. IEEE Communications Society/Create-Net, IEEE Computer Society Press.

[54] Il Jung Kim, Eun Young Choi, and Dong Hoon Lee. Secure Mobile RFID system against privacy and security problems. In P. Georgiadis, J. Lopez, S. Gritzalis, and G. F. Marias, editors, *The Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing – SECPerU 2007*, pages 67–72, Istanbul, Turkey, July 2007. Institute of Electrical and Electronics Engineers (IEEE), IEEE Computer Society Press.

[55] Kyoung Hyun Kim, Eun Young Choi, Su Mi Lee, and Dong Hoon Lee. Secure EPCglobal Class-1 Gen-2 RFID System Against Security and Privacy Problems. In Mário Freire, Sim ao Melo de Sousa, and Vitor Santos, editors, *The First International Workshop on Information Security – IS 2006*, volume 4277 of *Lecture Notes in Computer Science*, pages 362–371, Montpellier, France, October–November 2006. Springer.

[56] K. H. S. Sabaragamu Koralalage, Mohammed Reza Selim, Junichi Miura, Yuichi Goto, and Jingde Cheng. POP Method: An Approach to Enhance the Security and Privacy of RFID Systems Used in Product Lifecycle with an Anonymous Ownership Transferring Mechanism. In *ACM Symposium on Applied Computing – SAC 2007*, pages 270–275, Seoul, Korea, March 2007. ACM Press.

[57] Karl Koscher, Ari Juels, Vjekoslav Brajkovic, and Tadayoshi Kohno. EPC RFID Tag Security Weaknesses and Defenses: Passport Cards, Enhanced Drivers Licenses, and Beyond. In Somesh Jha and Angelos D. Keromytis, editors, *The Sixteenth ACM Conference on Computer and Communications Security – CCS 2009*, pages 33–42, Chicago, Illinois, USA, November 2009. Association for Computing Machinery (ACM), ACM Press.

[58] Junzuo Lai, Robert Huijie Deng, and Yingjiu Li. Revisiting Unpredictability-Based RFID Privacy Models. In Jianying Zhou and Moti Yung, editors, *The Eighth International Conference on Applied Cryptography and Network Security – ACNS 2010*, volume 6123 of *Lecture Notes in Computer Science*, pages 475–492, Beijing, China, June 2010. International Communications and Information Security Association (ICISA), Springer.

[59] Jooyoung Lee and Yongjin Yeom. Efficient RFID Authentication Protocols Based on Pseudorandom Sequence Generators. Cryptology ePrint Archive, Report 2008/343, August 2008.

[60] Su Mi Lee, Young Ju Hwang, Dong Hoon Lee, and Jong In Lim. Efficient Authentication for Low-Cost RFID Systems. In Osvaldo Gervasi, Marina L. Gavrilova, Vipin Kumar, Antonio Laganà, Heow Pueh Lee, Youngsong Mun, David Taniar, and Chih Jeng Kenneth Tan, editors, *The Fifth International Conference on Computational Science and Its Applications – ICCSA 2005*, volume 3480 of *Lecture Notes in Computer Science*, pages 619–627, Singapore, May 2005. Springer.

[61] Yingjiu Li and Xuhua Ding. Protecting RFID Communications in Supply Chains. In Robert Deng and Pierangela Samarati, editors, *The Third ACM Symposium on Information, Computer and Communications Security – ASIACCS 2007*, pages 234–241, Singapore, March 2007. Association for Computing Machinery (ACM), ACM Press.

[62] Chae Hoon Lim and Taekyoung Kwon. Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *The Eighth International Conference on Information and Communications Security – ICICS 2006*, volume 4307 of *Lecture Notes in Computer Science*, pages 1–20, Raleigh, North Carolina, USA, December 2006. Springer.

[63] Nai-Wei Lo and Kuo-Hui Yeh. An Efficient Mutual Authentication Scheme for EPCglobal Class-1 Generation-2 RFID System. In Mieso K. Denko, Chi-Sheng Shih, Kuan-Ching Li, Shiao-Li Tsao, Qing-An Zeng, Soo-Hyun Park, Young-Bae Ko, Shih-Hao Hung, and Jong Hyuk Park, editors, *The Second International Workshop on Trustworthiness, Reliability and Services in Ubiquitous and Sensor neTworks – TRUST 2007*, volume 4809 of *Lecture Notes in Computer Science*, pages 43–56, Taipei, Taiwan, December 2007. Springer.

[64] Nai-Wei Lo and Kuo-Hui Yeh. Novel RFID Authentication Schemes for Security Enhancement and System Efficiency. In Willem Jonker and Milan Petković, editors, *The Fourth VLDB Workshop on Secure Data Management – SDM 2007*, volume 4721 of *Lecture Notes in Computer Science*, pages 203–212, Vienna, Austria, September 2007. Very Large Data Base (VLDB) Endowment Inc., Springer.

[65] Changshe Ma, Yingjiu Li, Robert Huijie Deng, and Tieyan Li. RFID Privacy: Relation Between Two Notions, Minimal Condition, and Efficient Construction. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *The Sixteenth ACM Conference on Computer and Communications Security – CCS 2009*, pages

54–65, Chicago, Illinois, USA, November 2009. Association for Computing Machinery (ACM), ACM Press.

[66] Máire McLoone. Low-Cost SHA-1 Hash Function Architecture for RFID Tags. In *The Fourth International Workshop on Radio Frequency Identification: Security and Privacy Issues – RFIDsec 2008*, Budapest, Hungary, July 2008. Institute for Applied Information Processing and Communications (IAIK) TU Graz.

[67] David Molnar, Andrea Soppera, and David Wagner. A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags (Extended Abstract). In Bart Preneel and Stafford Tavares, editors, *The Twelfth International Workshop on Selected Areas in Cryptography – SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 276–290, Kingston, Ontario, Canada, August 2005. Springer.

[68] David Molnar, Andrea Soppera, and David Wagner. A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags (Long Version). Cryptology ePrint Archive, Report 2005/315, September 2005.

[69] David Molnar and David Wagner. Privacy and Security in Library RFID: Issues, Practices, and Architectures. In Birgit Pfitzmann and Peng Liu, editors, *The Eleventh ACM Conference on Computer and Communications Security – CCS 2004*, pages 210–219, Washington, D.C., USA, October 2004. Association for Computing Machinery (ACM), ACM Press.

[70] Ching Yu Ng, Willy Susilo, Yi Mu, and Rei Safavi-Naini. RFID Privacy Models Revisited. In Sushil Jajodia and Javier Lopez, editors, *The Thirteenth European Symposium on Research in Computer Security – ESORICS 2008*, volume 5283 of *Lecture Notes in Computer Science*, pages 251–266, Málaga, Spain, October 2008. Springer.

[71] Ching Yu Ng, Willy Susilo, Yi Mu, and Rei Safavi-Naini. New Privacy Results on Synchronized RFID Authentication Protocols against Tag Tracing. In Michael Backes and Peng Ning, editors, *The Fourteenth European Symposium on Research in Computer Security – ESORICS 2009*, volume 5789 of *Lecture Notes in Computer Science*, pages 321–336, Saint-Malo, France, September 2009. Springer.

[72] Ching Yu Ng, Willy Susilo, Yi Mu, and Rei Safavi-Naini. Practical RFID ownership transfer scheme. In Yingjiu Li and Jianying Zhou, editors, *The Second*

Radio Frequency Identification System Security Asia Workshop (RFIDsec Asia), volume 4 of *Cryptology and Information Security Series*, pages 33–48, Singapore, February 2010. IOS Press.

[73] Ching Yu Ng, Willy Susilo, Yi Mu, and Rei Safavi-Naini. Practical RFID ownership transfer scheme. *Journal of Computer Security*, 19(2):319–341, March 2011.

[74] Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic Approach to "Privacy-Friendly" Tags. In *RFID Privacy Workshop*, MIT, Massachusetts, USA, November 2003.

[75] Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. RFID Privacy Issues and Technical Challenges. *Communications of the ACM – Special issue: RFID*, 49(9):66–71, September 2005.

[76] Kyosuke Osaka, Tsuyoshi Takagi, Kenichi Yamazaki, and Osamu Takahashi. An Efficient and Secure RFID Security Method with Ownership Transfer. In Yunping Wang, Yiu ming Cheung, and Hailin Liu, editors, *Computational Intelligence and Security – CIS 2006*, volume 4456 of *Lecture Notes in Computer Science*, pages 778–787, Guangzhou, China, November 2006. Springer.

[77] Radu-Ioan Paise and Serge Vaudenay. Mutual Authentication in RFID: Security and Privacy. In Masayuki Abe and Virgil D. Gligor, editors, *The Third ACM Symposium on Information, Computer and Communications Security – ASIACCS 2008*, pages 292–299, Tokyo, Japan, March 2008. Association for Computing Machinery (ACM), ACM Press.

[78] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda. RFID Systems: A Survey on Security Threats and Proposed Solutions. In Pedro Cuenca and Luiz Orozco-Barbosa, editors, *The Eleventh International Conference on Personal Wireless Communications – PWC 2006*, volume 4217 of *Lecture Notes in Computer Science*, pages 159–170, Albacete, Spain, September 2006. International Federation for Information Processing (IFIP), Springer.

[79] Roberto Di Pietro and Refik Molva. Information Confinement, Privacy, and Security in RFID Systems. In Joachim Biskup and Javier Lopez, editors, *The Twelfth European Symposium on Research in Computer Security – ESORICS 2007*, volume

4734 of *Lecture Notes in Computer Science*, pages 187–202, Dresden, Germany, September 2007. Springer.

[80] Junichiro Saito, Kenji Imamoto, and Kouichi Sakurai. Reassignment Scheme of an RFID Tag's Key for Owner Transfer. In Tomoya Enokido, Lu Yan, Bin Xiao, Daeyoung Kim, Yuan-Shun Dai, and Laurence Tianruo Yang, editors, *Embedded and Ubiquitous Computing Workshops – EUC 2005*, volume 3823 of *Lecture Notes in Computer Science*, pages 1303–1312, Nagasaki, Japan, December 2005. Springer.

[81] Sanjay E. Sarma, Stephen A. Weis, and Daniel W. Engels. RFID Systems and Security and Privacy Implications. In Burton S. Kaliski Jr., Çetin K. Koç, and Christof Paar, editors, *The Fourth International Workshop on Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 454–469, Redwood Shores, California, USA, August 2002. Springer.

[82] Youngjoon Seo, Hyunrok Lee, and Kwangjo Kim. A Scalable and Untraceable Authentication Protocol for RFID. In Xiaobo Zhou, Oleg Sokolsky, Lu Yan, Eun-Sun Jung, Zili Shao, Yi Mu, Dong-Chun Lee, Daeyoung Kim, Young-Sik Jeong, and Cheng-Zhong Xu, editors, *The Second International Workshop on Security in Ubiquitous Computing Systems – SecUbiq 2006*, volume 4097 of *Lecture Notes in Computer Science*, pages 252–261, Seoul, Korea, August 2006. Springer.

[83] Boyeon Song. RFID Tag Ownership Transfer. In *The Fourth International Workshop on Radio Frequency Identification: Security and Privacy Issues – RFIDsec 2008*, Budapest, Hungary, July 2008. Institute for Applied Information Processing and Communications (IAIK) TU Graz.

[84] Andrea Soppera and Trevor Burbridge. Secure by Default: The RFID Acceptor Tag (RAT). In *The Second International Workshop on RFID Security – RFIDSec 2006*, Graz, Austria, July 2006. Institute for Applied Information Processing and Communications (IAIK) TU Graz.

[85] Gene Tsudik. YA-TRAP: Yet Another Trivial RFID Authentication Protocol. In *The Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops – PerCom Workshops 2006*, pages 640–643, Pisa, Italy, March 2006. Institute of Electrical and Electronics Engineers (IEEE), IEEE Computer Society.

[86] Gene Tsudik. A Family of Dunces: Trivial RFID Identification and Authentication Protocols. In Nikita Borisov and Philippe Golle, editors, *The Seventh International Symposium on Privacy Enhancing Technologies – PET 2007*, volume 4776 of *Lecture Notes in Computer Science*, pages 45–61, Ottawa, Canada, June 2007. Springer.

[87] Information Security Group Université catholique de Louvain. RFID Security & Privacy Lounge. http://www.avoine.net/rfid/index.php, August 2011.

[88] Ton van Deursen, Sjouke Mauw, and Saša Radomirović. Untraceability of RFID Protocols. In Jose Antonio Onieva, Damien Sauveron, Serge Chaumette, Dieter Gollmann, and Konstantinos Markantonakis, editors, *The Second International Workshop in Information Security Theory and Practices. Smart Devices, Convergence and Next Generation Networks – WISTP 2008*, volume 5019 of *Lecture Notes in Computer Science*, pages 1–15, Seville, Spain, May 2008. International Federation for Information Processing (IFIP), Springer.

[89] Ton van Deursen, Sjouke Mauw, Saša Radomirović, and Pim Vullers. Secure Ownership and Ownership Transfer in RFID Systems. In Michael Backes and Peng Ning, editors, *The Fourteenth European Symposium on Research in Computer Security – ESORICS 2009*, volume 5789 of *Lecture Notes in Computer Science*, pages 637–654, Saint-Malo, France, September 2009. Springer.

[90] Ton van Deursen and Saša Radomirović. Attacks on RFID Protocols. Cryptology ePrint Archive, Report 2008/310, July 2008.

[91] Ton van Deursen and Saša Radomirović. On a New Formal Proof Model for RFID Location Privacy (Extended Version). Cryptology ePrint Archive, Report 2008/477, November 2008.

[92] Ton van Deursen and Saša Radomirović. On a New Formal Proof Model for RFID Location Privacy. *Information Processing Letters*, 110(2):57–61, December 2009.

[93] Serge Vaudenay. On Privacy Models for RFID. In Kaoru Kurosawa, editor, *The Thirteenth International Conference on the Theory and Application of Cryptology and Information Security – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87, Kuching, Malaysia, December 2007. International Association for Cryptologic Research (IACR), Springer.

[94] Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In Dieter Hutter, Günter Müller, Werner Stephan, and Markus Ullmann, editors, *The First 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.

[95] Jeongkyu Yang, Jaemin Park, Hyunrok Lee, Kui Ren, and Kwangjo Kim. Mutual Authentication Protocol for Low-cost RFID. In *The First International Workshop on Radio Frequency Identification and Lightweight Crypto – RFIDsec 2005*, Graz, Austria, July 2005. Institute for Applied Information Processing and Communications (IAIK) TU Graz.

[96] Xiaolan Zhang and Brian King. Integrity Improvements to an RFID Privacy Protection Protocol for Anti-counterfeiting. In Jianying Zhou, Javier Lopez, Robert Huijie Deng, and Feng Bao, editors, *The Eighth International Information Security Conference – ISC 2005*, volume 3650 of *Lecture Notes in Computer Science*, pages 474–481, Singapore, September 2005. Springer.

[97] Xiaolan Zhang and Brian King. Modeling RFID Security. In Dengguo Feng, Dongdai Lin, and Moti Yung, editors, *The First SKLOIS Conference on Information Security and Cryptology – CISC 2005*, volume 3822 of *Lecture Notes in Computer Science*, pages 75–90, Beijing, China, December 2005. State Key Laboratory of Information Security (SKLOIS), Springer.

[98] Xiaolan Zhang and Brian King. An Anti-Counterfeiting RFID Privacy Protection Protocol. *Journal of Computer Science and Technology*, 22(3):438–448, May 2007.

[99] Xiaolan Zhang and Brian King. Security Requirements for RFID Computing Systems. *International Journal of Network Security*, 6(2):214–226, March 2008.