2015

Security and Privacy in RFID Systems

Nan Li

Security and Privacy in RFID Systems

Nan Li

This thesis is presented as part of the requirements for the conferral of the degree:

Doctor of Philosophy

The University of Wollongong
School of Computing and Information Technology

July 20, 2015

Declaration

I, Nan Li, declare that this thesis submitted in partial fulfilment of the requirements for the conferral of the degree Doctor of Philosophy, from the University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.

Nan Li
July 20, 2015

Abstract

Radio Frequency Identification (RFID) technology has been widely applied in many applications, such as the supply chain and warehouse management. A typical RFID system consists of RFID tags, RFID readers and a back-end server. Usually, a tag is a constrained device attached to an object. It can be scanned by a reader and identified by the server.

An important branch of RFID research is RFID authentication protocols. There have been many research works addressing RFID authentication issues. The security and privacy of RFID protocols have become hot research topics. Many protocols have been proposed to achieve various requirements. According to the different features of protocols, the related security and privacy issues should be considered. The security and privacy solutions, including the model definition and protocol analysis, are the major work of this thesis.

Some existing protocols have attempted to achieve a strong privacy. Unfortunately, even if they sacrifice the efficiency and cost of protocols, most of them cannot reach the highest privacy level called wide-strong privacy. To address this issue, we review these problems existing in previous work and propose some potential solutions. We then present a wide-strong private RFID authentication protocol with a formal proof of privacy. We also study a scenario that a server can only assist a reader to authenticate tags without obtaining any tag's identity. Accordingly, we introduce a novel concept called authorized RFID authentication and propose three protocols to adapt to different environments.

As an extension of RFID authentication protocols, RFID tag ownership transfer protocols have different security requirements. An ownership transfer protocol transfers a tag's ownership from the current owner to a new owner. A precondition of a tag ownership transfer is to verify the validity of tags. Apart from the tag authentication, it is also necessary to check the authenticity of the current owner. Based on this observation, we introduce an ownership chain to solve the problem and we propose a secure ownership transfer protocol. Moreover, the ownership of a tag may be shared by multiple users. A shared tag ownership transfer is allowed if all the owners agree. In an environment where a trusted third party is unavailable, one needs to prevent some owners from forging the transfer agreements of others. We present a hybrid protocol that a tag stores constant-size secret keys and an owner can individually prove a partial ownership. The proposed protocol guarantees that the shared ownership cannot be transferred prior to the presence of a full agreement.

An example of using untrusted readers is yoking-proof which guarantees the simultaneous presence of two tags. It also requires that a proof can be verified by an offline verifier. Grouping proof, which allows multiple tags to be verified, is a generalized version of yoking-proof. A limitation of grouping proof is that it can only authenticate a single group of tags, which is insufficient in many cases. We introduce the yoking-group proof so that it can verify the existence of multiple individual tag groups. We propose an anonymous yoking-proof protocol as a building block and present an anonymous yoking-group proof protocol. The security of two proposed protocols are analyzed in the universal composible framework.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Professor Yi Mu, for his patient guidance of research methodology and problem solving skills. I appreciate that Professor Yi Mu provided innumerable suggestions to my study and great assistance of my academic writing. This thesis cannot be presented without his support.

I would like to give my deep appreciation to my co-supervisor Professor Willy Susilo, for his distinctive view of research problems and the guidance of research. I would like to thank Dr. Fuchun Guo, for his invaluable suggestions. I wish to extend my thanks to people who supported helpful discussions and suggestions to me. The non-exhausted list includes: Man Ho Au, Hui Cui, Rongmao Chen, Ibrahim Elashry, Jinguang Han, Yinhao Jiang, Jongkil Kim, Jorge Munilla, Tran Phong, Shams Ud Din Qazi, Yang Wang, Guilin Wang, Kefeng Wang, Yong Yu, Guomin Yang, Jiangshan Yu, Miao Zhou, Zhenfei Zhang and Minjie Zhang. I would also like to give many thanks to all staff of School of Computer Science and Software Engineering for their support, as well as anonymous reviewers of the papers included in this thesis.

I would like to thank University of Wollongong and Australian Research Council Discovery Project (DP110101951) for offering me scholarship.

At last, I am grateful to my wife Bai Xue, for her patience, support and love. Without her, my life would be extremely chaotic and this research work would never be possible.

Publications

The following publications/manuscripts are related to this thesis.

1. Nan Li, Yi Mu, Willy Susilo, Fuchun Guo, Vijay Varadharajan. Privacy-preserving authorized RFID authentication protocols. In *Radio Frequency Identification: Security and Privacy Issues - 10th International Workshop, RFIDSec 2014*, volume 8651 of LNCS, pages 108–122, 2014.

2. Nan Li, Yi Mu, Willy Susilo, Vijay Varadharajan. Anonymous yoking-group proofs. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIACCS '15*, pages 615–620, 2014.

3. Nan Li, Yi Mu, Willy Susilo, Vijay Varadharajan. Secure RFID ownership transfer protocols. In *Information Security Practice and Experience - 9th International Conference, ISPEC 2013*, volume 7863 of LNCS, pages 189–203, 2013.

4. Nan Li, Yi Mu, Willy Susilo, Fuchun Guo, Vijay Varadharajan. On RFID authentication protocols with wide-strong privacy. In *Radio Frequency Identification System Security, RFIDsec Asia*, volume 11 of *Cryptology and Information Security Series*, pages 3–16, 2013.

5. Nan Li, Yi Mu, Willy Susilo, Vijay Varadharajan. Shared RFID ownership transfer protocols. *Computer Standers & Interfaces*, 42(0):95 – 104, 2015.

6. Nan Li, Yi Mu, Willy Susilo, Fuchun Guo, Vijay Varadharajan. Vulnerabilities of an ECC-based RFID authentication scheme. *Security and Communication Networks*, 2015. (accepted)

List of Abbreviations

The following abbreviations are used throughout this thesis. Some special abbreviations will be defined when they are first used.

ARA	Authorized RFID Authentication;
CRC	Cyclic Redundancy Check;
ECC	Elliptic Curve Cryptography;
EC-RAC	Elliptic Curve Discrete Logarithm Problem based Randomized Access Control;
ECDH	Elliptic Curve Diffie-Hellman;
ECDLP	Elliptic Curve Discrete Logarithm Problem;
EDBDH	Extended Decisional Bilinear Diffie-Hellman;
IND-CCA2	Indistinguishability against Adaptive Chosen-Ciphertext Attacks;
IND-CPA	Indistinguishability against Adaptive Chose-Plaintext Attacks;
MITM	Man-in-the-Middle;
ODH	Oracle Diffie-Hellman;
PKC	Public Key Cryptography;
PPT	Probabilistic Polynomial Time;
PRNG	Pesudorandom Number Generators;
PUF	Physically Uncloneable Function;
RFID	Radio Frequency Identification;
SROT	Shared RFID Ownership Transfer;
TTP	Trusted Third Party;
TC	Trusted Center;
V-l-wDBDHI	Various l-weak Decisional Bilinear Diffie-Hellman Inversion;

[This page is intentionally left blank]

Contents

List of Tables

List of Figures

[This page is intentionally left blank]

Chapter 1

Introduction

1.1 Overview

A typical RFID system consists of three main components, namely RFID tags, readers, and a back-end server. It exploits radio waves to exchange data between wireless devices. Usually, an RFID reader sends a request to target tags and collects the tags' responses. The reader then forwards data to the back-end server which manages a database. The server checks the validity of the responses and a tag is valid if its valid record is stored on the database. Compared with a barcode system, RFID is an improved technique that has several features: 1) RFID systems provide automated contactless authentication of tags. 2) Each tag is assigned a unique identifier which indicates a specific object. 3) Because of the uniqueness of RFID, an authorized user can locate and trace a tag. It is useful in many applications, such as warehouse management and child care.

RFID protocols describe interactions between participating entities. There are varied types of protocols which serve for different requirements, such as security, efficiency and others. In Figure 1.1, we provide a map of RFID protocol considerations. The research regarding the tag side is challenging, because tags have extremely limited capabilities. In an RFID system, all the components, except tags, can utilize traditional security techniques which are sufficient to achieve security and privacy objectives. For example, typical RFID protocols assume that the communication channel between the reader and back-end server is secure. Therefore, most research focuses on protecting the tag's security and privacy.

Privacy and Security of Tags. Correctness and soundness are important to the security of RFID authentication protocols. A secure protocol guarantees that an adversary has negligible probability to impersonate an uncorrupted legitimate tag. The privacy of a protocol preserves the tag anonymity and untraceability. A tag should be identifiable by an authorized verifier and it cannot be recognized by an attacker who has protocol instances. Many proposed protocols focus on the security and privacy problems and some solutions also use *security* to describe privacy issues.

Tag unforgeability is one aspect of the security of protocols. It guarantees that an adversary cannot forge valid responses of tags. Additionally, a low-cost RFID tag is usually without tamper-proof protection. If a tag has been corrupted and cloned, a back-end server should have abilities to detect and reject counterfeited tags.

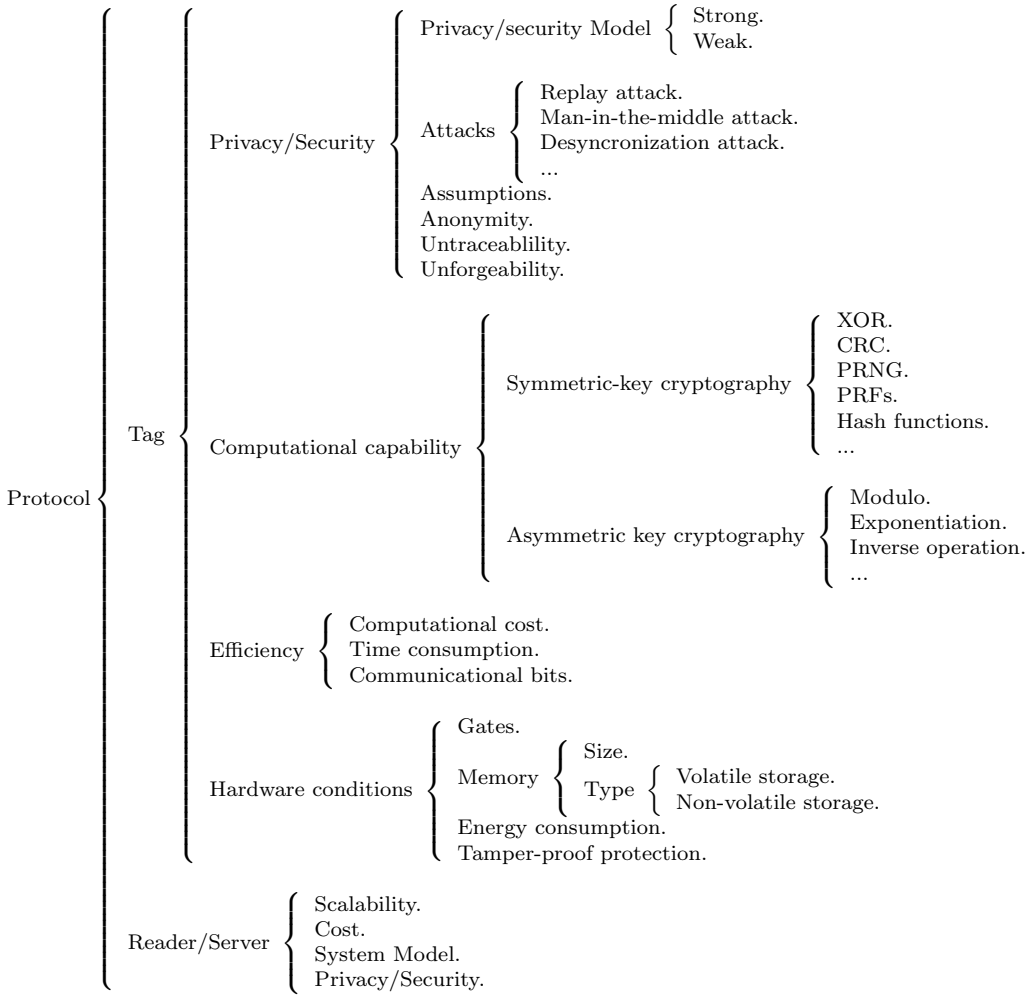

Figure 1.1: RFID protocol considerations.

Physically uncloneable function (PUF) [MV10, SD07] is a tool to prevent tags from counterfeiting and many PUF-based protocols [TB06, BR07, BCI08b] have been proposed. PUFs use physical features of tags to generate unpredictable outputs and physical attacks against a tag's PUF will immediately modify the PUF. An attack which attempts to corrupt and clone tags can be easily prevented. Moreover, side-channel attacks exploit physical information, such as timing information and power consumption, to capture the secret of tags. Even if PUF is used, it cannot prevent a side-channel attacker from counterfeiting uncompromised tags. In RFID systems, clone detection [Aba09, LOIM09, ZCJ13, BJRS13] can detect counterfeited tags with high probability. Prevention, which is provided by cryptographic tools and PUFs, and detection are two general strategies of resisting tag cloning attacks.

Apart from the tag unforgeability, various security problems are illustrated in different applications. For example, yoking-proof protocols [Jue04, CYH$^+$08, HP12] need to convince a verifier that two tags are presented simultaneously. An attacker,

who does not have both tags, violates the security of a protocol if he/she can output a valid proof. We discuss details of yoking-proof protocols and their variants in Chapter 7. Some other security issues will also be demonstrated and solved in this thesis.

Tag anonymity and untraceability are two important privacy requirements. An RFID authentication protocol is anonymous if an attacker cannot recover a tag's identity from the protocol instances. Tag untraceability usually means that tags are indistinguishable in different sessions. Given two tags and the protocol instances, an adversary should only have a negligible advantage to correctly find the tag involved in the specific instance. RFID privacy models focus on tag untraceability since it normally implies tag anonymity. Tag privacy also has several levels, such as weak privacy and forward privacy. There have been many proposed privacy models [LDL10, Vau07, NSMSN08], but they are still incomplete [CM13]. For example, these models cannot completely capture some important features of adversaries, like the use of corrupted tags.

To analyze the security and privacy of RFID protocols, there are two approaches. The first is to analyze the protocol under different attacks, so that it is considered as an informal analysis. The other approach is to prove the tag's security and privacy in formal models. If a protocol is proven secure, an adversary is unlikely to succeed in attacks which are captured in the model. A privacy model defines the abilities of adversaries and a game which starts by providing the adversary with public parameters. The characterization of the adversary is captured by oracles and the adversary queries these oracles following some rules of the game. In the privacy proof, an adversary interacts with the system initialized by the challenger. We say that the adversary breaks the privacy of a protocol if he wins the game. The oracles and the rules also specify the level of privacy models. As an example, Vaudenay's privacy model [Vau07, Vau10] defines eight levels of RFID privacy such that each level is restricted by the access of oracles. Security models formalize the security requirements, such as tag unforgeability. The approach of security reduction is similar to the approach of privacy proofs.

Assumptions are important in security and privacy analyses. There are two types of assumptions: the assumptions related to the adversary's power and the computational complexity assumptions. RFID protocols are different from traditional cryptographic schemes since the hardware severely limits capabilities of tags. In some scenarios, we have to make assumptions which restrict the actions of adversaries. Consider that a protocol adopts symmetric-key cryptography and the adversary is allowed to corrupt tags, so it is impossible to preserve the tag's backward privacy after it has been compromised. Many symmetric-key based RFID protocols [Tsu06, CLL05, FMTRCRDF11a] assume that the tag is incorruptible and add re-

strictions to adversaries. System assumptions, such as the secure communication channel between the reader and server, can also specify the abilities of attackers. Complexity assumptions are used in formal proofs. To show security and privacy, we prove the equality between breaking the protocol and solving mathematically hard problems.

Informal security and privacy analysis consider a set of attacks, such as replay attack and man-in-the-middle attack, against protocols. It utilizes security assumptions and operations of the protocol to demonstrate security and privacy. A drawback of informal analysis is that the adversary's actions are restricted, thus he protocol may be vulnerable to variants of attacks if it uses the informal analysis.

Computational Capabilities of Tags. Low-cost RFID tags are normally assumed to perform lightweight protocols. Most traditional cryptographic schemes, such as AES, DES and public key encryption schemes, are unaffordable. An important task of RFID protocols is to reduce the computational cost of tags and achieve the required security.

There are two types of RFID protocols: symmetric-key based protocols and asymmetric key based protocols. A low-cost tag is desired to carry lightweight operations such as XOR, cyclic redundancy check (CRC) and pesudorandom number generators (PRNG). Some protocols use CRC to provide confidentiality, while it is vulnerable to several attacks due to the lack of cryptographic operations [PLOHCvdL11]. Symmetric-key based schemes are necessary to achieve weak privacy. Collision-resistant cryptographic hash functions and lightweight symmetric-key encryption schemes [LK06a] can also achieve forward privacy if there is a secure tag key update. Moreover, public key cryptography (PKC) is needed in protocols which require strong privacy. PKC provides many useful properties and an RFID protocol cannot achieve wide-strong privacy without using public key encryption schemes [Vau07]. A drawback is that tags have to sacrifice efficiency and cost. Asymmetric key based RFID protocols usually require tags to perform some heavy computations, such as modular operations and exponentiation. One challenging task of using PKC is to efficiently implement these operations so that some research work focuses on hardware platform design [HWF08, PPH11]. In addition, elliptic curve cryptography (ECC) has been widely used in public key based RFID protocols [LBV08, LBV09, BLS+10, vDR10, BSSV11, BLS+12, PH13].

Efficiency. The efficiency of RFID protocols is an essential requirement of RFID systems. There are three main aspects which are computational costs of tags, time consumption and communication bits. Types of cryptographic schemes influence the computational cost of tags. Symmetric-key based protocols normally provide high efficiency even if some hash functions require many calculations. Many

cryptographic operations, such as modular exponentiation, in ECC-based protocols are heavy and they make tag authentication inefficient. The cost of tag-reader communication is another consideration of the efficiency. HB-like protocols [HB01, Pir06, BCD06, GRS08, RG12] are lightweight RFID protocols that computational costs are affordable to low-cost tags. Since these protocols need to run several times for a tag authentication, large amount of communication bits is a disadvantage. An important problem is to make trade-offs between communication bits and computational costs. It is hard to find an efficient solution for both of them if a protocol requires strong privacy. The back-end server's performance also impacts efficiency of the whole system. An authentication process which needs exhaustively tag key search could be inefficient.

Hardware Conditions. Capabilities of RFID tags depend on hardware platforms. There are three types of tags which are active tags, semi-active tags and passive tags. Active tags have strong computational capabilities since they are battery embedded. Traditional security solutions are normally affordable to preserve security and privacy of tags, but active tags are expansive. Semi-active tags also have batteries and their capabilities are between active tags and passive tags. Compared with active tags, they only use their batteries to activate chips. Passive tags are low-cost and suitable for large-scale deployment, such as the supply chain. A restriction of passive tags is that they have limited capabilities. For example, some cryptographic operations like hashing may not be implemented on passive tags because they need a large number of gates. The size and type of tag memory are important to RFID protocol design. Some protocols [BR05, Pir06] which require tags to store counters in non-volatile memory will increase the cost of tags. The use of counters have some potential security and privacy problems as well. A passive tag can usually store several hundred bits of information, including identity, keys and other data. Large memory causes high power consumption so that RFID protocols should reduce the requirement of tag memory. Finally, tamper-proof resistant hardware cannot be implemented on low-cost passive RFID tags.

Reader and Server. In an RFID system, readers and the back-end server are powerful devices to perform various tasks. RFID readers use radio waves to interact with tags and server's database stores all the information of tags. Some system models also allow a reader to possess keys of tags and we usually assume that readers and back-end server are mutually trusted. System scalability which requires efficient tag key search is another direction of RFID research. Many proposed protocols [Jue06, WLHL07] utilize special data structures to assign and store keys of tags, and then reduce the complexity of key search during tag authentication.

1.2 Problems

Our work focuses on some problems motivated by observations of current RFID research. An important one is that many proposed protocols cannot be provably secure and they may be vulnerable to potential attacks. It is necessary to give formal proofs of security and privacy for protocol analysis. In this section, we provide an overview of the problems addressed in this thesis.

RFID Protocols on Strong Privacy. Tag privacy has been widely discussed in previous work and many authentication protocols have been proposed. However, wide-strong privacy is still hard to achieve. We find several common problems of some current protocols and it is useful to provide general ideas of solving these issues.

Tag Ownership Chains. RFID tag ownership transfer protocols use ownership verification to verify current owners. Existing protocols illustrate that the knowledge of the tag's secret shows the current ownership. If a tag cannot provide any tamper-proof protection, current ownership verification is indeed insufficient. A powerful adversary can recover internal state of tags and impersonate the current owner. We need a solution which checks the validity of previous ownership to resist this attack.

Shared RFID Tag Ownership Transfer. Traditional ownership transfer protocols assume that a tag has a single owner and the secret information of the tag is completely possessed by one owner. Some protocols extend the concept to multiple owners and these solutions require a trusted third party (TTP) to transfer the secret. We consider an environment where TTP is unavailable. Several security problems, such as individual ownership proof, agreed ownership transfer and secure key update are challenging to design a secure shared ownership transfer protocol in two-party model.

Relationship of RFID Entities. A back-end server normally controls the whole RFID system that it stores the information of readers and tags. Readers transmit tag's responses to the server and receives the result. We have to consider some new privacy issues in scenarios where tags, readers and back-end server are independent. For instance, the reader's privacy is important if readers only need the computational assistance of the server rather than the tag authentication. A protocol should prevent the result of authentication like the tag's identity from being obtained by the server. Also, the server needs to provide verifiable responses to the readers. There is no existing solution that addresses these problems.

Group Tags Authentication. Yoking-proof protocols allow an offline server to check the existence of two tags. A reader is usually assumed to be untrusted and it

is prevented from obtaining the tag's identity. Grouping proof protocol [SS05, BR05, PHER07] which is a generalized version of yoking-proof guarantees the simultaneous presence of multiple tags. Imagine that several individual groups of tags need to be scanned together and a verifier requires to recognize the different groups. We cannot apply grouping proof protocols since these tags cannot be considered as a single group. The group integrity and anonymity are also important security and privacy requirements. A solution which solves these issues is needed.

1.3 Thesis Organization

This thesis consists of 8 chapters which present from the background of RFID authentication protocols to the outcomes of our work. The first chapter briefly introduces considerations of RFID authentication protocols and previews some issues. The rest of this thesis is organized as follows.

Chapter 2 introduces some fundamental backgrounds on RFID protocols and cryptographic primitives. We give the definition of RFID tag authentication protocols and several typical attacks. We present tools related to this thesis, including bilinear maps, cryptographic hash functions, Merkle trees, forking lemma, complexity assumptions and random oracles.

Chapter 3 focuses on the strong privacy of RFID tag authentication. We review a recent proposed protocol and show two common security problems of existing protocols. We then introduce a new RFID authentication protocol which achieves wide-strong privacy.

Chapter 4 addresses a problem of RFID ownership transfer protocols. In the chapter, we analyze limitations of current ownership transfer protocols and describe a system model with transfer chain. We define two types of adversaries and present the corresponding security models. An RFID ownership transfer protocol which is provably secure will be proposed.

Chapter 5 considers the shared tag ownership transfer. We introduce a scenario of tag ownership sharing and a system model without TTP. Some security requirements and challenges are illustrated and a security model is given. We propose the first shared ownership transfer protocol for two-party system model and discuss the hardware cost of tags.

Chapter 6 considers a new relationship between RFID tags, readers and the back-end server. We describe a system model where all entities are relatively independent and introduce the privacy of readers. We address some security and privacy issues and propose three authorized RFID authentication protocols. The proposed protocols have different features that adapt to different environments.

Chapter 7 introduces a concept of yoking-group proofs. We review yoking-proof

protocols and grouping proof protocols and show some restrictions. We extend these two concepts and discuss their relationship. A novel system model and an anonymous yoking-group proof protocol are also presented.

Chapter 8 concludes this thesis.

Chapter 2

Background

This chapter provides the background related to this thesis. We give the definition of RFID authentication protocols and review some attacks, cryptographic tools and complexity assumptions.

2.1 RFID Authentication Protocols

An RFID protocol defines a set of rules performed between tag, reader and back-end server. An essential purpose of RFID protocols is to authenticate tags that it is so-called *RFID tag authentication*. There are main requirements of tag authentication protocols: 1) Accept a legitimate tag if the tag's valid record is stored on back-end database, otherwise reject; 2) Prevent an unauthorized third party from obtaining tag's identity; 3) Provide tag untraceability that is the location of a tag is sensitive to adversaries. A typical RFID tag authentication protocol consists of four algorithms which are Setup, SendTag, SendReader and Verify.

- Setup: It takes as input a security parameter k, initiates a server S, readers $\{R_i\}$ and creates tags $\{T_i\}$, where

$$(S, \{R_i\}, \{T_i\}) \leftarrow \mathsf{Setup}(k).$$

- SendTag: It takes as input a message m and outputs a challenge/request c, where

$$c \leftarrow \mathsf{SendTag}(m),$$

  and then c is sent from R_i to T_i.

- SendReader: It takes as input a message m and tag's secret parameters tsp, outputs a response m', where

$$m' \leftarrow \mathsf{SendReader}(m, tsp),$$

  and then m' is sent from T_i to R_i.

- Verify: It takes as input a tag's response m and server's secret parameters ssp, outputs 1 if the tag is valid, otherwise it outputs 0.

$$\{0, 1\} \leftarrow \mathsf{Verify}(m, ssp).$$

A reader usually launches a session by using SendTag algorithm. Upon receiving the request, the tag uses its secret data, such as the state information and secret keys, to generate a response m'. Note that SendTag and SendReader algorithms can be repeated several times in this phase. Upon receiving the tag's response, the reader sends it to the back-end server through a secure channel. The server then runs Verify and checks the validity of the tag. We say that a tag is authenticated if its information is in the database. During the authentication, messages which contain tag's information should only be verifiable by the server.

RFID Mutual Authentication Protocols. Some RFID authentication protocols provide mutual authentication where tags and readers can mutually check the validity. The reader authentication is similar to tag authentication, while the reader authentication is started by a tag. A reader needs to show the knowledge of the secret, such as the tag's key and generates correct responses. The tag accepts the reader if the response is valid. In symmetric-key based mutual authentication protocols [PV08, CYK11, Pir11], a reader authentication normally starts after a successful tag authentication. One reason to this is that only if the reader knows the tag's identity, it can use the shared secret to convince the tag of its validity. It is otherwise hard to find the correct secret since a tag is usually anonymous prior to valid tag authentication. As a solution, public key based mutual authentication protocols allow reader authentication to be performed before tag authentication [HPP14].

2.2 Attacks on RFID Protocols

RFID protocols are vulnerable to various attacks due to different reasons. In terms of communication channel, RFID tag and reader interact in wireless environments allowing an adversary to eavesdrop the communication. Eavesdropping is considered as an elementary ability to collect information which can be used in further operations. An attacker is also allowed to actively query tags to obtain responses. Apart from gathering information, a strong adversary can corrupt tags and manipulate messages, such as interception and modification. We briefly introduce several typical attacks on RFID protocols as follows.

Replay Attack. It is a simple but effective attack which does not require any secret information. In this attack, an adversary firstly eavesdrops the interaction between tag and reader to collect transmitted messages. The adversary later can trivially resend those messages to achieve a goal like passing tag authentication.

There are two approaches to replay messages based on different purposes. One is to replay a tag's response to a reader when an adversary attempts to impersonate the tag. For instance, [OSK03] is a 2-pass protocol where a reader only sends as

query a "request" instead of a fresh challenge. An attacker can obtain the next valid tag response by sending the same request and replay it to the reader later on. The replay attack can also be used to trace a tag. A protocol is likely to be attacked if it adopts deterministic functions, such as symmetric-key encryption schemes and hash functions, without key update or fresh nonces. [FDW04] is an example where an attacker can replay a reader's challenge to the tag then trace its location. A solution to overcome replay attacks is for a tag to take fresh values as input to generate responses.

Desynchronization Attack. It is an important attack on stateful protocols which need to synchronize the shared information between tag and server in each successful session. Generally, an adversary attempts to cheat one entity that the other has successfully updated the state. If a legitimate tag is desynchronized with the server, it can no longer be authenticated and the denial-of-service (DoS) attack is occurred.

A low-cost tag is vulnerable to desynchronization attack if there is no reader authentication. For example, a tag in protocol [OSK04] updates its secret key without the verification of queries. An adversary can continuously query the tag and let it exceed the maximum number of tag authentication. Any future tag response then will be rejected and the server cannot re-synchronize with the tag. Therefore, a reader authentication is needed before tag state updates.

In another attack model, an adversary needs to alter messages exchanged between reader and tag. A protocol which employs reader authentication usually changes the state on server and tag after a successful tag authentication and reader authentication, respectively. Consider that an adversary modifies the reader's response and makes it invalid. The response will be rejected and the tag remains the current state. If the server has updated the state, then it is desynchronized with the tag. Someone may advise that a tag sends reader an acknowledgment to inform the key update. However, the acknowledgment can also be modified that it prevents the server from updating keys. Many protocols [SM08, FMTRCRDF11a, ZZLW10] adopt the solution which keeps both *old key* and *new key* on server. The server and tag can then be re-synchronized in the next authentication if they have been attacked.

Man-in-the-Middle (MITM) Attack. It is a powerful attack against many RFID authentication protocols. In this attack, an adversary engages the interaction between tag and reader without being detected. For each exchanged message, the adversary can manipulate it in some manners, such as redirection and modification.

Hopper and Blum [HB01] proposed a lightweight RFID authentication protocol called HB protocol which only requires a tag to perform an inner product over $GF(2)$. Juels and Weis [JW05] later found flaws in the HB protocol and introduced the HB$^+$

protocol as a solution. However, the protocol is vulnerable to an active MITM attack [GRS05]. Roughly speaking, an adversary modifies one bit of the reader's challenge during tag authentication. According to the result of authentication, the adversary can determine whether the corresponding bit of the tag's secret key is 1. The shared secret keys then can be revealed by repeating this operation enough times.

A requirement of tag authentication is that the tag must be in close physical proximity to the reader during the authentication. Some adversaries use MITM attack to make a distance fraud which forges the distance information between reader and tag. In this case, an adversary redirects messages and convinces the reader of a tag's false location. MITM attack is also referred as the *relay attack*. Kfir and Wool [KW05] then demonstrated the possibility of relay attack in RFID protocols. Distance bounding protocols [BMV13a, BMV13b, BC93, HK05, One12, BMV13c] which measure the distance between tag and reader are considered as solutions to this attack.

Side-channel Attacks. Compared with the above attacks, side-channel attacks exploit physical information, such as timing information and power consumption, to break protocols. Different input of a protocol may cause distinct features of power consumption based on the hardware implementation of tags. For instance, an input bit 1 usually requires more calculations than the value 0. It indicates the difference related to the computational cost and an adversary is possible to recover the input from obtained information. To resist such attacks, a secure hardware implementation or obfuscating computation techniques are needed. These topics are important but they are out of scope for our research.

2.3 Cryptographic Tools

2.3.1 Cryptographic Hash Functions

Cryptographic hash functions [RS04, Riv92, ZPS92, KR00, AdM04] are used to achieve several information security purposes, such as data integrity checks and authentication. In symmetric-key based RFID authentication protocols, hash functions are considered to be adequate tools to protect the communication. Although the hardware cost of hash functions is questionable to some low-cost tags, many secure lightweight protocols use cryptographic hash functions. For example, the hash lock protocol [WSRE03] and its variants [Lee10, Khe13] are based on secure hash functions.

A hash function $H : \{0,1\}^* \rightarrow \{0,1\}^l$ is a deterministic function that takes as input an arbitrary length bit string and outputs a fixed l-bit string which is called a hash value. One feature of hash functions is that the hash value y can be efficiently

computed from a given input string x. We say that H is a secure cryptographic hash funtion if it satisfies the following requirements.

- **Pre-image resistance**: A hash function H is one-way (pre-image resistance) if given a hash value y, it is computationally infeasible to find an input x, such that $H(x) = y$.
- **Second pre-image resistance**: A hash function H is second pre-image resistance if given an input x_1, it is computationally infeasible to find another input $x_2 \neq x_1$, such that $H(x_2) = H(x_1)$ [NY89].
- **Collision resistance**: A hash function H is collision resistance if it is computationally infeasible to find two different inputs (x_1, x_2), such that $H(x_1) = H(x_2)$ [Dam87, BR97].

2.3.2 Merkle Trees

A Merkle tree is a tree structure based on hash functions and it is also named *hash tree*. There are various applications of Merkle trees, such as Merkle signature scheme [Mer89], data storage and integrity checks in cloud computing [HHPS11]. Practically, binary Merkle trees are normally applied in protocols. We give the definition of a binary Merkle tree as follows.

Definition 2.1 (Merkle Tree)**.** *A binary Merkle tree based on a hash function H is a binary tree which satisfies the following conditions:*

- *For each label L of a leaf, there is a preimage m, such that $L = H(m)$.*
- *For each label N of an intermediate node, it is a hashing of its left L_1 and right L_2 children, such that $N = H(L_1, L_2)$.*

Given the root of a Merkle tree, both leaves and intermediate nodes are deterministic in a specific order. If the underlying cryptographic hash function is secure, there is a negligible probability to generate another Merkle tree which has the same root value. A root value then can be used to represent the relationship between a Merkle tree and the data. In addition, a sibling path which is as following shows the significant knowledge of a Merkle tree.

Definition 2.2 (Sibling Path)**.** *We say that a path $\mathcal{P} = (L_1, N_0, N_1, \cdots, N_i)$ is a sibling path of a leaf L_1 if it consists of labels of all siblings of nodes on the path from L_1 to the root, where i is the height of the Merkle tree, and the root value of Merkle tree is the same as the value computed on $\mathcal{P}$.*

2.3.3 Random Oracles

Many cryptographic protocols employ a powerful tool named *random oracle* in their security reductions. Random oracles were introduced by Bellare and Rogaway [BR93]. They refer to an ideal cryptographic hash function as a random oracle which outputs *uniform* and *truly random* values. In a security reduction, a hash function can be replaced by a random oracle if it is needed. Then, the challenger who controls random oracles simulates corresponding outputs for queries issued by an adversary. According to the truly randomness of oracle outputs, an adversary cannot detect any difference and the simulation is ideal. In hash-based RFID protocols, the random oracle is a tool if we need to formally analyze the security of protocols. However, it is clear that truly random hash values are infeasible in real applications. A protocol which is provably secure with random oracles may be insecure in practice [CGH04]. Nevertheless, it is worthwhile to evaluate the security of RFID protocols with random oracles (or other oracles), otherwise strong security is extremely hard to achieve.

2.3.4 Forking Lemma

The concept of *forking lemma* depicted in Lemma 2.1 was firstly introduced by Pointcheval and Stern [PS96]. The idea is adopted in security reduction of digital signatures where random oracles are used. Roughly speaking, forking lemma utilizes the oracle replay attack which applies the same random tape in two different random oracles and thus obtains different outputs regarding the same input. If there is an adversary who can break the scheme, the challenger can solve the underlying hard problem by using the forking lemma. Indeed, a precondition of using forking lemma is that two different hash functions can be applied in a scheme. On the other hand, a hash function is usually fixed in a scheme and it cannot be alternated in practice. Therefore, random oracles which can be manipulated in the simulation have to be employed to solve this problem. Bellare and Neven [BN06] presented a generalized version of forking lemma and we review the definition as below.

Lemma 2.1. *Let $\mathcal{A}$ be a probabilistic polynomial time (PPT) adversary, given public parameters as input, if $\mathcal{A}$ has non-negligible probability to find a tuple (m, r, σ, h), where σ is a valid signature of a message m and h is the oracle output of a random tape r, then $\mathcal{A}$ as in Figure 2.1 has non-negligible probability to find another valid tuple (m, r, σ', h') with the same random tape r and different random oracles such that $h \neq h'$.*

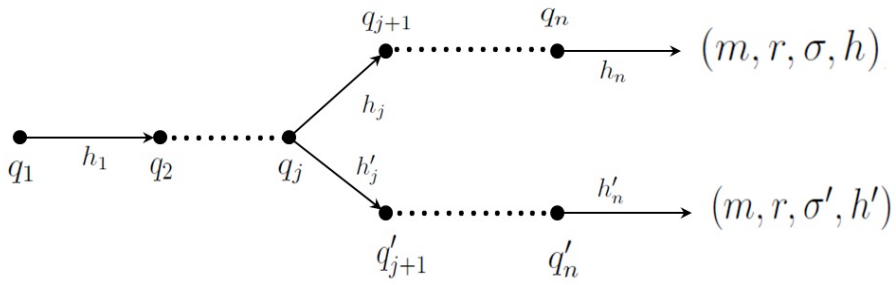

Figure 2.1: Forking answers to random oracle queries.

2.3.5 Bilinear Maps

Bilinear maps (pairings) was introduced by Menezes, Okamoto and Vanstone [MOV93] to attack elliptic curve based cryptographic schemes. Some properties of bilinear maps are found that they can be used to construct efficient cryptographic schemes. Later on, diverse types of digital signatures and encryption schemes were proposed, such as the identity-based cryptography. Two important bilinear pairings used in public key cryptography are Weil pairing [BF01] and Tate pairing [BLS04a, FMR99]. Moreover, PKC primitives rely on cyclic groups which can be either additive or multiplicative. A difference between two groups is that the elliptic curve requires a shorter element length to achieve the same security of traditional multiplicative groups over the finite field.

The size of parameters plays a crucial role in RFID protocols so that we need to reduce the length to the minimum. ECC has been adopted in many public key based RFID authentication protocols [LIM, vDR11, BSSV11] and we give the definition of bilinear maps as follows.

Definition 2.3. *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two additive cyclic groups and $\mathbb{G}_T$ be a multiplicative cyclic groups of same large prime order p. P, Q are two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. The map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear mapping and $(P, Q, p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$ is a bilinear group. Specifically, we say that a bilinear group is symmetric if $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G} = \mathbb{G}_1 = \mathbb{G}_2$. Some properties of bilinear maps are as follows:*

- **Bilinearity:** *For all $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ and for all $a, b \in \mathbb{Z}_p^*$, we have the equation $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.*
- **Non-degeneracy:** *For all $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, if P, Q are respectively a generator of $\mathbb{G}_1$ and $\mathbb{G}_2$, we have $\hat{e}(P, Q) \neq 1$ is a generator of $\mathbb{G}_T$.*
- **Efficiency:** *There is an efficient algorithm to calculate $\hat{e}(P, Q)$ for all $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$.*

2.3.6 Digital Signatures

The notion of digital signatures was envisioned by Diffie and Hellman [DH76] in 1976. In public key cryptography, the signer who holds the private key can generate a signature of a message by a one-way trapdoor function. A receiver can check the validity of a signature via the signer's public key. A digital signature is used to be a proof of the authorship of a message. The signature unforgeability makes digital signatures to be an potential solution of RFID authentication. Unfortunately, traditional digital signatures described as in Definition 2.4 cannot provide the confidentiality that unauthorized people are allowed to verify the signature and identify the tag. A signature cannot be stored in a tag and transmitted directly during the tag authentication either. In Chapter 5, we show that a traditional digital signature scheme can be integrated in a hybrid RFID protocol.

Definition 2.4. *A digital signature scheme consists of three algorithms:* KeyGen*,* Sign *and* Verify*.*

- KeyGen*: It takes as input a security parameter k, outputs a pair of private and public keys (sk, pk), where*

$$(sk, pk) \leftarrow \mathsf{KeyGen}(k).$$

- Sign*: It takes as input a message m and a private key sk, outputs a signature σ, where*

$$\sigma \leftarrow \mathsf{Sign}(m, sk).$$

- Verify*: It takes as input a message m, a signature σ and the signer's public key pk, outputs* 1 *if the signature is valid, otherwise outputs* 0*.*

$$\{1, 0\} \leftarrow \mathsf{Verify}(m, \sigma, pk).$$

2.4 Complexity Assumptions

In this section, we introduce some complexity assumptions used in this thesis. Note that definitions in this section are described in elliptic curves.

Elliptic Curve Diffie-Hellman (ECDH) Assumption. Let P be a generator of additive cyclic group $\mathbb{G}$ of order p. Given a tuple $< P, aP, bP >$, where $a, b \xleftarrow{\$} \mathbb{Z}_p^*$, the ECHD problem is to output $abP \in \mathbb{G}$. We say that the (ϵ, t)-ECDH assumption holds in group $\mathbb{G}$, if no t-time algorithm $\mathcal{A}$ can solve the ECDH problem in $\mathbb{G}$ with advantage at least ϵ.

Elliptic Curve k+1 Exponent (k+1-Exponent) Assumption. Let P be a generator of additive cyclic group $\mathbb{G}$ of order p. Given k+1 values $< P, aP, a^2P, \ldots, a^kP >$, where k is an integer and $a \xleftarrow{\$} \mathbb{Z}_p^*$, the elliptic curve $k+1$ exponent problem is to compute $a^{k+1}P$. We say that the (ϵ, t)-$k+1$Exponent assumption holds in group $\mathbb{G}$, if no t-time algorithm $\mathcal{A}$ can solve the $k+1$Exponent problem in $\mathbb{G}$ with advantage at least ϵ.

Oracle Diffie-Hellman Assumption (ODH) [ABR01]. Let P be a generator of additive cyclic group $\mathbb{G}$ of order p. Given aP, bP, a function $H : \{0,1\}^* \to \{0,1\}^l$ and an oracle $\mathcal{O} = H(bX)$, where $X \neq aP$, the advantage of an adversary $\mathcal{A}$ in violating the ODH assumption is

$$Adv^{odh}_{\mathcal{A},H} = \left|\Pr\left[a, b : \mathcal{A}^{\mathcal{O}}(aP, bP, H(abP)) = 1\right] - \Pr\left[a, b : \mathcal{A}^{\mathcal{O}}(aP, bP, t) = 1\right]\right|,$$

where $t \in \{0,1\}^l$. We say that the ODH assumption holds, if $Adv^{odh}_{\mathcal{A},H}$ is negligible.

Chapter 3

Wide Strong Private RFID Authentication

RFID tag privacy is an important issue to RFID authentication protocols. To date, there have been several attempts to achieve wide-strong privacy by using zero-knowledge protocols. Unfortunately, most of them are vulnerable to strong adversaries. In this chapter, we firstly take as an example a recent ECC-based RFID authentication scheme with ID-verifier transfer protocol which was claimed to be secure against many attacks and satisfies essential security requirements of RFID systems. We then demonstrate that the protocol suffers from several attacks, in contrast to its claims. To solve the problems, we propose a new protocol which is provably secure in the desired security model. We also show an example of extracting a formal privacy model from security requirements. Some techniques to solve similar problems are introduced and two concrete constructions of wide-strong private authentication protocols are given. Finally, the formal privacy analysis shows the proposed zero-knowledge based protocols offer wide-strong privacy. The security analysis in Section 3.2 was presented in [LMS+15] and the wide-strong private RFID authentication scheme was proposed in [LMS+13].

3.1 Introduction

RFID has been widely adopted to the identification of objects. The RFID technology has many advantages compared with the barcode based approaches. In practice, RFID systems have many applications, such as Internet of Things (IoT), supply chain, warehouse management and car tracking. However, RFID tags have very limited computation and storage resources and are usually not tamper-resistant. Thus, an attacker could physically access a tag and collect its internal state. Also, the communication between a tag and a reader is in a wireless environment. An attacker can identify a compromised tag by using the collected information. Therefore, we have to prevent a tag's privacy even the secret of the tag has been leaked.

A typical RFID system comprises RFID readers, RFID tags and a backend server. Usually, the communication channel between a backend server and a reader can be secured by traditional security techniques, while the channel between a reader and a tag poses a challenge due to the low computation and storage capacity of tags. This is one reason that many research works on RFID focus on the tag privacy pro-

tection. Moreover, privacy models are needed to evaluate the privacy of a proposed protocol. Vaudenay [Vau07] introduced the strong privacy model which captures a number of RFID privacy cases, which are corresponding to eight classes with respect to eight different privacy levels from weak to strong. The strongest level is the wide-strong privacy. Later, Ng, Susilo, Mu and Safavi-Nain [NSMSN08] refined the Vaudenay's model and claimed that wide-strong privacy was possible. Based on the Bohli-Pashalidis' model [BP09, BP11] and Vaudenay's model, Hermans, Pashalidis, Vercautern and Preneel [HPVP11] proposed a new practical RFID privacy model which relies on the indistinguishability of tags.

Wide-strong privacy is achievable by using asymmetric key cryptography [Vau07, NSMSN08]. ECC is feasible to be implemented on RFID tags [HWF08, LSBV08, KSLE09] as it requires relatively affordable computational operations. An important contribution of ECC-based protocols is to enhance the security of the protocol. Normally, asymmetric key based protocols can achieve higher security level than symmetric-key based protocols.

ECC-based RFID protocols can be constructed with either encryption schemes or signature schemes. Vaudenay [Vau07] proved that any indistinguishability against chosen-plaintext attacks (IND-CPA) secure public key encryption scheme can provide narrow-strong security. Later, Hermans, Pashalidis, Vercauteren, and Preneel [HPVP11] claimed that wide-strong privacy requires the underlying public key encryption scheme is indistinguishability against adaptive chosen-ciphertext attacks (IND-CCA2) secure. In fact, a wide-strong private RFID authentication protocol equals an IND-CCA2 secure encryption scheme. A concrete construction which is wide-strong secure was proposed by Deursen and Radomirovi [vDR11]. The proposed protocol is obtained from Cramer-Shoup encryption [CS98] that the efficiency needs to be improved.

The digital signature is an alternative cryptographic primitive in PKC. However, a traditional digital signature is hard to preserve the tag's privacy as a signature is publicly verifiable. That is, anyone can trace a tag by checking the validity of the signature. Fortunately, digital signatures, such as strong designated verifier signatures [JSI96], can be obtained by applying IND-CCA2 encryption schemes. Thus, it is possible to construct a wide-strong private identification protocol based on strong designated verifier signature schemes. Many ECC-based RFID authentication protocols [TB06, LBV08, LBV09, LBSV10b, BSSV11, PH12] were proposed. However, most of them have been broken later in [FHV10, LBV08, vDR09, vDR10, BCI08a].

Recently, Liao and Hsiao [LH13] proposed an interesting ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol for IoT. The protocol provides mutual authentication between sever (verifier) and tag (prover). They described the adversary model and security issues and claimed that their scheme

is secure against various types of attacks, such as forward security and ID-verifier confidentiality.

Related Work. Lee, Batina and Verbauwhede [LBV08] proposed the first ECC-based RFID authentication protocol called Elliptic Curve Discrete Logarithm Problem (ECDLP) based Randomized Access Control (EC-RAC). The protocol is based on the hardness of the elliptic curve discrete logarithm problem. Indeed, the authors adopt the Schnorr identification scheme [Sch89] as a building block. Due to the linearity of the protocol construction, the protocol was broken in both [vDR08] and [BCI08a]. Later, the second version (EC-RACII) was proposed in [LBV09]. The authors claimed that EC-RACII is untraceable against a strong adversary. However, Deursen [vDR09, vDR10] pointed out that it is vulnerable under the conventional man-in-the-middle attack. The third version (EC-RACIII) [LBSV10b] was presented by Lee, Batina Singelée and Verbauwhede in 2010. In this chapter, they proposed two schemes: one was claimed as wide-strong privacy-preserving and the other was considered as wide-weak privacy-preserving. Although the protocols use a non-linear function to solve problems in EC-RAC and EC-RACII, Fan, Hermans and Vercauteren [FHV10] show that the proposed scheme is still insecure under the man-in-the-middle attack. The latest version EC-RACIV proposed in [LBSV10b] has also been broken later on. A summary of attacks to all versions of EC-RAC protocols was presented in [vDR10]. Finally, Deursen [vDR11] introduced a strong attack named *insider attack*. Generally, an attacker uses the knowledge of a compromised tag to violate the privacy or the security of other tags. Recently, an ECC-based mutual authentication protocol secure against wide-strong attackers was proposed by Peeters, Hermans and Fan [PHF13].

Organization of This Chapter. The rest of this chapter is organized as follows. In Section 3.2, we review Liao and Hsiao's protocol and shows attacks against the protocol. We propose the repaired protocol in Section 3.3 and we analyse the privacy of the proposed protocol in Section 3.4. Section 3.5 introduces a wide-strong privacy model and Section 3.6 present two wide-strong private protocols. A formal privacy analysis of the proposed protocols is given in Section 3.7. Section 3.8 concludes the chapter.

3.2 Liao and Hsiao's Protocol

In this section, we firstly review Liao and Hsiao's protocol [LH13] and its security requirements. Note that notions described in this section only applied in Liao and Hsiao's protocol and the repaired protocol. Then, we show attacks against the proposed protocol and give solutions with the security proof.

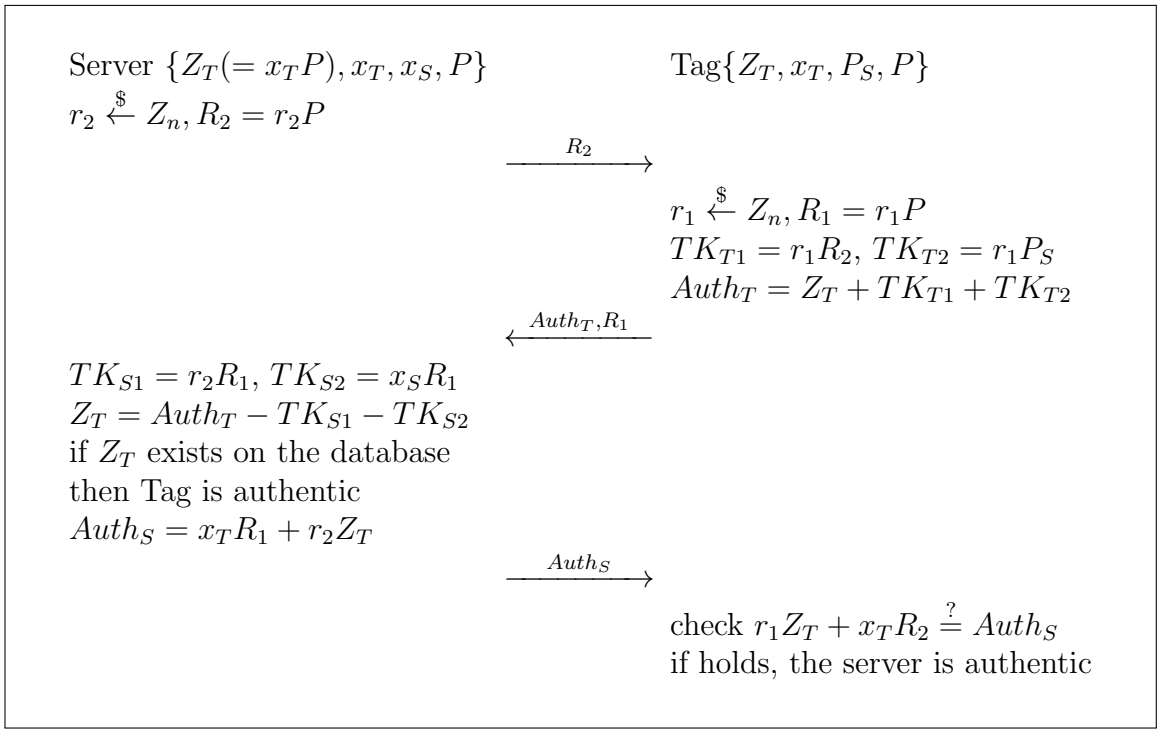

Figure 3.2: Liao and Hsiao's proposed ECC-based protocol.

3.2.1 Description

$F(q)$:	the finite field over q, where q is a prime.
$\mathbb{G}$:	an elliptic curve with prime order n.
(a, b):	the parameters of $\mathbb{G}$ over $F(q)$.
P:	a generator of $\mathbb{G}$.
n:	the order of generator P.
h:	cofactor, such that $h = \#F(q)/n$.

Figure 3.1: Notion of symbols.

Liao and Hsiao's protocol consists of two phases: namely *setup phase* and *authentication phase*. In the setup phase, the server generates elliptic curve domain parameters $D = \{q, a, b, P, n, h\}$ and initiate tags. The notion of symbols are represented as in Figure 3.1. First, the server randomly chooses his private key $x_S \xleftarrow{\$} Z_n$ and sets the corresponding public key P_S as $P_S = x_S P$. Then, the server randomly picks $x_T \xleftarrow{\$} Z_n$ and sets the tag's public/private key pair as $(x_T P, x_T)$. Z_T, where $Z_T = x_T P$, is also called the tag's ID-verifier. Finally, the server adds the record $\{Z_T, x_T\}$ of the tag into the database and stores $\{Z_T, x_T, P_S, D\}$ into the tag's memory.

The proposed protocol has three passes during the authentication phase. The protocol is depicted as in Figure 3.2. Firstly, the server randomly chooses a nonce $r_2 \xleftarrow{\$} Z_n$ and computes $R_2 = r_2 P$. R_2 is issued to the tag as a challenge. Upon

receiving the challenge, the tag randomly picks a nonce $r_1 \xleftarrow{\$} Z_n$ and computes $R_1 = r_1 P$. Then, it calculates two temporary Diffie-Hellman keys TK_{T1} and TK_{T2}, where $TK_{T1} = r_1 R_2, TK_{T2} = r_1 P_S$. At last, the tag uses the temporary keys to encrypt its ID-verifier Z_T as

$$Auth_T = Z_T + TK_{T1} + TK_{T2},$$

and sends $Auth_T$ to the server as a response. After the server receives the tag's response, it computes the temporary keys $TK_{S1} = r_2 R_1$ and $TK_{S2} = x_S R_1$ and decrypts as

$$Z_T = Auth_T - TK_{S1} - TK_{S2}.$$

If Z_T appears in the database, the tag authentication is successful. Then, the server replies to the tag

$$Auth_S = x_T R_1 + r_2 Z_T.$$

The server is authentic if the following equation holds.

$$r_1 Z_T + x_T R_2 = Auth_S.$$

3.2.2 Security Assumptions

Liao and Hsiao made some assumptions in order to analyse the security of their proposed protocol. Under the following assumptions, Liao and Hsiao claimed that the protocol is secure against location tracking attack and provides forward security.

- A1: r_1 is fresh in different sessions.
- A2: r_2 is fresh in different sessions.
- A3: x_S is known to the reader only.
- A4: Z_T and x_T are known to the tag and the server only.
- A5: The tag is corruptible where common parameters and P_S can be known to the attacker.

To capture the operations defined in [LH13], we define the following oracles which are employed in the privacy experiment. Clearly, privacy models in this section only interpret security assumptions and requirements defined above.

- $\mathsf{SetupTag}(ID) \rightarrow T$: Taking as input a tag's identity ID, the oracle generates a new tag T and registers it to the database.

Experiment $\mathbf{Exp}^{private}_{\mathcal{A},\mathcal{S}}[s,k]$:

- **Setup**: The challenger $\mathcal{C}$ runs the algorithm SetupServer [LH13] to generate public parameters *params* and returns to the adversary $\mathcal{A}$. It initializes the server S.
- **Learning Phase**: $\mathcal{A}$ can query SetupTag, SendTag, SendReader and CorruptCom oracles to $\mathcal{C}$ and outputs $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$.
- **Extract Phase**: $\mathcal{A}$ outputs an ID-verifier Z^*_T.

Exp outputs *success* if $Z^*_T \in \mathcal{T}$.

Figure 3.3: Privacy experiment of tag ID-verifier confidentiality and anonymity.

- $\mathsf{SendTag}(T, m) \rightarrow m'$: Taking as input a tag T and a message m, the oracle outputs the tag's response m'.
- $\mathsf{SendReader}(\pi, m) \rightarrow m'$: Taking as input a session π and a message m', the oracle sends m to the reader in session π and outputs the reader's response m'. If π is not activated, it outputs $\perp$.
- $\mathsf{Launch}(T) \rightarrow \pi$: Taking as input a tag T, the oracle invokes SendTag and SendReader to output a complete session π.
- $\mathsf{CorruptCom}(T) \rightarrow ComPara$: Taking as input a tag T, the oracle outputs T's inside common parameters $ComPara$.
- $\mathsf{CorruptTag}(T) \rightarrow sk$: Taking as input a tag T, the oracle outputs T's secret information. Note that the oracle destructs the tag and T cannot be involved in the future communications.

The experiment $\mathbf{Exp}^{private}_{\mathcal{A},\mathcal{S}}[s,k]$ depicted in Figure 3.3 represents an adversary $\mathcal{A}$ who attempts to break a tag's confidentiality and anonymity. In the experiment, $\mathcal{A}$ can issue s oracle queries based on k tags to the challenger and then outputs an identifier Z^*_T. $\mathcal{A}$ succeeds if Z^*_T is in the database.

Definition 3.1. *An RFID authentication protocol is confidential and anonymous if any $\mathcal{A}$ who succeeds in $\mathbf{Exp}^{private}_{\mathcal{A},\mathcal{S}}[s,k]$ has advantage*

$$\Pr[success \leftarrow \mathbf{Exp}^{private}_{\mathcal{A},\mathcal{S}}[s,k]] \leq \epsilon,$$

where ϵ is negligible in k.

The experiment $\mathbf{Exp}^{forward}_{\mathcal{A},\mathcal{S}}[s,k]$ depicted in Figure 3.4 represents an adversary $\mathcal{A}$ who attempts to break a tag's forward security. In the experiment, $\mathcal{A}$ can issue

Experiment $\mathbf{Exp}^{forward}_{\mathcal{A},\mathcal{S}}[s,k]$:

- **Setup**: The challenger $\mathcal{C}$ runs the algorithm SetupServer to generate public parameters *params* and returns to the adversary $\mathcal{A}$. It initializes the server S.
- **Learning Phase**: $\mathcal{A}$ can query SetupTag, Launch and CorruptTag oracles to $\mathcal{C}$ and outputs $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ and sessions $\Sigma = \{\pi_1, \pi_2, \ldots, \pi_n\}$.
- **Challenge Phase**:
  - $\mathcal{A}$ chooses a target tag $T^* \in \mathcal{T}$ and queries CorruptTag(T^*) to obtain T^*'s secret information sk^*.
  - $\mathcal{A}$ can query oracles as in the learning phase.
- **Guess**: $\mathcal{A}$ outputs a session $\pi^* \in \Sigma$.

Exp outputs *success* if $T^* \in \pi^*$.

Figure 3.4: Experiment of tag forward security.

s oracle queries based on k tags to the challenger and then outputs a session π^*. $\mathcal{A}$ succeeds if π^* is a session between the target tag and the server.

Definition 3.2. *An RFID authentication protocol is forward secure if any $\mathcal{A}$ who succeeds in $\mathbf{Exp}^{forward}_{\mathcal{A},\mathcal{S}}[s,k]$ has advantage*

$$\Pr[success \leftarrow \mathbf{Exp}^{forward}_{\mathcal{A},\mathcal{S}}[s,k]] \leq \epsilon,$$

where ϵ is negligible in k.

3.2.3 Security Analysis

In this section, we show that Liao and Hsiao's protocol is not secure against location tracking attack and hence, it does not provide forward security without changing the security assumptions.

3.2.3.1 Attack against Forward Security

Theorem 3.1. *In Liao and Hsiao's protocol, given the secret key x_T of target tag T, an attacker is able to break the forward security with advantage $\Pr[\bar{E}] = 1 - \frac{1}{n}$, where n is a domain parameter in D.*

Proof. Suppose the tag T's secret key x_T is leaked to an attacker as SR5 in [LH13], the attacker breaks the forward security as follows.

Given a past communication record $\{Auth_S, Auth_T, R_1, R_2\}$ of a mutual authentication, attacker checks

$$x_T^{-1} Auth_S \stackrel{?}{=} R_1 + R_2. \tag{3.1}$$

If this holds, it means that the communication is between the server and tag T. We show the correctness of equation (1) as below.

$$\begin{aligned} x_T^{-1} Auth_S &= x_T^{-1}(x_T R_1 + r_2 Z_T) \\ &= x_T^{-1}(x_T R_1 + r_2 x_T P) \\ &= x_T^{-1} x_T (R_1 + R_2) \\ &= R_1 + R_2. \end{aligned}$$

Assume there is a past communication record $\{Auth'_S, Auth'_T, R'_1, R'_2\}$ of tag T', where the tag's keys are $\{Z'_T, x'_T\}$. Let $R = R_1 + R_2$ and $R' = R'_1 + R'_2$, the event E is that the above test fails. It occurs when we have

$$Auth_S \neq Auth'_S \wedge x_T \neq x'_T \wedge R = R'.$$

Since that R_1, R_2, R'_1, R'_2 are random elements, R and R' can be considered as two random values. We have the probability of E occurs is $\frac{1}{n}$, which is negligible. Hence, the attacker succeeds in the attack with high probability

$$\Pr[\bar{E}] = 1 - \frac{1}{n}.$$

□

3.2.3.2 Attack against ID-Verifier Confidentiality

The authors claimed that the ID-verifier of a tag is anonymous to anyone except the server and tag. However, we show that the attacker can extract the tag's ID-verifier without using any secret of the tag. Then, it is easy to trace the location of the tag with the ID-verifier[a].

Theorem 3.2. *In Liao and Hsiao's protocol, given common parameters $\{P, P_S\}$, the attacker is able to extract a tag T's ID-verifier Z_T.*

Proof. According to A5 in Section 3.2, we assume that the attacker knows $\{P, P_S\}$. Then, the attacker can reveal any tag's ID-verifier described as follows (depicted in Figure 3.5).

[a]We note that this attack has also been independently pointed out by Peeters and Hermans very recently [PH13].

Step 1: The attacker randomly chooses $k \xleftarrow{\$} Z_n$ and computes

$$R_2 = kP - P_S.$$

Then, he sends R_2 to a tag. Since k is a random number, R_2 is also a random element that the tag cannot realise any difference.

Step 2: Upon receiving R_2, the tag computes as follows,

$$r_1 \xleftarrow{\$} Z_n, \quad R_1 = r_1 P,$$
$$TK_{T1} = r_1 R_2, \quad TK_{T2} = r_1 P_S,$$
$$Auth_T = Z_T + TK_{T1} + TK_{T2}.$$

Then, it sends $\{Auth_T, R_1\}$ to the server.

Step 3: Upon receiving the response $\{Auth_T, R_1\}$ form the tag T, the attacker can calculate T's ID-verifier Z_T by computing

$$Z_T = Auth_T - kR_1. \tag{3.2}$$

Equation (3.2) is correct:

$$\begin{aligned} Auth_T - kR_1 &= Z_T + TK_{T1} + TK_{T2} - kR_1 \\ &= Z_T + r_1 R_2 + r_1 P_S - kR_1 \\ &= Z_T + r_1(kP - P_S) + r_1 P_S - kr_1 P \\ &= Z_T. \end{aligned}$$

□

3.3 The Revised Protocol

Liao and Hsiao's protocol [LH13] is insecure because the same random number r_1 is applied twice to generate tag's response. It is vulnerable since we can do the linear transformation to extract some information. The attack against ID-verifier confidentiality introduced in Section 3.2.3.2 is an example due to such situation. To resist this attack, we provide two solutions in this section. The tag corruption is a powerful active attack where it can disclose the tag's secret keys. Our attack against forward security exploits the secret keys of the tag to trace the tag's past communications. The repaired protocol, which resists the proposed attacks is depicted in Figure 3.6. Note that the repaired protocol intends to show the method of

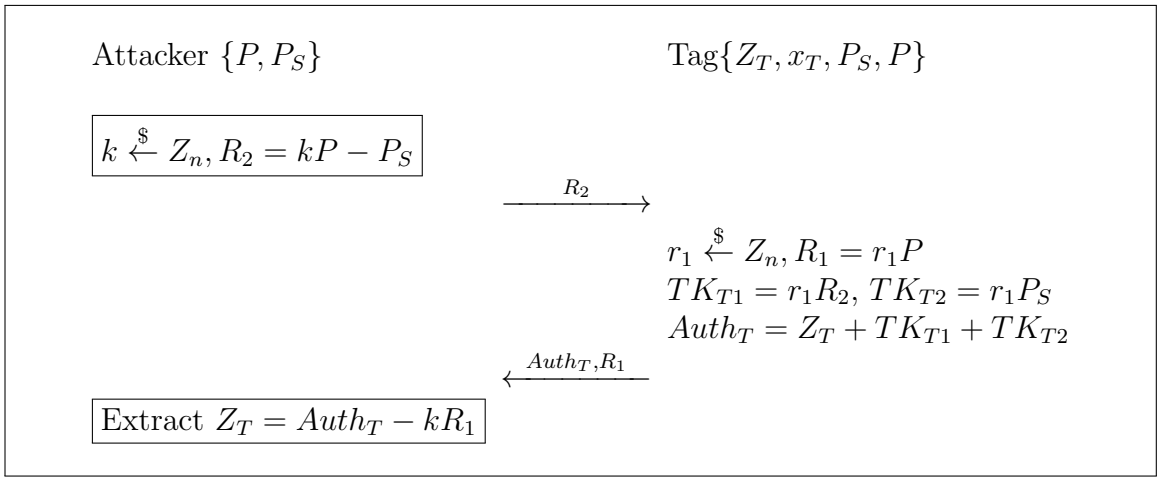

Figure 3.5: Attack against ID-verifier confidentiality.

fixing protocols, while it is not a strong private protocol. Section 3.6 presents two constructions of wide-strong private RFID authentication protocols.

3.3.1 Revised Attack against ID-verifier Confidentiality

There are two approaches with different natures to repair the protocol. Note that solutions in this section provide the general idea against similar linear attacks. Although the efficiency can be improved, the protocol has to be fully re-designed. Therefore, we prefer not to do it, as we want to keep the main feature of original protocol.

3.3.1.1 Double Random Numbers Based solution

To avoid the attack of ID-verifier confidentiality, a tag can simply chooses two different random numbers $r_1, r_1' \xleftarrow{\$} Z_n$ instead of using r_1 only. The tag computes:

$$R_1 = r_1 P, \quad R_1' = r_1' P,$$

$$TK_{T1} = r_1 R_2, \quad TK_{T2} = r_1' P_S,$$

$$Auth_T = Z_T + TK_{T1} + TK_{T2}.$$

Then, the tag responds $\{Auth_T, R_1, R_1'\}$ to the server. On the server side, it calculates TK_{S2} as $TK_{S_2} = x_S R_1'$.

This solution does not require any additional hardware implementations. Nevertheless, it needs to compute one more group element during the authentication. It also requires more memory to store R_1' and communication requirements are increased.

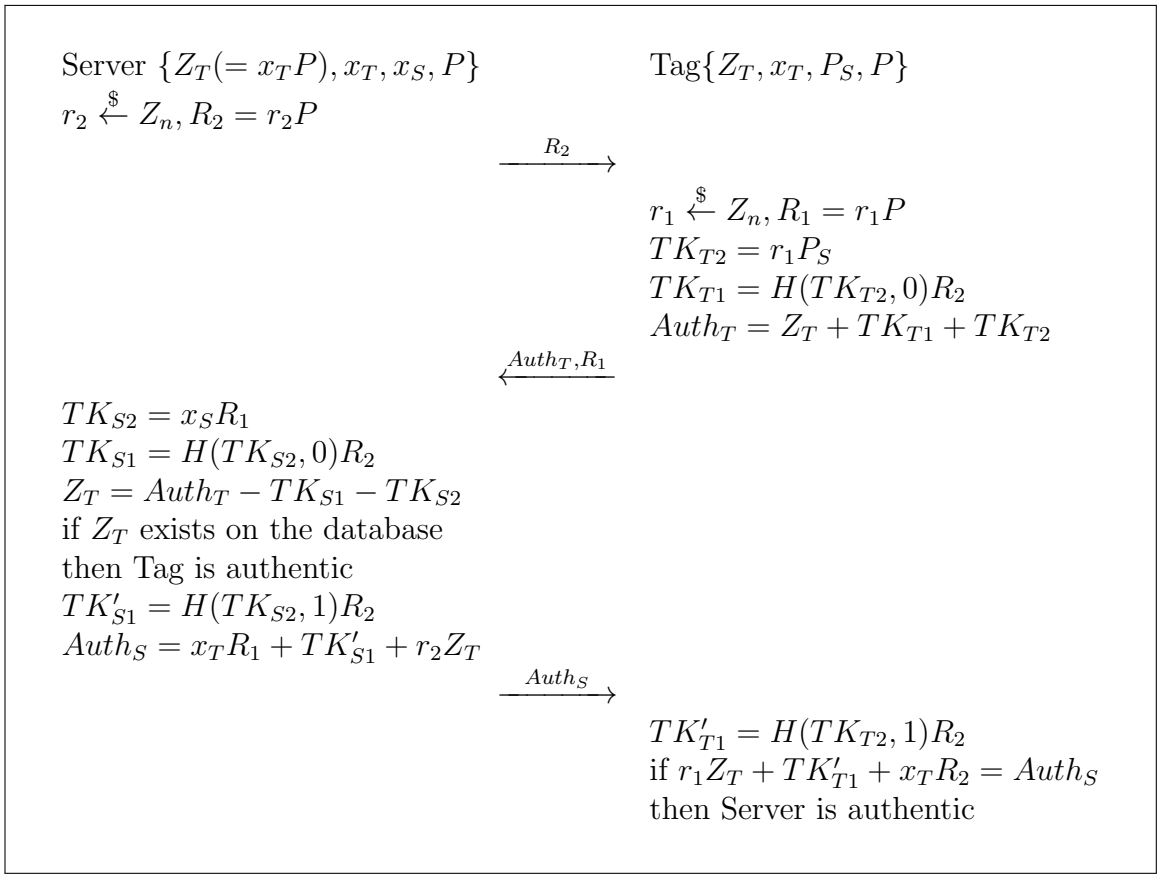

Figure 3.6: Repaired Protocol.

3.3.1.2 Non-linear Function Based Solution

Non-linear function is a tool to prevent attackers from performing linear transformation. For instance, we may employ a hash function. Let a hash function H be $H : \{0,1\}^* \rightarrow Z_n$, where $\mathbb{G}$ is an elliptic curve group. We firstly compute TK_{T2} as in protocol and then slightly modify the calculation of TK_{T1} as $TK_{T1} = H(TK_{T2}, 0)R_2$. In this approach, it avoids the reuse of random number r_1 and it is against the attack described as in Section 3.2.3.2.

3.3.2 Revised Attack against Forward Security

The adversary attacks the forward security using the secret keys of a tag. It is due to that the server generates a response by using the tag's secret keys only. Moreover, in Liao and Hsiao's protocol, the tag's secret keys are long-term keys in order to satisfy the availability property. Once the tag is compromised, its previous locations are traceable as described in Section 3.2.3.1. Our solution is that the server uses both its private key x_S and the tag's private key x_T to generate the response to tag.

For example, $Auth_S$ could be computed as

$$Auth_S = x_T R_1 + H(TK_{S2}, 1)R_2 + r_2 Z_T.$$

Correspondingly, the tag verifies the servers response by checking

$$r_1 Z_T + H(TK_{T2}, 1)R_2 + x_T R_2 \stackrel{?}{=} Auth_S.$$

3.4 Privacy Analysis of Repaired Protocol

Theorem 3.3. *Our protocol provides tag ID-verifier confidentiality and anonymity if ECDH problem is hard.*

Proof. Suppose that there is an adversary $\mathcal{A}$ who can (ϵ, q_H, t)-win the experiment $\mathbf{Exp}^{private}_{\mathcal{A},\mathcal{S}}[s,k]$. Let $\mathcal{A}$ has an advantage ϵ' to solve the ECDH problem.Given an instance (P, aP, bP), we can construct an algorithm $\mathcal{B}$ to find the solution abP of ECDH problem using the adversary $\mathcal{A}$. $\mathcal{B}$ interacts with $\mathcal{A}$ through the following oracles queries.

- Setup: $\mathcal{B}$ selects P as a generator of the additive cyclic group $\mathbb{G}$. Let the public key of the server be $P_S = aP$ and the private key of the reader be $x_S = a$, which is unknown to $\mathcal{B}$. $\mathcal{B}$ maintains the lists $L_H = \{< TK_{T2}, v, \{0,1\} >\}$, $L_S = \{< T, \pi, w, r, z >\}$ and a database of tags $\mathcal{T} = \{< ID, T, Z_T, x >\}$, which are initially empty.

- H Query: $\mathcal{A}$ issues H query on input (TK_{T2i}, c_i) at most q_H times. $\mathcal{B}$ outputs v_i if $< TK_{T2i}, v_i, c_i >$ is in the list L_H. Otherwise, $\mathcal{B}$ picks $v_i \stackrel{\$}{\leftarrow} \mathbb{Z}_n^*$ and sets $H(TK_{T2i}, c_i) = v_i$. Then, $\mathcal{B}$ outputs v_i and adds $< TK_{T2i}, v_i, c_i >$ into the list L_H.

- SetupTag Query: $\mathcal{A}$ issues the oracle query on input a tag's identity ID_i. $\mathcal{B}$ ignores the query if ID_i exists. Otherwise, $\mathcal{B}$ randomly chooses $x_i \stackrel{\$}{\leftarrow} \mathbb{Z}_n^*$ and computes $Z_{Ti} = x_i aP$. $\mathcal{B}$ sets $ComPara_i = (P_S, P, H)$ as T_i's common parameters. Then, $\mathcal{B}$ creates a new tag and sets $(Z_{Ti}, x_i a)$ as its public and private key pair, where $x_i a$ is unknown to $\mathcal{B}$. $\mathcal{B}$ outputs the tag T_i and adds $< ID_i, T_i, Z_{Ti}, x_i >$ into the database $\mathcal{T}$.

- CorruptCom Query: $\mathcal{A}$ issues the oracle query on input a tag T_i. If T_i is not in $\mathcal{T}$, $\mathcal{B}$ creates a new tag by running SetupTag Query. $\mathcal{B}$ then outputs the tag's common parameters $ComPara_i$.

- SendTag Query: $\mathcal{A}$ issues the oracle query on input a tag T_i and a message R_{2i}. $\mathcal{B}$ outputs $\perp$ If $T_i \notin \mathcal{T}$. Otherwise, $\mathcal{B}$ retrieves the referenced tag T_i's public key Z_i and x_i computes as follows.
  - Randomly selects $r_i \xleftarrow{\$} \mathbb{Z}_n^*$ and computes $R_{1i} = r_i P + bP$.
  - Let $H(r_{1i}P_S, 0) = v_i$, $\mathcal{B}$ randomly picks $w_i \xleftarrow{\$} \mathbb{Z}_n^*$ and sets

$$v_i = \frac{w_i - ab}{r_{2i}},$$

    where $R_{2i} = r_{2i}P$. Note that $\mathcal{B}$ does not need to compute ab, v_i, r_{2i}, we only assume that there is such a relationship.
  - Computes $Auth_{Ti} = Z_g + w_i P + r_i aP$ and sets $m_i' = (Auth_{Ti}, R_{1i})$, $\pi_i = (R_{2i}, m_i')$.

  If R_{2i} is not generated by $\mathcal{B}$, it adds $< T_i, \pi_i, w_i, r_i, \cdot >$ into the list L_S, otherwise $\mathcal{B}$ sets $z_i = r_{2i}$ and adds $< T_i, \pi_i, w_i, r_i, z_i >$ into L_S. Then, $\mathcal{B}$ outputs m_i'. We show that the simulation is perfect as

$$\begin{aligned} Auth_{Ti} &= Z_g + w_i P + r_i aP \\ &= Z_g + w_i P - abP + r_i aP + abP \\ &= Z_g + \frac{w_i - ab}{r_{2i}} R_{2i} + (r_i + b)aP \\ &= Z_g + TK_{T1i} + TK_{T2i} \end{aligned}$$

- SentReader Query: $\mathcal{A}$ issues the oracle query on input a session π_i and a message $m_i = (Auth_{Ti}, R_{1i}, R_{2i})$. $\mathcal{B}$ outputs $\perp$ if $(R_{2i}, \cdot)$ is not in the list L_S or R_{2i} is not generated by $\mathcal{B}$. Otherwise, $\mathcal{B}$ responds as follows.
  - If π_i is in the list L_S, $\mathcal{B}$ retrieves $< ID_i, T_i, Z_{Ti}, x_i >$ from $\mathcal{T}$ and $< T_i, \pi_i, w_i, r_i, z_i >$ from L_S. Let $H(r_{1i}P_S, 1) = v_i$, $\mathcal{B}$ sets $r_{1i} = r_i + b, r_{2i} = z_i$ and $v_i = \frac{x_i(w_i - ab)}{r_{2i}}$. Then, $\mathcal{B}$ computes

$$Auth_{Si} = r_i x_i aP + x_i w_i P + r_{2i} Z_{Ti}.$$

    $\mathcal{B}$ sets $m_i' = Auth_{Si}$ and outputs m_i'.
  - If $(Auth_{Ti}, R_{1i})$ is not in the list L_S, $\mathcal{B}$ rejects the request.

Eventually, $\mathcal{A}$ outputs an ID-verifier Z_T^*. If $Z_T^* \in \mathcal{T}$, $\mathcal{A}$ succeeds and $\mathcal{B}$ can utilize it to solve ECDH problem. Since $\mathcal{A}$ has to query hash oracle in the experiment, there is at least one query $(TK_{T1i}, 0)$ to the Hash query is correct. $\mathcal{B}$ retrieves TK_{T1i}

from the list L_H and computes $abP = TK_{T1i} - r_iP$, where $r_i \in L_S$, to be a solution of the given ECDH problem.

The simulation fails if $\mathcal{B}$ rejects a valid session in a SendReader query. It occurs when $(Auth_{Ti}, R_{1i}) \notin L_S$. It implies that $\mathcal{A}$ outputs $Auth_{Ti}$, such that

$$Z_{Ti} = Auth_{Ti} - TK_{T2i} - v_iR_{2i}$$

and $Z_{Ti} \in \mathcal{T}$. Since Z_{Ti} is a randomly generated and unknown to $\mathcal{B}$, $Auth_{Ti}$ is a random guess or $\mathcal{A}$ can solve ECDH problem. Let the event E be the simulation fails. We have the negligible probability $\Pr[E] \leq \epsilon' + \frac{k}{n}$, where k is the number of tags in $\mathcal{T}$.

□

Theorem 3.4. *Our repaired protocol provides the forward security if ECDH problem is hard.*

Proof. Suppose that there is an adversary $\mathcal{A}$ who can (ϵ, q_H, t)-win the experiment $\mathbf{Exp}^{forward}_{\mathcal{A},\mathcal{S}}[s,k]$. Let $\mathcal{A}$ has an advantage ϵ' to solve the ECDH problem.Given an instance (P, aP, bP), we can construct an algorithm $\mathcal{B}$ to find the solution abP of ECDH problem using the adversary $\mathcal{A}$. $\mathcal{B}$ interacts with $\mathcal{A}$ through the following oracles queries. Note that we give the simulation of oracles, the adversary shall use oracle calls in different phases of the experiment.

- Setup: $\mathcal{B}$ selects P as a generator of the additive cyclic group $\mathbb{G}$. Let the public key of the server be $P_S = aP$ and the private key of the reader be $x_S = a$, which is unknown to $\mathcal{B}$. $\mathcal{B}$ maintains the lists $L_H = \{< TK_{T2}, v, \{0,1\} >\}$, $\Sigma = \{< \pi, r_i >\}$ and a database of tags $\mathcal{T} = \{< ID, T, sk >\}$, which are initially empty.

- H Query: $\mathcal{A}$ issues H query on input (TK_{T2i}, c_i) at most q_H times. $\mathcal{B}$ outputs v_i if $< TK_{T2i}, v_i, c_i >$ is in the list L_H. Otherwise, $\mathcal{B}$ picks $v_i \overset{\$}{\leftarrow} \mathbb{Z}^*_n$ and sets $H(TK_{T2i}, c_i) = v_i$. Then, $\mathcal{B}$ outputs v_i and adds $< TK_{T2i}, v_i, c_i >$ into the list L_H.

- SetupTag Query: $\mathcal{A}$ issues the oracle query on input a tag's identity ID_i. $\mathcal{B}$ ignores the query if ID_i exists. Otherwise, $\mathcal{B}$ randomly chooses $x_{Ti} \overset{\$}{\leftarrow} \mathbb{Z}^*_n$ and computes $Z_{Ti} = x_{Ti}P$. $\mathcal{B}$ sets $sk_i = (x_{Ti}, Z_{Ti})$ as T_i's secret information. Then, $\mathcal{B}$ outputs the tag T_i and adds $< ID_i, T_i, sk_i >$ into the database $\mathcal{T}$.

- CorruptTag Query: $\mathcal{A}$ issues the oracle query on input a tag T_i. If T_i is not in $\mathcal{T}$, $\mathcal{B}$ creates a new tag by running SetupTag Query. $\mathcal{B}$ then outputs the T_i's secret information sk_i.

- Launch Query: $\mathcal{A}$ issues the oracle query on input a tag T_i, $\mathcal{B}$ outputs $\perp$ if T_i is corrupted, otherwise $\mathcal{B}$ randomly selects $r_i, r_{2i}, w_i, v_i \xleftarrow{\$} \mathbb{Z}_n^*$. Let $v_i' = \frac{w_i - ab}{r_{2i}}$, $\mathcal{B}$ computes

$$R_{1i} = r_i P + bP,\ R_{2i} = r_{2i} P,$$

$$Auth_{Ti} = Z_{Ti} + r_i aP + w_i P,$$

$$Auth_{Si} = x_{Ti} R_{1i} + v_i R_{2i} + r_{2i} Z_{Ti}.$$

  $\mathcal{B}$ sets $\pi_i = (T_i, R_{1i}, R_{2i}, Auth_{Ti}, Auth_{Si})$ and adds $< \cdot, v_i, 1 >$ into the list L_H, $< \pi_i, r_i >$ into the list Σ.

Eventually, $\mathcal{A}$ outputs a session π^* and a target tag T^*. If $\pi^* \in \Sigma$ and $T^* \in \pi^*$, $\mathcal{A}$ succeeds and $\mathcal{B}$ can utilize it to solve ECDH problem. Since $\mathcal{A}$ has to query hash oracle in the experiment, there is at least one query $(TK_{T1i}, 0)$ to the Hash query is correct. $\mathcal{B}$ retrieves TK_{T1i} from the list L_H and computes $abP = TK_{T1i} - r_i P$, where $r_i \in \Sigma$, to be a solution of the given ECDH problem.

The simulation fails if $\mathcal{A}$ queries a correct item $(TK_{S2i}, 0)$ to the hash oracle. It implies that $\mathcal{A}$ can either solve the ECDH problem or output a correct guess. Let the event E be the simulation fails. We have the negligible probability $\Pr[E] \leq \epsilon' + \frac{q_H}{n}$, where q_H is the number of hash oracle calls. □

3.5 Wide-Strong Privacy Model

We now review a privacy model defined in [HPVP11]. The oracles defined in the model are as follows.

- CreateTag(ID) $\rightarrow T_i$: Taking as input a tag's identifier ID, the oracle sets up and registers a new tag to server. Then, it outputs the reference T_i of the tag.

- Launch() $\rightarrow \pi, m$: It launches a new session π and returns the first message m sent by the reader.

- DrawTag$(T_i, T_j) \rightarrow vtag$: Taking as input a pair of tag references (T_i, T_j), it outputs $vtag$ which is a virtual tag reference linked to either T_i or T_j according to the value of g, where $g \in \{0, 1\}$. The oracle outputs $\perp$, if T_i or T_j is already drawn.

- Free$(vtag)$: Taking as input a virtual tag $vtag$, it retrieves the tuple $(vtag, T_i, T_j)$ and moves (T_i, T_j) to the set of free tags and resets T_i's (if $g = 0$) or T_j's (if $g = 1$) volatile memory.

- $\mathsf{SendTag}(vtag, m) \rightarrow m'$: Taking as input a virtual tag $vtag$ and a message m, the oracle retrieves $(vtag, T_i, T_j)$ and sends m to the tag T_i (if $g = 0$) or T_j (if $g = 1$). It outputs the tag's response m'.

- $\mathsf{SendReader}(\pi, m) \rightarrow m'$: Taking as input an instance π and a message m, the oracle sends m to the reader in session π and outputs the reader's response m'. If the session π is not activated, the oracle outputs $\perp$.

- $\mathsf{Result}(\pi) \rightarrow c$: Taking as input an instance π, the oracle outputs the result c of the authentication if π exists, otherwise outputs $\perp$.

- $\mathsf{Corrupt}(T_i) \rightarrow s$: Taking as input a reference T_i of the tag, the oracle outputs the state s of the tag if T_i is not drawn, otherwise outputs $\perp$.

The model defined eight different classes of privacy and adversary. In each class, the adversary is restricted by the capability of oracle access. The strongest adversary in the model is the wide-strong adversary who can access the all above oracles as many times as he needs in polynomial time. The privacy experiment $\mathbf{Exp}_{\mathcal{A},\mathcal{S}}^{ws-private}$ for the wide-strong adversary is as follows:

1. Setup: The system $\mathcal{S}$ sets up the system depending on the security parameter k and chooses a random bit $g \in \{0, 1\}$.

2. Learning: The adversary $\mathcal{A}$ can interact with $\mathcal{S}$ in polynomial time and queries all above oracles.

3. Guess: The adversary outputs a bit g'. If $g' = g$, the experiment outputs 1, 0 otherwise.

We say that the adversary $\mathcal{A}$ wins the wide-strong privacy game if and only if the experiment outputs 1.

Definition 3.3. *A RFID authentication protocol is privacy-preserving if there is no adversary $\mathcal{A}$ who wins the wide-strong privacy game in polynomial time t with the advantage $Adv_{\mathcal{A}}$ at least ϵ, where*

$$Adv_{\mathcal{A}} = \left| \Pr[\mathbf{Exp}_{\mathcal{A},\mathcal{S}}^{ws-privacy} = 1] - \frac{1}{2} \right| \geq \epsilon.$$

3.6 Proposed Protocols

Many ECC-based RFID identification protocols employ Diffie-Hellman keys to preserve the privacy of the tag. Usually, there are two approaches to generate the

Diffie-Hellman key: 1) The tag uses its private key and the nonce(s) to compute with the reader's public key (e.g., [BSSV11, LBV08, LBV09]); 2) The tag chooses a random number to compute with the reader's public key (e.g., [PH12]). However, a strong adversary can compromise the tag and obtain the tag's private key. Hence, the two ways provide the equal level of privacy protection under the strong attack. In this chapter, we adopt the second approach.

Clearly, a tag's response should not be transferable to another valid response even if the tag's private key is known to the adversary. In our protocols, we protect the tag's private key by using two random values. Given a valid tag's response, anyone who does not have the tag's temporary key or the reader's private key cannot output a new valid tag's response.

3.6.1 Protocol 1

Our protocol is a variant of the Schnorr identification protocol [Sch89]. The identification process consists of two passes where the reader initiates the session. Prior to identifying the tag, both of the reader and the tag are required to store particular states. Let $\mathbb{G}$ is an additive group with the prime order p and P is a generator of the group. The public/private key pairs of the tag and the reader are $(X = xP, x)$ and $(Y = yP, y)$, respectively, where $x, y \xleftarrow{\$} \mathbb{Z}_p^*$. Initially, the back-end server inserts the tag's public key X into the database DB as the tag's identifier. The server sets the tuple (x, Y, P) as the tag's state and stores it into the tag. The reader receives its pair of public/private keys and it is allowed to access the database.

To identify a tag, the reader randomly chooses $C \in \mathbb{G}$ and sends C as a challenge to the tag. Upon receiving the challenge, the tag firstly picks a random number $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $R = rP$. Let $H : \mathbb{G} \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_p^*$ be a cryptographic hash function. The tag generates a signing message

$$v = H(R, rY, C),$$

where rY is a temporary Diffie-Hellman key. The signing message is computable if and only if either the tag's choice r or the reader's private key y is known. It is significant to preserve the tag's privacy. Then the tag computes

$$s = xv + r \pmod{p},$$

and sends (R, s) to the reader. On receiving the tag's response, the reader extracts the tag's identity as

$$v' = H(R, yR, C),\ X' = (sP - R){v'}^{-1}.$$

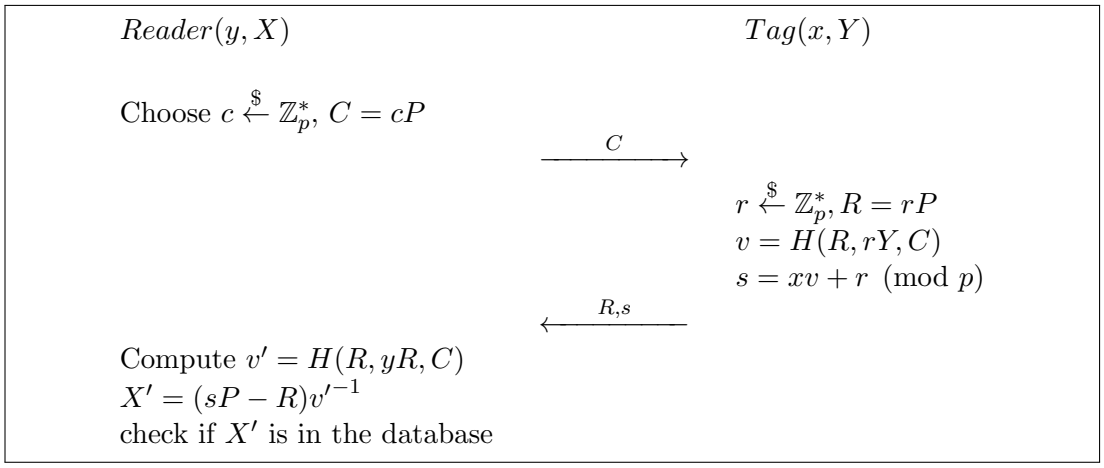

Figure 3.7: Basic protocol.

If X' exists in the database, the tag is identified, otherwise it is rejected. The proposed basic RFID identification protocol is depicted as in Figure 3.7.

The reader can extract the tag's signature after a successful tag authentication. Given yR and C, anyone who has the tag's public key X can verify the validity of the signature (R, s). It is an important difference between the encryption based protocols and the zero-knowledge based protocols.

3.6.2 Protocol 2

RFID tags are resource-constrained devices which have limited gates to implement protocols. The increase of the tag's gates costs more in production. In terms of the hardware implementation of our basic protocol, the tag is required to do the modular in both of the prime field and the binary field. Although the modular is an efficient operation, it consumes large number of gates for the hardware implementation [OF09, Sha08]. Unfortunately, most of RFID identification protocols which are based on public key cryptography need modular calculations in both of the prime field and the binary field.

In this section, we propose an optimized protocol and show that the number of required gates are reduced. As a feature, there is no modular operation in the prime filed required to the tag. Instead, only modular calculations in the binary field is needed. The optimized protocol also consists of two passes where the reader initiates the session. Let $\mathbb{G}$ be an additive group with the prime order q and $\hat{e}$ be a bilinear pairing, where $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. P_1 and P_2 are two generators of the group $\mathbb{G}$. The public/private key pairs of the tag and the reader are $(xP_2, X = \hat{e}(P_1, xP_2))$ and $(y, Y = yP_2)$, respectively, where $x, y \xleftarrow{\$} \mathbb{Z}_p^*$. The back-end server inserts the entry of the tag into the databse and stores the tuple (xP_2, Y, P_1, P_2) into the tag.

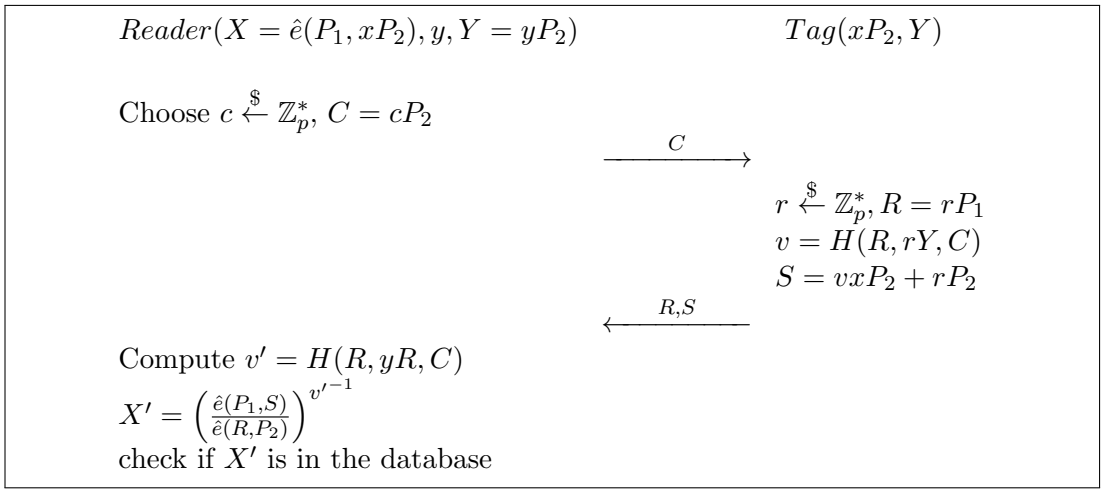

Figure 3.8: Optimized protocol.

The reader receives its pair of public/private keys and it is allowed to access the database.

To identify a tag, the reader randomly selects $C \xleftarrow{\$} \mathbb{G}$ and sends C as a challenge to the tag. Upon receiving the challenge, the tag chooses a random number $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $R = rP_1$. Then, the tag generate a signing message v as in the basic protocol, where $v = H(R, rY, C)$. The tag computes

$$S = vxP_2 + rP_2,$$

and sends (R, S) to the reader. On receiving the tag's response, the reader extracts the tag's identity as

$$v' = H(R, yR, C),\ X' = \left(\frac{\hat{e}(P_1, S)}{\hat{e}(R, P_2)}\right)^{v'^{-1}}.$$

If X' exists in the database, the tag is identified, otherwise it is rejected. The optimized RFID identification protocol is depicted as in Figure 3.8.

3.7 Privacy Analysis

We analyse the privacy of proposed protocols and show that they are wide-strong private in model [HPVP11].

Theorem 3.5. *The proposed basic RFID authentication protocol is private against wide-strong attack if the ECDH problem is hard.*

Proof. Suppose that there is an adversary $\mathcal{A}$ who can (ϵ, q_h, t)-distinguish the 'left' and 'right' world in the wide-strong privacy experiment. Let $\mathcal{A}$ has an advantage ϵ' to

solve the ECDH problem. We can construct an algorithm $\mathcal{B}$ run by the challenger to solve the ECDH problem using the adversary $\mathcal{A}$. Given the ECDH instance (P, aP, bP), algorithm $\mathcal{B}$ aims to output abP. On behalf of the system $\mathcal{S}$, $\mathcal{B}$ interacts with the adversary $\mathcal{A}$ as follows.

- Setup: $\mathcal{B}$ sets P as the generator of the additive cyclic group $\mathbb{G}$. Let the public key of the reader be $Y = aP$ and the private key of the reader be $y = a$, which is unknown to $\mathcal{B}$. $\mathcal{B}$ maintains the lists $L_h = \{< R, rY, C, v >\}$, $L_{Ref} = \{< vtag, T_i, T_j >\}$, $L_S = \{< T, \pi, z >\}$ and a database of tags $\mathcal{T} = \{< ID, T, X, x >\}$, which are initially empty. $\mathcal{B}$ tosses a coin and sets $g = 0$ or $g = 1$, where $\Pr[g = 0] = \Pr[g = 1] = \frac{1}{2}$. The virtual tag reference $vtag$ is an incremental counter starts from 0.

- H Query: $\mathcal{A}$ issues $h_{\mathcal{E}}$ query on input (R_i, r_iY, C_i) at most q_h times. $\mathcal{B}$ outputs v_i if (R_i, r_iY, C_i) is in the list L_h. Otherwise, $\mathcal{B}$ randomly selects $v_i \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $H(R_i, r_iY, C_i) = v_i$. Then, $\mathcal{B}$ outputs v_i and adds $< R_i, r_iY, C_i, v_i >$ into the list L_h.

- CreateTag Query: $\mathcal{A}$ issues the oracle query on input a tag identity ID_i. If ID_i is not in $\mathcal{T}$, $\mathcal{B}$ sets up a new tag T_i and generates the tag's pubic/private key pair (x_i, X_i), where $x_i \xleftarrow{\$} \mathbb{Z}_p^*$, $X_i = x_iP$. $\mathcal{B}$ outputs the reference T_i and adds $< ID_i, T_i, X_i, x_i >$ into the database $\mathcal{T}$. If ID_i exists, $\mathcal{B}$ ignores the query.

- DrawTag Query: $\mathcal{A}$ issues the oracle query on input a pair of tag references (T_i, T_j). If any of the issued tags is not free, which is currently referenced, the oracle outputs $\perp$. If $g = 0$, $\mathcal{B}$ references $vtag$ to T_i, T_j otherwise. $\mathcal{B}$ outputs $vtag$ and adds $< vtag, T_i, T_j >$ into the list L_{Ref}.

- Free Query: $\mathcal{A}$ issues the oracle query on input a reference $vtag$. If $vtag$ is in the list L_{Ref}, $\mathcal{B}$ deletes the entry $< vtag, T_i, T_j >$ and erases the volatile memory of the referenced tag, which is T_i or T_j.

- Corrupt Query: $\mathcal{A}$ issues the oracle query on input a tag reference T_i. If T_i is not in $\mathcal{T}$, $\mathcal{B}$ firstly creates a new tag by using CreateTag Query. $\mathcal{B}$ then outputs the tag's secret key x_i.

- SendTag Query: $\mathcal{A}$ issues the oracle query on input $vtag$ and a message C_i. If the entry $< vtag, T_i, T_j >$ is not in the list L_{Ref}, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ retrieves the the referenced tag T_g's secret key x_g and computes as follows.

  – Randomly selects $z_i \xleftarrow{\$} \mathbb{Z}_p^*$ and let $r_i = b + z_i$. Then, $\mathcal{B}$ computes $R_i = bP + z_iP$.

  – $\mathcal{B}$ randomly picks $w_i \xleftarrow{\$} \mathbb{Z}_p^*$ and lets $v_i = w_i - \frac{b}{x_g}$.

– Computes $s_i = x_g w_i + z_i$ and sets $m_i = (R_i, s_i)$, $\pi_i = (C_i, m_i)$.

$\mathcal{B}$ outputs m_i and adds $< T_i, \pi_i, z_i >$ into the list L_S. We show that the simulation is perfect as

$$\begin{aligned} s_i &= x_g w_i + z_i \\ &= x_g(w_i - \frac{b}{x_g}) + (b + z_i) \\ &= x_g v_i + r_i. \end{aligned}$$

- SendReader Query: Since there is no reply message from the reader, $\mathcal{B}$ ignores the query to this oracle.
- Result Query: $\mathcal{A}$ issues the oracle query on input a session π_i. $\mathcal{B}$ responses as follows.
  - If π_i is in the list L_S, $\mathcal{B}$ accepts the session and outputs 1.
  - If π_i,is not in the list L_S, $\mathcal{B}$ looks up the list L_h. If $< R_i, \cdot, C_i, v_i >$ is not in L_h, $\mathcal{B}$ outputs 0 and rejects the session.
  - $\mathcal{B}$ Computes $X_i = (s_i P - R_i)v_i^{-1}$ and verifies it by checking if X_i in the the database $\mathcal{T}$. $\mathcal{B}$ outputs 1 if it exists, 0 otherwise.

Eventually, the adversary has to output a bit $g' \in \{0, 1\}$ in the guess phase. That is, to determine which world ('left' or 'right') the simulation has encountered. If the adversary successfully outputs $g' = g$, he wins the experiment and $\mathcal{B}$ can use it to solve the ECDH problem. Since $\mathcal{A}$ has to query the hash oracle to determine which tag is referenced during the experiment, there is at least one query input $(R_i, r_i Y, c_i)$ to the Hash Query is correct. $\mathcal{B}$ retrieves $r_i Y$ from the list L_h and computes $abP = r_i Y - z_i Y$, where $z_i \in L_S$, to be a solution of the given ECDH problem.

The simulation fails when $\mathcal{B}$ rejects a valid session. It occurs when $\mathcal{A}$ issued a valid session π to Result while $< R_i, \cdot, C_i, v_i >$ is not in the list L_h. A valid session which is not generated by $\mathcal{B}$ implies that the adversary could find the Diffie-Hellman key $r_i Y$ or guess the correct s_i. Let the event E be that the simulation fails. We have the negligible probability $\Pr[E] \leq \epsilon + \frac{n}{q}$, where n is the number of tags in $\mathcal{T}$. □

Theorem 3.6. *The proposed optimized RFID identification protocol is private against the wide-strong adversary if the ECDH problem is hard.*

Proof. Suppose that there is an adversary $\mathcal{A}$ who can (ϵ, q_h, t)-distinguish the 'left' and 'right' world in the wide-strong privacy experiment. Let $\mathcal{A}$ has an advantage

ϵ' to solve the ECDH problem. Given an instance (P, aP, bP), we can construct an algorithm $\mathcal{B}$ to find the solution abP of ECDH problem using the adversary $\mathcal{A}$. $\mathcal{B}$ interacts with the adversary $\mathcal{A}$ as follows.

- Setup: $\mathcal{B}$ selects k, where $k \xleftarrow{\$} \mathbb{Z}_p^*$ and sets P_1, P_2, where $P_1 = kP, P_2 = P$, as two generators of the additive cyclic group $\mathbb{G}$. Let the public key of the reader be $Y = aP$ and the private key of the reader be $y = a$, which is unknown to $\mathcal{B}$. $\mathcal{B}$ maintains the lists $L_h = \{< R, rY, C, v >\}$, $L_{Ref} = \{< vtag, T_i, T_j >\}$, $L_S = \{< T, \pi, z >\}$ and a database of tags $\mathcal{T} = \{< ID, T, X, xP >\}$, which are initially empty. $\mathcal{B}$ tosses a coin and sets $g = 0$ or $g = 1$, where $\Pr[g = 0] = \Pr[g = 1] = \frac{1}{2}$. The virtual tag reference $vtag$ is an incremental counter starts from 0.

- H Query: $\mathcal{A}$ issues $h_{\mathcal{E}}$ query on input (R_i, r_iY, C_i) at most q_h times. $\mathcal{B}$ outputs v_i if (R_i, r_iY, C_i) is in the list L_h. Otherwise, $\mathcal{B}$ picks $v_i \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $H(R_i, r_iY, C_i) = v_i$. Then, $\mathcal{B}$ outputs v_i and adds $< R_i, r_iY, C_i, v_i >$ into the list L_h.

- CreateTag Query: $\mathcal{A}$ issues the oracle query on input a tag's identity ID_i. $\mathcal{B}$ ignores the query if ID_i exists. Otherwise, $\mathcal{B}$ randomly chooses $x_i \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $X_i = \hat{e}(kP, x_iP)$. Then, $\mathcal{B}$ creates a new tag and sets (X_i, x_iP) as its public and private key pair. $\mathcal{B}$ outputs the reference T_i and adds $< ID_i, T_i, X_i, x_iP >$ into the database $\mathcal{T}$.

- DrawTag Query: $\mathcal{A}$ issues the oracle query on input a pair of tag references (T_i, T_j). If any of the issued tags is not free, the oracle outputs $\perp$. Depending on the value of g, $\mathcal{B}$ references $vtag$ to T_i (if $g = 0$) or T_j (if $g = 1$). $\mathcal{B}$ outputs $vtag$ and adds $< vtag, T_i, T_j >$ into the list L_{Ref}.

- Free Query: $\mathcal{A}$ issues the oracle query on input a reference $vtag$. If $vtag$ is in the list L_{Ref}, $\mathcal{B}$ removes the entry $< vtag, T_i, T_j >$ and erases the volatile memory of the referenced tag.

- Corrupt Query: $\mathcal{A}$ issues the oracle query on input a tag reference T_i. If T_i is not in $\mathcal{T}$, $\mathcal{B}$ creates a new tag by running CreateTag Query. $\mathcal{B}$ then outputs the tag's secret key x_iP.

- SendTag Query: $\mathcal{A}$ issues the oracle query on input $vtag$ and a message C_i. $\mathcal{B}$ outputs $\perp$ If $< vtag, T_i, T_j >$ is not in the list L_{Ref}. Otherwise, $\mathcal{B}$ retrieves the the referenced tag T_g's secret key x_gP and randomly selects $z_i, w_i \xleftarrow{\$} \mathbb{Z}_p^*$. Then, $\mathcal{B}$ computes

$$R_i = kbP + z_ikP,\ S_i = w_ix_gP + z_iP,$$

and sets $m_i = (R_i, S_i)$, $\pi_i = (C_i, m_i)$. $\mathcal{B}$ outputs m_i and adds $< T_i, \pi_i, z_i >$ into the list L_S.

- SendReader Query: Since there is no reply message from the reader, $\mathcal{B}$ ignores the query to this oracle.
- Result Query: $\mathcal{A}$ issues the oracle query on input a session π_i. $\mathcal{B}$ outputs 1 if π_i is in the list L_S, otherwise $\mathcal{B}$ outputs 0 if $< R_i, \cdot, C_i, v_i >$ is not in the list L_h. If $< R_i, \cdot, C_i, v_i >$ exists, $\mathcal{B}$ computes $X_i = (\frac{\hat{e}(P_1,S_i)}{\hat{e}(R_i,P_2)})^{v_i^{-1}}$ and outputs 1 if X_i appears in $\mathcal{T}$, 0 otherwise.

Eventually, if the adversary outputs a guess g', where $g' = g$, $\mathcal{B}$ has at least one correct value of r_iY in the list L_h. $\mathcal{B}$ can find the solution of ECDH problem as $abP = r_iY - z_iY$, where $z_i \in L_S$. The simulation fails when $\mathcal{B}$ outputs a false rejection with the negligible probability at most $\epsilon + \frac{n}{q}$, where n is the number of tags in $\mathcal{T}$. □

3.8 Conclusion

In this chapter, we demonstrated attacks against Liao and Hsiao's protocol and offered different solutions to the issues. Then, a concrete repaired protocol was presented and it was proven secure. Based on Liao and Hsiao's security requirements, Section 3.2.2 gave an example of translating formal privacy models. After that, a wide-strong privacy model is given and we proposed two novel zero-knowledge based RFID authentication protocols. As a feature, a reader can obtain a tag's signature after a successful tag authentication. In addition, the optimized protocol eliminates modular computations over the prime field. Finally, two proposed protocols were proven secure against wide-strong adversaries.

Chapter 4

Secure RFID Ownership Transfer

An RFID tag could change hands many times during its lifetime. In a retail chain, the ownership of the tag is instituted by the supplier who initially owns the tag. In the view of a buyer, the validity of the current tag ownership and the originality of supplier are most important. In typical RFID ownership transfer protocols, the knowledge of the tag's autIn our scheme,hentication key proves the ownership. However, it is insufficient against an active attacker, since tags are usually lack of tamper-proof protections. Ownership transfer relies on a successful verification of tag's supplier and current ownership. In this chapter, we formally define the security model of ownership transfer protocols and propose a secure ownership transfer protocol, which was originally proposed in [LMSV13]. In our scheme, the current owner provides a new owner with the evidence of transfer and a proof of tag origin. Key management becomes easy in our system, since one asymmetric verification key of the owner can be used to verify multiple tags that belong to the owner.

4.1 Introduction

RFID has exhibited many practical applications such as serving as identity of an object in supply chains, supermarkets and hospitals. A tag attached to a product has a unique identifier stored in a back-end database. In practice, a product (with a tag) is owned by a user. Often, the product needs to change hands due to selling or buying. This process is referred to as ownership transfer.

In the lifetime of a tag, its ownership is likely to be transferred from one owner to another. An ownership transfer protocol runs between the current owner and the new owner. Generally speaking, the protocol is considered in two phases, namely ownership verification and ownership transfer. A new owner firstly verifies the current ownership of the tag. If the current ownership is confirmed, he can request the ownership transfer. After a successful ownership transfer, the current owner who becomes the previous owner of the tag can no longer access the tag and the new owner who becomes the current owner of the tag can prove the ownership of the tag. According to the current ownership of the tag, a user can be a previous owner, current owner or new owner of the tag.

The security of RFID ownership transfer protocols is considered in three aspects: the secure ownership, exclusive ownership and secure ownership transfer [vDMRV09]. The first two properties are related to the phase of the ownership

verification. Informally, they guarantee that the actual owner always has the ownership of a tag and no others can simultaneously obtain the ownership. The criteria of secure ownership transfer evaluates the phase of the tag ownership transfer. A new user who is unauthorized by the current owner cannot gain the ownership of the tag. A secure ownership transfer protocol should satisfy all these requirements.

The traditional RFID ownership transfer protocols are based on the lightweight symmetric-key authentication schemes. The back-end server and a tag share a predefined symmetric-key and the tag's identity. The tag's ownership is checked by implementing the authentication protocol. However, most (passive) tags are not tamper-resistant, so that adversaries can launch active attacks. It is possible to physically corrupt or clone a tag and obtain the internal state. Once the internal state is leaked, the adversary can control the tag as the real owner. Therefore, it can prove the ownership and even transfer the tag to others.

Motivation. The aim of this chapter is to propose a secure RFID ownership transfer protocol. In most previous RFID ownership transfer protocols, the proof of ownership relies on the knowledge of the tag's authentication key. If the user can provide a valid secret key, the verifier accepts its ownership of the tag. However, it is insufficient against the attacker who compromises the tag. In practice, we call the party who currently owns the tag a seller and the party who receives the ownership a buyer. The symmetric authentication key shared between the seller and the tag provides no identity information about the seller. Anyone who has the key is able to prove the ownership and transfer it to other parties. It may injure the rights of the seller and the buyer. As a buyer, it usually concerns the origin of the product and the validity of the seller. He expects to check them during a purchase. The key management in large RFID system is also an issue. A tag normally requires a unique key for proving the ownership. The buyer has to obtain a large number of keys to check ownerships of tags. It not only requires a secure channel in communication, but it is hard to maintain the records of these transactions. It would be desirable that one verification key can do the job. With this key, anyone can verify the ownership of tags that belong to the owner.

We look into an RFID system, where a supplier obtains products from a manufacturer. The supplier authorizes the manufacturer, via a warrant, to make specific products. After the products are ready for the supplier, the manufacturer setups RFID tags and attaches them to products, respectively. When a buyer purchases the product from the supplier, the ownership transfer is required. The buyer checks the information of the supplier and the product prior to making a payment. Once the deal is complete, the buyer owns the tag and the supplier can no longer claim ownership. Meanwhile, the seller provides the undeniable transfer proof which includes the information of seller, buyer and tag. The buyer can also resell the product

in the future. One aim in this chapter is to construct an ownership transfer scheme in this scenario.

Symmetric-key based protocols are insufficient to reach a strong security level for ownership transfer protocols. It is a challenge to resist an active attack. We assume that the tag authentication can be done using a traditional RFID authentication protocol, while we only focus on the ownership transfer protocol. An owner is usually a powerful entity which can perform public key cryptographic algorithms for ownership transfer, which does not rely on the computation power of the tag.

Related Work. Saito, Imamoto and Sakurai [SIS05] introduced an ownership transfer protocol using two approaches. Both provide the privacy and security protection of the current owner and the new owner. One is based on the three-party model and the other is on two-party model. Since the schemes are based on symmetric-key cryptographic algorithms, the secret key of tag is pre-shared with the owner. In the three-party model, the second key is shared between the TTP and the tag. In ownership transfer, the TTP helps the new owner to update the tag's new secret. While the online TTP is required during the ownership transfer. Once the tag is compromised, the shared secret key between the tag and the TTP is also disclosed.

Independently, Molnar, Soppera and Wagner [MSW05] proposed an ownership transfer protocol of RFID tags. The protocol addresses the privacy problems of ownership transfer through the *pseudonym*. The proposed scheme employs a tree based key structure to enable the *time-limited delegation* for temporarily ownership transfer. It is that the current owner can temporarily delegate the ownership of the tag to another party. After a period of time, the ownership is returned to the original owner without the agreement of the delegatee. However, the scheme needs a counter which is in the non-volatile memory to count the number of authentications. A *Trusted Center* (TC) who controls all the secret of the tags assists the readers to authenticate the tag. Unfortunately, most TTP-based ownership transfer protocols [FA07, KYWG10, OTYT06, MSW05] suffer from the similar issues as in [SIS05].

Several security properties of ownership transfer protocols were introduced by Ng, Susilo, Mu and Safavi-Naini [NSMSN11], where they introduced four new properties: tag assurance, current ownership proof, undeniable ownership transfer and owner initiation. The proposed scheme satisfies most security properties of ownership transfer while only some hash calculations are required on the tag. Elkhiyaoui, Blass and Molva [EBM11] presented the problem of issuer verification during the ownership transfer. In this chapter, the privacy and security of ownership transfer protocols are formally defined and the proposed scheme achieves the constant time authentication. The scheme prevents the attacker from injecting fake tags in the supply chains. The origin of the tag is verified prior to the transfer. Abyaneh

[Aby12] shows that forward and backward privacy are broken if the attacker was an owner of the tag. Additionally, the definition of the security model does not allow the adversary to rewrite the tag's content. It may be vulnerable against some active attacks.

A scalable authentication protocol which supports the ownership transfer was proposed in [FMTRCRDF11b]. The protocol provides the controlled delegation without using the non-volatile memory to store a counter. The feature of desynchronization engages the protocol runs without the TTP. It employs a table which consists of two hash chains to identify a tag. While, the cost of storage on the server is questionable when the maximum size of the hash chains increased. Meanwhile, it also suffers form the denial-of-service attack.

Deursen, Mauw, Radomirović and Vullers [vDMRV09] introduced a formal definition of secure ownership transfer in RFID systems. They described two roles: the *tag owner* and the *tag holder*. Basically, both of them can pass the ownership test but only the owner is engaged to transfer the ownership. It was claimed that the tag owner and holder are coincide in the notion of secure ownership. However, the holder of the tag may not be the owner in decentralized systems. Since the security of ownership is based on the authentication of the tag, most symmetric-key ownership transfer protocols [Son08, SIS05, MSW05, RRG09] assume that the tag is incorruptible. In [NSMSN11] and [EBM11], a tag is allowed to be compromised. Nevertheless, the content of the tag cannot be rewritten after the adversary disclosed the key.

Organization of This Chapter. The rest of this chapter is organized as follows. In Section 4.2, we introduce a new system model of RFID tag ownership transfer. The proposed secure ownership transfer protocol is given in Section 4.3. We define the adversary and security models of ownership transfer protocols in Section 4.4. Section 4.5 presents the formal security proof of the proposed scheme. Finally, Section 4.6 concludes the chapter.

4.2 System Model

In this section, we formally define ownership transfer protocols using the retail chain as an instance.

4.2.1 Entities

- **Tag** T_i: An object is attached by one tag T. The tag has a small memory which stores the current state s_i of the tag. T_i is a low-cost device which can at most calculate the hash function F.

- **Manufacturer** M_i: The manufacturer is the one who makes the products for suppliers. One manufacturer can cooperate with many different suppliers while the product must be authorized by the specific supplier.

- **Supplier** S_i: The supplier is the one who sells the products to customers. It handles the first ownership transfer of the tag. The supplier authorizes the manufacturer to produce expected number of products meanwhile S provides a unique warrant for each product.

- **Previous Onwer** $O_{(t_i,k-1)}$: The previous owner $O_{(t_i,k-1)}$ is the one who previously owns the tag T_i at the time $k-1$. It provides the proof of transfer $\Sigma_{(t,k-1,k)}$ to the current owner.

- **Current Owner** $O_{(t_i,k)}$: The current owner $O_{(t_i,k)}$ is the one who currently owns the tag T_i at the time k. It maintains a database which stores the states of tags and authenticates tags though a reader R_k. The current owner can prove the current ownership $\sigma_{(t_i,k)}$ of the tag and show the valid transfer obtained from the previous owner. $O_{(t_i,k)}$ is allowed to transfer the current ownership of T_i to the new owner.

- **New Owner** $O_{(t_i,k+1)}$: The new owner $O_{(t_i,k+1)}$ is the one who is a potential owner of the tag T_i. Prior to accepting the ownership of tag T_i, the new owner verifies the tag's supplier S, the previous transfer proof $\Sigma_{(t,k-1,k)}$ and the current ownership $\sigma_{(t_i,k)}$. It provides an evidence of the acceptance once the transfer is completed.

The supplier can be considered as a special owner of tag and the manufacturer is an agent of particular supplier. The previous owner, current owner and new owner are roles which are changeable in different periods of the tag ownership. That means the new owner becomes a current owner or previous owner once he receives or transfers the tag ownership, respectively.

4.2.2 RFID Ownership Transfer Systems

In our system model, we do not employ the centralized server which is normally a TTP. Instead, we adopt the two-party mode where each party maintains an isolated database and readers. A party who engages in the ownership transfer is an owner of a tag. From now on, we refer to an owner as an entity which is supported by RFID readers and a back-end database. While one owner has a public/private key pair where the public key is known to anyone. In the model, we only need the secure communication channel during the authentication key exchange. Since the proposed scheme applies symmetric-key based tag authentication, it is impossible to securely

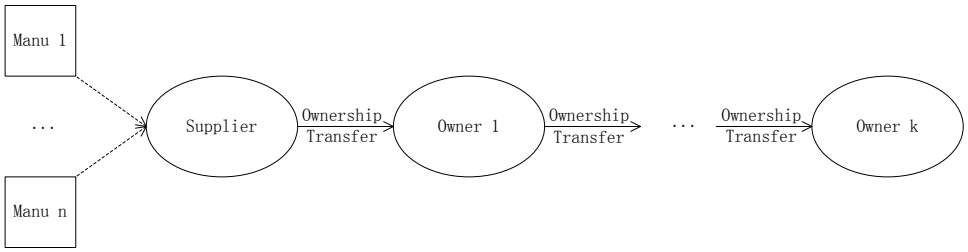

Figure 4.1: Ownership transfer systems

update the key with shared secret [KZP11]. The key update of the protocol should be performed outside the control range of the previous owner.

The ownership transfer system is described in Figure 4.1. Different from the previous models, we consider the ownership of the tag as a chain. To handle an ownership transfer, the information of tag's supplier, previous owner, current owner and new owner are all required. Nevertheless, only the current owner needs to provide its secret.

In the model, the ownership transfer stems from the supplier. Let one owner be a level. Level 0 is the supplier of the tag. The manufacturer generates the proof of ownership under the supplier's warrant and stores it on the tag. Anyone who has the supplier's public key can verify the ownership of the product. In this level, the supplier simultaneously plays the role of the previous owner since the product is brand new. Then, it transfers the ownership to a new owner who is in level 1. Owner 1 accepts the ownership from the supplier and takes the role of the current owner. At this time, the supplier transferred the current ownership but remains the role of supplier and previous owner of the tag. Following the process, the ownership of the tag is generally in the k-th level.

A complete ownership transfer process has two phases: ownership verification and ownership transfer. In the ownership verification phase, the buyer checks the supplier of the tag, previous authenticated transfer proof and the validity of current ownership. Only if all the verifications are successful, two owners play the game of the ownership transfer. In the completion of an ownership transfer, the seller outputs a new authenticated transfer proof and the buyer outputs a proof of new ownership.

4.2.3 Ownership Transfer Protocols

An RFID ownership transfer protocol consists of seven algorithms: system setup (Setup), key generation (KeyGen), tag initiation (TagInit), authentication (Auth), ownership transfer (Transfer), ownership prove (OwnerProve) and ownership verification (OwnerVerify). The seven algorithms in RFID ownership transfer protocols are defined as follows.

- $params \leftarrow \mathsf{Setup}(\lambda)$: Taking as input a security parameter λ, outputs a set of public parameters $params$.
- $(pk, sk) \leftarrow \mathsf{KeyGen}(params)$: Taking as input the system parameters $params$, outputs a pair of public and private keys (pk, sk).
- $(c, \sigma_{(t,0)}) \leftarrow \mathsf{TagInit}(T, pk_s, sk_s, pk_m, sk_m)$: Taking as input a tag T, a pair (pk_s, sk_s) of supplier's public/private keys and a pair (pk_m, sk_m) of manufacturer's public/private keys, outputs the tag's initial state c and ownership proof $\sigma_{(t,0)}$. It runs between a manufacturer and a supplier.
- $Info \leftarrow \mathsf{Auth}(T, O_{(t,k)})$: Taking as input a tag T and the current owner $O_{(t,k)}$, outputs a set of information $Info$ of tag. It runs between the current owner and the tag.
- $\Sigma_{(t,k,k+1)} \leftarrow \mathsf{Transfer}(ID_t, pk_s, pk_{k-1}, pk_k, sk_{k+1}, \Sigma_{(t,k-1,k)})$: Taking as input a tag's identity ID_t, the public key pk_s of supplier, a pair of public/private key (pk_k, sk_k) of current owner and a new owner's public key pk_{k+1}, outputs an authenticated transfer proof $\Sigma_{(t,k,k+1)}$. It is run by the current owner.
- $\sigma_{(t,k)} \leftarrow \mathsf{OwnerProve}(ID_t, sk_k, \Sigma_{(t,k-1,k)}, \sigma_{(t,k-1)})$: Taking as input a tag's identity ID_t, a private key sk_k of current owner and an authenticated transfer proof $\Sigma_{(t,k-1,k)}$, outputs a proof $\sigma_{(t,k)}$ of ownership. It is run by the current owner.
- $\{true, false\} \leftarrow \mathsf{OwnerVerify}(ID_t, pk_s, pk_{k-1}, pk_k, \sigma_{(t,k)})$: Taking as input a tag's identity ID_t, the supplier's verification key pk_s, the previous owner's verification key pk_{k-1} and the current owner's verification key pk_k and a proof $\sigma_{(t,k)}$ of ownership, outputs $true$ if the proof is valid, outputs $false$ otherwise.

Without loss of generality, we describe the Auth algorithm in the protocol. Also, it is unnecessary to the security of ownership transfer protocols. In this chapter, Auth is assumed to be a privacy-preserving authentication protocol. The interaction of one ownership transfer is depicted as in Figure 4.2.

4.3 Proposed Protocol

The mathematical preliminaries and concrete construction of the proposed scheme are presented in the section.

4.3.1 Construction

- Setup: Select a symmetric bilinear pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where the order of group $\mathbb{G}$ and $\mathbb{G}_T$ are the same prime number p. Let $P, U \in \mathbb{G}$ be two generators.

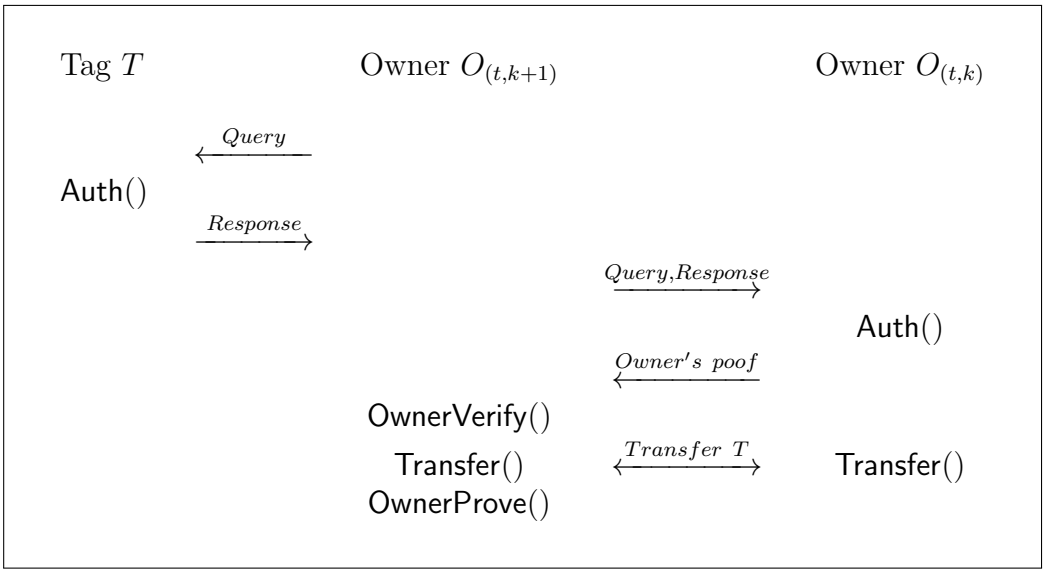

Figure 4.2: Ownership transfer protocol.

$H_1 : \mathbb{G} \times \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, $H_2 : \mathbb{G} \times \mathbb{G} \times \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $F : \{0,1\}^* \rightarrow \{0,1\}^l$ are collision-resistant cryptographic hash functions. Sets the public parameters $params = (\mathbb{G}, \mathbb{G}_T, P, U, p, \hat{e}, H_1, H_2, F)$.

- KeyGen: Randomly chooses $x \in \mathbb{Z}_p^*$ and sets the public/private key pair as $(pk, sk) = (xP, x)$.

- TagInit: Let the public/private key pairs of a manufacturer M and a supplier respectively be $(pk_m, sk_m) = (xP, x)$ and $(pk_s, sk_s) = (\alpha P, \alpha)$. Firstly, the manufacturer and the supplier interacts as in Figure 4.3. The manufacturer generates an ownership proof σ_0 for the supplier. It randomly chooses an authentication key y from the key space $\mathcal{S}$ and sets the tag state $c = (y, F(\sigma_{(t,0)})$. The supplier is the owner $O_{(t,0)}$.

- Auth: It is a general symmetric-key based authentication protocol. The current owner $O_{(t,k)}$ interacts the tag T using a pre-shared symmetric authentication key y. Once the authentication protocol outputs 1, the owner collects the tag's information $info$ which includes the tag's identity ID_t, ownership proof $\sigma_{(t,k)}$, etc.

- Transfer: To transfer the ownership, the current owner $O_{(t,k)}$ interacts with the new owner $O_{(t,k+1)}$. If the current owner is a supplier, it follows the description as in Figure 4.4. Otherwise, it follows the description as in Figure 4.5. Assume that the identity of tags and public information of two owners are mutually known.

- OwnerProve: To generate a proof of ownership, the current owner $O_{(t,k)}$ retrieves the proof $\Sigma_{(t,k-1,k)} = (s_{(t,k-1,k)}, R_{(t,k-1,k)})$ of authenticated transfer and the ownership poof $\sigma_{(t,k-1)}$ of owner $O_{(t,k-1)}$. Computes $s_{(t,k)} = s_{(t,k-1,k)} + sk_k$, where sk_k

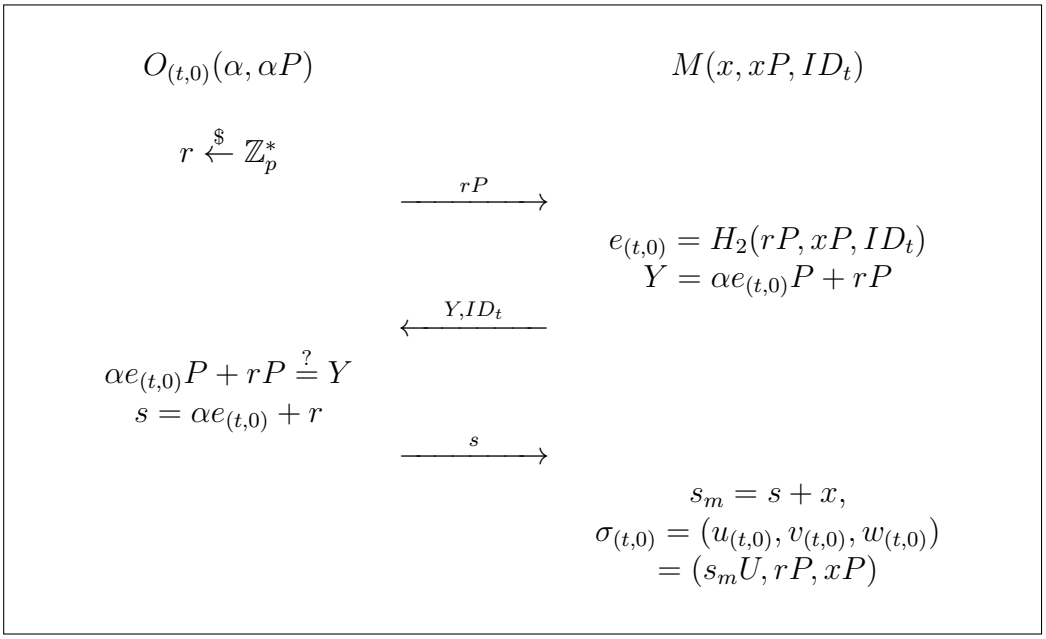

Figure 4.3: Ownership initiation.

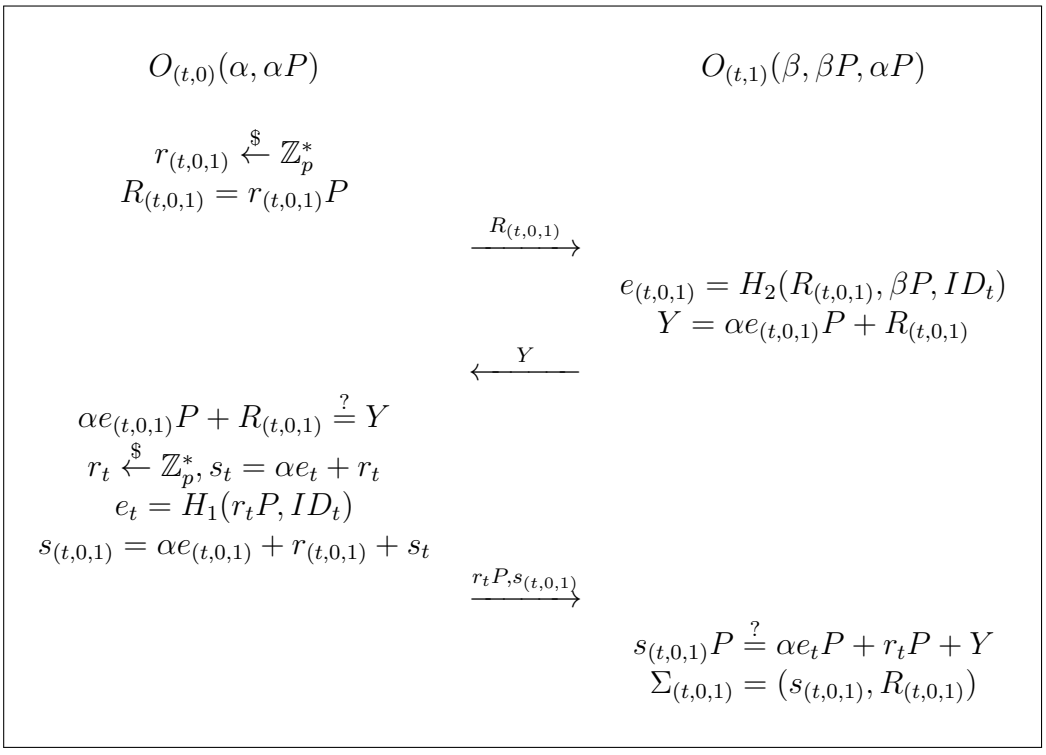

Figure 4.4: Transfer from supplier to new owner.

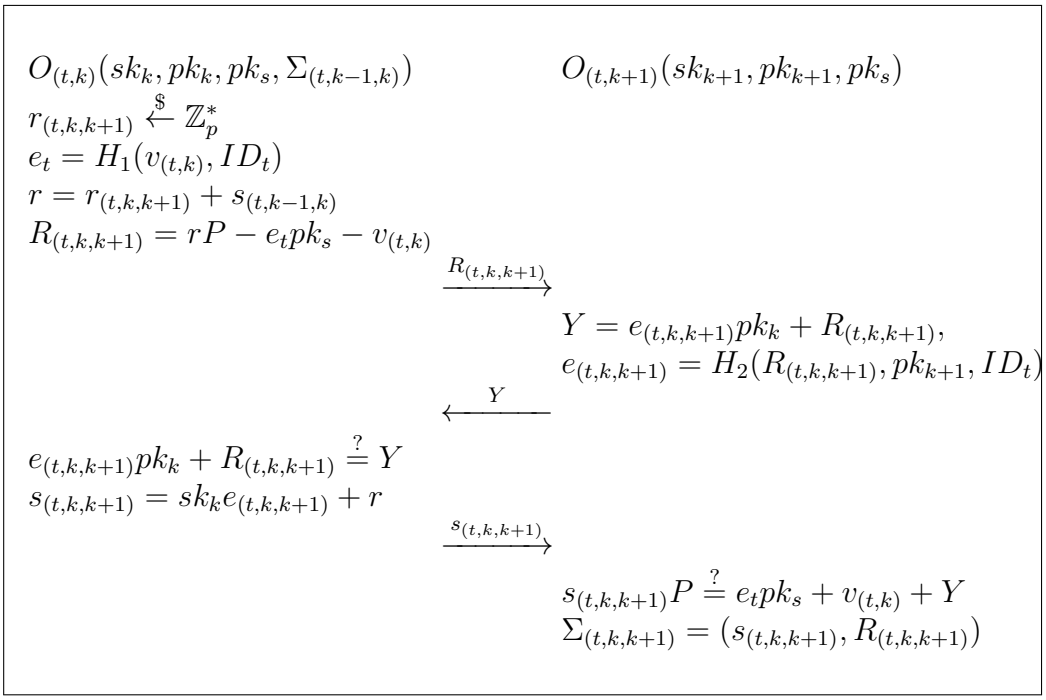

Figure 4.5: General transfer from current owner to new owner on level k.

is the private key of $O_{(t,k)}$, and sets the proof

$$\sigma_{(t,k)} = (u_{(t,k)}, v_{(t,k)}, w_{(t,k)}) = (s_{(t,k)}U, v_{(t,k-1)}, R_{(t,k-1,k)}).$$

In the case $k = 1$, set $v_{(t,1)} = r_t P$, where $r_t P$ is from Figure 4.4.

- OwnerVerify: On input a proof $\sigma_{(t,k)} = (u_{(t,k)}, v_{(t,k)}, w_{(t,k)})$ of tag T, there are three cases. The verifier checks as follows

  – Case 1 ($k = 0$):

  $$\hat{e}_{(t,0)} = H_2(v_{(t,0)}, w_{(t,0)}, ID_t), \quad \hat{e}(P, u_{(t,0)}) \stackrel{?}{=} \hat{e}(e_{(t,0)} pk_s + v_{(t,0)} + w_{(t,0)}, U).$$

  – Case 2 ($k = 1$):

  $$\hat{e}_t = H_1(v_{(t,1)}, ID_t), \quad \hat{e}_{(t,0,1)} = H_2(w_{(t,1)}, pk_1, ID_t),$$

  $$\hat{e}(P, u_1) \stackrel{?}{=} \hat{e}(pk_1 + (e_t + e_{(t,0,1)}) pk_s + v_{(t,1)} + w_{(t,1)}, U).$$

  – Case 3 ($k > 1$):

  $$\hat{e}_t = H_1(v_{(t,k)}, ID_t), \quad \hat{e}_{(t,k-1,k)} = H_2(w_{(t,k)}, pk_k, ID_t),$$

  $$\hat{e}(P, u_{(t,k)}) \stackrel{?}{=} \hat{e}(e_t pk_s + pk_k + e_{(t,k-1,k)} pk_{k-1} + v_{(t,k)} + w_{(t,k)}, U).$$

Outputs *true* if any equation holds, otherwise outputs $false$.

Correctness. Without loss of generality, we show the correctness of our RFID ownership transfer protocol in Case 3 as follows:

$$\begin{aligned}
&\hat{e}(P, u_{(t,k)}) = \hat{e}(P, (s_{(t,k-1,k)} + sk_k)U) \\
&= \hat{e}(P, (sk_{k-1}e_{(t,k-1,k)} + r + sk_k)U) \\
&= \hat{e}(e_{(t,k-1,k)}pk_{k-1} + rP + pk_k, U) \\
&= \hat{e}(e_t pk_s + r_t P + e_{(t,k-1,k)}pk_{k-1} + rP + pk_k - e_t pk_s - r_t P, U) \\
&= \hat{e}(e_t pk_s + pk_k + e_{(t,k-1,k)}pk_{k-1} + v_{(t,k)} + w_{(t,k)}, U).
\end{aligned}$$

4.4 Security Models of Ownership Transfer Protocols

The security of a RFID ownership transfer protocol usually relies on the underlining authentication protocols. It is extremely hard to provide strong security if a symmetric-key authentication protocol is employed. Typically, the security model of symmetric-key based ownership transfer protocols does not provide corruption oracle which outputs the state of a tag. Once the key is exposed, the security of the tag is completely compromised. Elkhiyaoui, Blass and Molva [EBM11] recently presented a ROTIV protocol secure against the key corruption. It applies the public key cryptography in the authentication while the tag is only required to compute a hash function. However, the proposed security model cannot capture the adversary who can rewrite the content of a tag. It is possible when an adversary gains the key of tag. In this section, we enhance the security models of ownership transfer protocols. A general assumption is that owners are not able to launch collusion attacks in an ownership transfer [NSMSN11].

4.4.1 Adversaries and Oracles

The ability of the adversary is essentially restricted by the actions that he is allowed to carry out. In security models, we specify the actions of adversary via the oracle queries. We now define the oracles which are used in the security models of ownership transfer protocols in this chapter.

Definition 4.1 (Oracles)**.** *The adversary plays with a challenger by given public information of the system and the following oracle calls.*

- $(O, pk) \leftarrow \mathsf{SetupOwner}(ID)$: *Taking as input an identity* ID*, it creates an owner* O *and runs the algorithm* KeyGen *to output a public key* pk*.*

- $T \leftarrow \mathsf{TagInit}(ID_t)$: *Creates a tag* T *with the identity* ID_t *and sets the authentication key* y*. It runs the algorithm* TagInit *and outputs the tag* T*.*

- $(ID_t, \sigma_{(t,k)}) \leftarrow \mathsf{Auth}(T, O_k)$: *Taking as input a current owner* O_k *and a tag* T*, it outputs the identity* ID_t *of tag and its ownership proof* $\sigma_{(t,k)}$ *if* T *is valid, outputs* $\perp$ *otherwise.*

- $c \leftarrow \mathsf{CorruptTag}(T)$: *Taking as input a tag* T*, and outputs the complete internal state* c *of* T*. Note that the oracle does not destroy the tag* T *and the tag is available in the future oracle calls.*

- $sk \leftarrow \mathsf{CorruptOwner}(ID)$: *Taking as input an owner's identity* ID*, and outputs the private key* sk *of the owner.*

- $\{0, 1\} \leftarrow \mathsf{Rewrite}(T, c', y)$: *Taking as input a tag* T*, a new state* c' *and an authentication key* y*, it rewrites the state by* c' *and outputs* 1 *if the key is valid,* 0 *otherwise.*

- $\sigma_{(t,k)} \leftarrow \mathsf{OwnerProve}(T, ID_s, ID_{k-1}, ID_k)$: *Taking as input a tag* T*, an identity* ID_s *of supplier, an identity* ID_{k-1} *of previous owner and an identity* ID_k *of current owner, it outputs an ownership proof* $\sigma_{(t,k)}$ *of the tag.*

- $\Sigma_{(t,k,k+1)} \leftarrow \mathsf{Transfer}(T, ID_k, ID_{k+1})$: *Taking as input a tag* T*, an identity* ID_k *of current owner and an identity* ID_{k+1} *of new owner, it outputs an authenticated ownership transfer proof* $\Sigma_{(t,k,k+1)}$ *of the tag.*

Definition 4.2 (Type I and Type II adversary)**.** *The adversary is defined by the oracle calls and the goal of the experiment.*

- Type I Adversary$(\mathcal{A}_I)$: *is allowed to query above oracles except the* CorruptOwner*. It aims to output a valid proof of authenticated transfer which cannot be detected during the transfer.*

- Type II Adversary$(\mathcal{A}_{II})$: *is allowed to query all above oracles. It aims to output a valid proof of ownership of the target tag which cannot be detected in the ownership verification.*

4.4.2 Security Models

We define the security models of ownership transfer protocols in this section. Each model captures the capability of different adversaries. A security model is defined

Experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_I,\mathcal{S}}[\lambda, r, s, n]$:

- **Setup:** The challenger runs the algorithm Setup to generate public parameters *params* and returns to $\mathcal{A}_I$. It initiates a supplier S^*.
- **Phase 1(Learning):**
  - $\mathcal{A}_I$ can query all above oracles except CorruptOwner to C.
  - Outputs two sets $\mathcal{T} = \{T_1, \ldots, T_n\}$ and $\mathcal{O} = \{O_1, \ldots, O_r\}$, which are created tags and owners.
- **Phase 2(Forge):**
  - $\mathcal{A}_I$ submits a target tag $\mathsf{T}^* \in \mathcal{T}$, current owner O^*_k and new owner O^*_{k+1} to $\mathcal{C}$, such that $(O^*_k, O^*_{k+1}) \in \mathcal{O} \cup \{S^*\}$.
  - $\mathcal{A}_I$ queries oracles Auth, CorruptTag, Rewrite, Transfer and OwnerProof to $\mathcal{C}$.
  - $\mathcal{A}_I$ outputs a proof Σ^* of authenticated ownership transfer.

Exp outputs success if $true \leftarrow \mathsf{OwnershipVerify}(ID^*_t, pk^*_s, pk^*_k, pk^*_{k+1}, \mathsf{OwnerProve}(ID^*_t, sk^*_{k+1}, \Sigma^*, \sigma^*))$, such that $\Sigma^* \nleftarrow Transfer(ID^*_t, ID^*_k, ID^*_{k+1})$.

Figure 4.6: Type I security experiment of the ownership transfer protocols.

as an experiment which plays between the adversary and the challenger. We denote that the security parameters as r, s and n, which are respectively the number of owner initiations, the number of oracle calls and the number of tag initiations. There are two experiments defined in our security model. An RFID ownership transfer protocol is secure iff it is secure in both experiments. The security models defined in this section are suitable to ownership transfer protocols in the two-party model.

4.4.2.1 Security Against Type I Attack

Type I adversary is a person who attempts to forge a valid proof of authenticated transfer. $\mathcal{A}_I$ interacts with the challenger via oracle calls and outputs a proof of transfer. It is described as in experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_I,\mathcal{S}}[\lambda, r, s, n]$ in Figure 4.6.

Definition 4.3. *An ownership transfer protocol is (r, s, n, ϵ)-secure against the Type I attack, if any $\mathcal{A}_I$ who succeeds in $\mathbf{Exp}^{secure}_{\mathcal{A}_I,\mathcal{S}}[\lambda, r, s, n]$ has advantage*

$$\Pr[success \leftarrow \mathbf{Exp}^{secure}_{\mathcal{A}_I,\mathcal{S}}[\lambda, r, s, n]] \leq \epsilon,$$

where ϵ is negligible in λ.

Experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_{II},\mathcal{S}}[\lambda, r, s, n]$:

- **Setup:**The challenger runs the algorithm Setup to generate public parameters *params* and returns to $\mathcal{A}_{II}$. It initiates a supplier S^*.
- **Phase 1(Learning):**
  - $\mathcal{A}_{II}$ can query all above oracles to $\mathcal{C}$.
  - Outputs two sets $\mathcal{T} = \{T_1, \ldots, T_n\}$ and $\mathcal{O} = \{O_1, \ldots, O_r\}$, which are created tags and owners.
- **Phase 2(Forge):**
  - $\mathcal{A}_{II}$ submits a target tag $\mathsf{T}^* \in \mathcal{T}$, previous owner $\mathsf{O}^*_{\mathsf{k-1}}$ and current owner $\mathsf{O}^*_{\mathsf{k}}$ to $\mathcal{C}$, such that $(\mathsf{O}^*_{\mathsf{k-1}}, \mathsf{O}^*_{\mathsf{k}}) \in \mathcal{O} \cup \{S^*\}$.
  - $\mathcal{A}_{II}$ queries oracles Auth, CorruptTag, Rewrite, Transfer and OwnerProve to $\mathcal{C}$.
  - $\mathcal{A}_{II}$ outputs a proof σ^* of ownership.

Exp outputs *success* if it satisfies the restrictions as follows,

1. $true \leftarrow \mathsf{OwnershipVerify}(ID^*_t, pk^*_s, pk^*_{k-1}, pk^*_k, \sigma^*)$,
2. $\sigma^* \nleftarrow \mathsf{OwnerProve}(\mathsf{ID}^*_{\mathsf{t}}, \mathsf{ID}^*_{\mathsf{s}}, \mathsf{ID}^*_{\mathsf{k-1}}, \mathsf{ID}^*_{\mathsf{k}})$,
3. $sk^*_k \nleftarrow \mathsf{CorruptOwner}(ID^*_k) \vee (\Sigma^* \nleftarrow \mathsf{Transfer}(ID^*_t, ID^*_{k-1}, ID^*_k) \wedge sk^*_{k-1} \nleftarrow \mathsf{CorruptOwner}(ID^*_{k-1}))$.

Figure 4.7: Type II security experiment of the ownership transfer protocols.

4.4.2.2 Security Against Type II Attack

The Type II adversary acts as a person who attempts to forge a valid proof of ownership. $\mathcal{A}_{II}$ interacts with the challenger $\mathcal{C}$ via oracle calls and outputs a proof of ownership at the end of the experiment. The experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_{II},\mathcal{S}}[\lambda, r, s, n]$ is defined as in Figure 5.7.

Definition 4.4. *An ownership transfer protocol is (r, s, n, ϵ)-secure against the Type II attack, if any $\mathcal{A}_{II}$ who succeeds in $\mathbf{Exp}^{secure}_{\mathcal{A}_{II},\mathcal{S}}[\lambda, r, s, n]$ has advantage*

$$\Pr[success \leftarrow \mathbf{Exp}^{secure}_{\mathcal{A}_{II},\mathcal{S}}[\lambda, r, s, n]] \leq \epsilon,$$

where ϵ is negligible in λ.

Lemma 4.1. *If an ownership transfer protocol is secure against the Type II attack, it is secure against the Type I attack.*

Proof. Suppose that there is a Type I adversary $\mathcal{A}_I$ who breaks the ownership transfer protocol with a non-negligible advantage ϵ. We can construct the algorithm $\mathcal{B}$ to win the experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_{II},\mathcal{S}}[r,s,n]$. $\mathcal{B}$ selects an current owner whose private key is known. It can easily run the algorithm OwnerProve to output a valid proof. That is, the adversary has similar probability to break the ownership transfer protocol in Type II attack. Hence, If an ownership transfer protocol is secure against the Type II attack, it is also secure against the Type I attack. □

4.5 Security Analysis

An ownership transfer protocol is secure if it is against two types of attacks defined in Section 4.4.2. Without loss of generality, we analyse the security of proposed protocol on the k-th level. According to Lemma 4.1, we only show the security proof of the proposed protocol in Type II experiment.

Theorem 4.1. *The proposed ownership transfer protocol is (r,s,n,ϵ)-secure against the Type II attack if the ECDH assumption is held.*

Proof. Assume that there is an Type II adversary $\mathcal{A}_{II}$ who can (r,s,n,ϵ)-break our ownership transfer protocol. We can construct an algorithm $\mathcal{B}$ to solve the ECDH problem. The algorithm $\mathcal{B}$ is given a ECDH instance (P, aP, bP) and aims to output abP. $\mathcal{B}$ sets up the Type II security experiment and interacts with $\mathcal{A}_{II}$. It answers the oracles queries as follows.

- Setup: $\mathcal{B}$ sets g and $U = aP$ as two generators of a group $\mathbb{G}$. It gives public parameter *params* to the adversary. Let $F : \{0,1\}^* \rightarrow \{0,1\}^l$ be a public hash function and $\mathcal{S}$ be a space of symmetric keys. $\mathcal{B}$ sets the supplier's public key $pk_s = bP$ and manufacturer's public key $pk_m = xP$, where b, x are unknown to $\mathcal{B}$. $\mathcal{B}$ maintains a database $D = \{< ID_t, ID, \sigma = (u,v,w), c = (y, F(\sigma)), r_tP >\}$, a list $L_{H1} = \{< m = (R, ID), n >\}$ and $L_{H2} = \{< m = (R, U, ID), z >\}$ which are initially empty. $\mathcal{B}$ adds the supplier $< ID_s, bP, 1, coin = 0 >$ into the list $L_O = \{< ID, pk, r, coin >\}$.

- H_1 Query: On input query a message m_i, where $m_i = (R_i, ID_i)$, $\mathcal{B}$ outputs n_i if $< m_i, n_i >$ appears in the list L_{H1}. Otherwise, $\mathcal{B}$ randomly chooses $n_i \overset{\$}{\leftarrow} \mathbb{Z}^*_P$ and sets $H(m_i) = n_i$. It outputs n_i and adds the item $< m_i, n_i >$ into the list L_{H1}.

- H_2 Query: On input query a message m_i, where $m_i = (R_i, pk_i, ID_i)$, $\mathcal{B}$ outputs z_i if $< m_i, z_i >$ appears in the list L_{H2}. Otherwise, $\mathcal{B}$ randomly selects $z_i \overset{\$}{\leftarrow} \mathbb{Z}^*_P$ and sets $H(m_i) = z_i$. It outputs z_i and adds the item $< m_i, z_i >$ into the list L_{H2}.

- SetupOwner: On input query identity ID_i, $\mathcal{B}$ outputs pk_i if ID_i is in the list L_O. Otherwise, $\mathcal{B}$ tosses a coin and outputs,

$$\begin{cases} coin_i = 0 : \ pk_i = r_i bP \\ \\ coin_i = 1 : \ pk_i = r_i P \end{cases},$$

  where $r_i \xleftarrow{\$} \mathbb{Z}_p^*$, the probability is $\Pr[coin = 0] = \rho$ and $\Pr[coin = 1] = 1 - \rho$. Then, $\mathcal{B}$ adds $< ID_i, pk_i, r_i, coin_i >$ into the list L_O

- TagInit: On input query identity ID_{t_i}, $\mathcal{B}$ outputs T_i if ID_{t_i} is in the Database D. Otherwise, $\mathcal{B}$ sets the supplier to be the owner of the tag. It randomly chooses $y_i \in \mathcal{S}$ and $z_i, s_i \in \mathbb{Z}_p^*$. $\mathcal{B}$ calculates $R_i = s_i P + z_i pk_s - pk_m$ and sets $\sigma_{(t_i,0)} = (s_i, R_i, pk_m)$ and $H_2(m_i) = z_i$, where $m_i = (R_i, pk_m, ID_{t_i})$. Then, $\mathcal{B}$ computes the hash $F(\sigma_{(t_i,0)})$ and sets $c_i = (y_i, F(\sigma_{(t_i,0)}))$. It outputs T_i and adds $< ID_{t_i}, ID_s, \sigma_{(t_i,0)}, c_i, \cdot >$ into the database D and adds $< m_i, z_i >$ into the list L_{H2}.

- Auth: On input query a tag T_i and an owner O_k, $\mathcal{B}$ submits the query to Auth algorithm and returns the output $(ID_{t_i}, \sigma_{(t_i,k)})$ if T_i is authenticated, otherwise outputs $\perp$.

- CorruptTag: On input query tag T_i, $\mathcal{B}$ finds the ID_{t_i} from the database D and outputs c_i if $< ID_{t_i}, ID_i, \sigma_i, c_i >$ is in the database D. Otherwise, it outputs $\perp$.

- CorruptOwner: On input query owner's identity ID_i, $\mathcal{B}$ looks up the list L_O. If $< ID_i, pk_i, r_i, \ coin = 1 >$ appears in the list, $\mathcal{B}$ outputs $sk_i = r_i$, outputs $\perp$ otherwise.

- Rewrite: On input query tag T_i, a new state c_i' and a key y_i, $\mathcal{B}$ checks if the item $< ID_{t_i}, \cdot, \cdot, c_i = (y_i, \cdot) >$ is in the database D. If it appears, $\mathcal{B}$ sets $c_i = c_i'$ and outputs 1, otherwise rejects and outputs 0.

- OwnerProve: On input query tag T_i, an identity ID_{k-1} of the previous owner and an identity ID_k of the current owner, $\mathcal{B}$ retrieves the key pairs (pk_{k-1}, sk_{k-1}) and (pk_k, sk_k) from the list L_O. If $ID_{k-1} = ID_s$, $\mathcal{B}$ firstly issues the oracle query Transfer(ID_s, ID_k, T_i). Depending on the value of $coin_i$, $\mathcal{B}$ computes as follows,

  - Case 1 ($coin_{k-1} = 1 \wedge coin_k = 1$): Taking as input private keys (sk_{k-1}, sk_k) and the tag's identity ID_{t_i}, $\mathcal{B}$ runs the algorithm Transfer and OnwerProve.
  - Case 2 ($coin_{k-1} = 0 \vee coin_k = 0$): We show the response in case of $coin_{k-1} = 0 \wedge coin_k = 0$. The other two cases are similar and we ignore here. Firstly, $\mathcal{B}$

retrieves $r_{ti}P$ from the database D and sets $v_{(t_i,k)} = r_{ti}P$. $\mathcal{B}$ randomly picks $s_i, z_i \xleftarrow{\$} \mathbb{Z}_p^*$ and computes

$$R_i = s_iP - n_ipk_s - pk_k - z_ipk_{k-1} - v_{(t_i,k)},$$

where $< m_i' = (v_{(t_i,k)}, ID_{t_i}), n_i = H_1(m_i') >$ appears in the list L_{H1}. Then, $\mathcal{B}$ sets $m_i = (R_i, pk_k, ID_{t_i})$, $H_2(m_i) = z_i$, $w_{(t_i,k)} = R_i$, $u_{(t_i,k)} = s_iU$. It adds the item $< m_i, z_i >$ into the list L_{H2}.

Finally, $\mathcal{B}$ outputs $\sigma_{(t_i,k)} = (u_{(t_i,k)}, v_{(t_i,k)}, w_{(t_i,k)})$ and replaces the item $< ID_{t_i}, \cdot, \cdot, \cdot, \cdot >$ as $< ID_{t_i}, ID_k, \sigma_{(t_i,k)}, c_i = (y_i, F(\sigma_{(t_i,k)}), r_{ti}P) >$.

- Transfer: On input query an identity ID_k of current owner, an identity ID_{k+1} of new owner, and a tag T_i, $\mathcal{B}$ looks up the list L_O and retrieves the item $< ID_k, pk_k, r_k, coin_k >$.
  - If $ID_k = ID_s$, $\mathcal{B}$ randomly selects $n_i, s_i \in \mathbb{Z}_p^*$ and computes $r_{ti}P = s_iP - n_ipk_s$, such that $H(m_i') = n_i$, $m_i' = (r_{ti}P, ID_{t_i})$. It sets the item $< ID_{t_i}, \cdot, \cdot, \cdot, r_{ti}P >$ in D and adds $< m_i, n_i >$ into the list L_{H1}.
  - If $coin_k = 1$, $\mathcal{B}$ outputs an authenticated transfer proof by running Transfer algorithm.
  - Otherwise, $\mathcal{B}$ randomly selects $s_i, z_i \in \mathbb{Z}_p^*$, $R_j \xleftarrow{\$} \mathbb{G}$ and computes

$$R_i = s_iP - n_ipk_s - z_ipk_k - v_{(t,k)},$$

where $< m_i' = (v_{(t_i,k)}, ID_{t_i}), n_i = H_1(m_i') >$ appears in the list L_{H1}. Then $\mathcal{B}$ sets $m_i = (R_i, pk_{k+1}, ID_{t_i})$, $H_2(m_i) = z_i$, $\Sigma_{(t_i,k,k+1)} = (s_i, R_i)$. It outputs $\Sigma_{(t_i,k,k+1)}$ and adds $< m_i, z_i >$ into the list L_{H2}.

- Forge: At the end of the experiment, $\mathcal{A}_{II}$ succeeds if and only if he outputs a forgery of ownership proof

$$\sigma^*_{(t^*,k)} = (u^*_{(t^*,k)}, v^*_{(t^*,k)}, w^*_{(t^*,k)}),$$

such that

$$true \leftarrow \mathsf{OwnershipVerify}(ID^*_{t_i}, pk^*_s, pk^*_{k-1}pk^*_k, \sigma^*_{(t^*,k)}),$$

$$sk^*_k \nleftarrow \mathsf{CorruptOwner}(ID^*_k) \vee (\Sigma^*_{(t^*,k-1,k)} \nleftarrow \mathsf{Transfer}(ID^*_t, ID^*_{k-1}, ID^*_k) \wedge$$

$$sk^*_{k-1} \nleftarrow \mathsf{CorruptOwner}(ID^*_{K-1}))$$

and

$$\sigma^*_{(\mathsf{t}^*,\mathsf{k})} \nleftarrow \mathsf{OwnerProve}(\mathsf{ID}^*_\mathsf{t}, \mathsf{ID}^*_\mathsf{s}, \mathsf{ID}^*_{\mathsf{k}-1}, \mathsf{ID}^*_\mathsf{k}).$$

According to the forking lemma, $\mathcal{B}$ then uses the same random tape and a different hash function H_2' to get another valid proof $\sigma^{*\prime}_{(t^*,k)} = (u^{*\prime}_{(t^*,k)}, v^{*\prime}_{(t^*,k)}, w^{*\prime}_{(t^*,k)})$, such that $H_2(m_i^*) = z_i^*, H_2'(m_i^*) = z_i^{*\prime},\ z_i^* \neq z_i^{*\prime}$. Assume that items $< ID_k^*, pk_k^*, r_k^*, 1 >$ and $< ID_{k-1}^*, pk_{k-1}^*, r_{k-1}^*, 0 >$ are in the list L_O. We compute the abP, where

$$abP = \frac{1}{r_{k-1}^* z_i^* - r_{k-1}^* z_i^{*\prime}}(u_{(t^*,k)}^* - u_{(t^*,k)}^{*\prime}) \ ,$$

as the solution of ECDH problem. If we consider a different hash function H_1', such that $H_1(m_i^*) = n_i^*, H_1'(m_i^*) = n_i^{*\prime},\ n_i^* \neq n_i^{*\prime}$, $\mathcal{B}$ can similarly solve the ECDH problem.

Probability. If $\mathcal{A}_{II}$ (r, s, n, ϵ)-breaks our ownership transfer protocol, $\mathcal{B}$ can use $\mathcal{A}_{II}$ to solve the ECDH problem. We analyze the events E_1 and E_2 for $\mathcal{B}$ needed to succeed as follows.

- E_1: $\mathcal{A}_{II}$ does not output $\perp$ as a result of any of CorruptOwner queries.
- E_2: $\mathcal{A}_{II}$ outputs a valid forgery of ownership proof $\sigma^*_{(t^*,k)}$ and $coin_{k-1} = 0$ appears in L_O.

$\mathcal{B}$ succeeds if both of E_1 and E_2 occurs. Let $\mathcal{A}_{II}$ queries CorruptOwner at most s times, we have,

$$\Pr[E_1 \wedge E_2] = (1-\rho)^s \rho\epsilon \leq \frac{1}{s+1}(1 - \frac{1}{s+1})^s \epsilon.$$

□

4.6 Conclusion

In this chapter, we defined a new secure model of ownership transfer protocols. It enhances the existing security models. We provided a definition of RFID ownership transfer and proposed a secure ownership transfer protocol. It achieves a single verification key to all the tags from an owner. The protocol satisfies all the security requirements. A formal proof of our proposed protocol was given.

Chapter 5

Shared RFID Ownership Transfer

The ownership of an object can be represented by the ownership of the RFID tag attached to the object. An ownership could be shared among different parties and should be transferable. Although many RFID ownership transfer protocols were proposed, a shared ownership transfer protocol remains a daunting task with absence of a trusted party. The challenge is also due to the physical restrictions of (passive) tags, which usually have a small non-volatile memory of several hundred bits; therefore asymmetric cryptography is not feasible. In this chapter, we introduce the first shared ownership transfer protocol, which was proposed in [LMSV15b], in the two-party system model. It requires merely hashing computations and has a constant key size. We provide a formal definition of the security model of a shared ownership transfer and prove that our protocol is secure.

5.1 Introduction

In practice, an RFID tag could be owned by two parties. As an example, two people jointly own (and hence, share the ownership of) a car with an RFID tag. Either of them can show the partial ownership of the car, but neither of them can sell the car without the agreement of the other, since he/she is not entitled to sell the car solely based on his/her will, without the other party's consent. Unfortunately, traditional RFID ownership transfer (ROT) protocols (e.g., [FMTRCRDF11a, EBM11, NSMSN11, FA07, LK06b]) does not capture this kind of application. We call an RFID ownership transfer protocol, which supports multiple owners as *a shared RFID ownership transfer* (SROT) protocol.

Intuitionally, an SROT protocol consists of two phases, namely ownership verification and ownership transfer. In the first phase, the current owners jointly prove their ownership of the tag and the potential (new) owners verify the ownership proof. If the verification outputs true, subsequently they request an ownership transfer in the second phase. Upon the completion of a successful transfer, the current owners become previous owners and cannot interpret any further communications of the tag anymore. The new owners become current owners of the tag. The SROT protocol provides forward secrecy, which means that the current owners cannot recover the previous communication flows.

SROT protocols should possess three properties: *secure ownership*, *exclusive ownership* and *secure ownership transfer* [vDMRV09]. These properties are also

considered in ROT, but the attackers of SROT protocols are more powerful than the attackers of ROT protocols, where a dishonest subgroup of current owners can launch collusion attacks.

Passive RFID tags have a very limited computational power. Usually, a tag can only cope with symmetric-key cryptography. The size of non-volatile memory of a tag is normally only several hundred bits; therefore it is not possible to store cryptographic keys for each owner. The power of tag is also not sufficient to support a large memory. Furthermore, passive tags are not tamper-resistant. An attacker can physically disclose the tag and read/write its internal state.

Related Work. Saito, Imamoto and Sakurai [SIS05] proposed an RFID ownership transfer protocol in the two-party model, where the protocol assumes the current owner and the new owner play an ownership transfer game with the tag. The communication between owner and tag is assumed in a backward channel [WSRE03]. However, this assumption is questionable in that the adversary can eavesdrop the channel in a sufficient short range [SIS05].

Another two-party ownership transfer protocol, based on [SM08], was introduced by Song [Son08]. The protocol intends to satisfy privacy and security requirements of ownership transfer. It reduces the cost of the tag's computation and the tag's non-volatile memory. Unfortunately, several attacks were found [vDR08, RRG09, PLHCT+10]. In addition, once a tag is compromised, its privacy and security are totally broken.

An ownership transfer protocol using both symmetric and asymmetric cryptography was put forth by Elkhiyaoui, Blass and Molva [EBM11]. Each owner has a public/private key pair and a symmetric-key which is shared with the tag. During a tag authentication process, the private key is used to decrypt the ciphertext received from the tag. The tag will be accepted if the related plaintext matches that on the database. A feature of this scheme is that the tag does not have to deal with asymmetric computations. However, the privacy of the tag is vulnerable if the adversary is one of the previous owners [Aby12].

Kapoor, Zhou and Piramuthu [KZP11] presented a solution for shared ownership transfer with a TTP. A group of owners share the same authentication key with the tag and the tag also shares a key, which is unknown to the owners, with the TTP. During the period of ownership transfer, the TTP needs to interact with the tag and update the tag key for a group of new owners. It is insecure against the corruption attack described in Section 5.2. The authors also proposed an ownership sharing protocol in the two-party model. Current owners share the ownership of the tag with the new owners so that both current owners and new owners can access the tag after a "transfer". Since the owners share the same authentication key, a subgroup of owners can launch a collusion attack against other owners.

Organization of This Chapter. The rest of this chapter is organized as follows. We illustrate some significant issues related to shared tag ownership transfer in Section 5.2. The notions and cryptographic tools used in this chapter are described in Section 5.3. The system model and the definition of SROT protocols are presented in Section 5.4. In Section 5.5, we describe the concrete construction of our protocol. A security model of SROT protocol is given in Section 5.6, followed by the security proof in Section 5.7. Section 5.8 concludes the chapter.

5.2 Problem Statement

The ownership transfer protocols for RFID systems can be modelled using either a TTP-based model or two-party model [SIS05]. In the TTP based model, the solution is trivial, since the proof of ownership can be provided by the TTP, who can even transfer the ownership to new owners on behalf of current owners. However, this system model requires the availability of a TTP, which can interact with the owner and the tag, during the ownership transfer (e.g., [MSW05, KP12]). This model is undesired due to the scalability problem, because a TTP may need to simultaneously participate in a number of ownership transfers. To eliminate this problem, we adopt an *extended two-party* model. In this model, there are two types of owners, namely the current owner and the new owner. Specifically, it does not require the presence of TTP and supports multiple current/new owners.

In symmetric-key based single-owner ownership transfer protocols, the owner shares an authentication key with the tag. In a multi-owner scenario, each valid owner also shares an authentication key with the tag, while we found some new security issues:

1. *Collusion attack* A trivial solution for multi-owner ownership transfer is that owners share the same key with the tag. However, a dishonest owner can use the knowledge of the key to change the secret of the tag without the agreement of other owners, therefore, he can claim that he is the only owner of the tag. We consider that at most $n-1$ dishonest owners could launch such an attack, where n is the number of owners.

2. *Key size*: To resist the collusion attack in the extended two-party model, each owner can share a separate key with the tag. However, the number of keys to be stored on the tag could become a burden to the tag.

3. *Computational capability*: An alternative solution to collusion attacks is to adopt public key cryptography, which allows the keys to be accumulated into a constant size. Unfortunately, it is too heavy for passive tags.

ID_t:	identity of tag T.
ID_i:	identity of owner O_i.
$ID^c_{t,i}$:	identity of ith current owner of tag T.
$ID^n_{t,i}$:	identity of ith new owner of tag T.
$O^c_{t,i}$:	ith current owner of tag T.
$O^n_{t,i}$:	ith new owner of tag T.
$\mathbb{O}^c_t$:	a set of current owners of tag T.
$\mathbb{O}^n_t$:	a set of new owners of tag T.
$N_{d,j}$:	label of a node of a Merkle tree, where d is the level of the node, j is the index from left to right.
$L_{i,j}$:	label of a leaf of a Merkle tree, where i is the index of the leaf's parent, j means the left/right child.
$\|\|$:	concatenation of two bit strings.
$\{0,1\}^l$:	a space of bit strings where each element is l bits.
$Oracle.\mathsf{Sign}(m, ID)$:	Sign oracle in [BLS04b] on input message m with a specified identity ID.

Table 5.1: Notations of symbols.

4. *Corruption attack* Most (passive) tags are not tamper-resistant. A powerful attacker can read the state of a tag [SSAQ02, Wei00]. It is particularly a problem if only symmetric-key cryptography is applied.

5.3 Preliminaries

In our proposed protocol, we apply the BLS signature scheme [BLS04b] as a tool. A brief description of the BLS signature is as follows.

- BLS.KeyGen: Randomly select $x \xleftarrow{\$} \mathbb{Z}^*_p$ and compute xP, where P is a generator of a cyclic additive group $\mathbb{G}$. The public/private key pair is $(pk, sk) = (xP, x)$.
- BLS.Sig: Given a message $m \in \{0,1\}^*$, compute signature $\sigma = xH(m)$, where H is a full-domain hash function defined in the BLS scheme.
- BLS.Verify: Given a message-signature pair (m, σ) and a public key pk, check $\hat{e}(pk, H(m)) \stackrel{?}{=} \hat{e}(P, \sigma)$.

The notations used in this chapter are depicted in Table 5.1.

5.4 System Model

In this section, we present the formal definition of SROT protocols in the extended two-party model.

A shared RFID ownership transfer scheme defines the following components: Tags, Previous Owners, Current Owners and New Owners. According to the knowledge

of the tag's secret, an owner can be considered as a previous owner, current owner or new owner in different periods.

- **Tags** T_i: A low-cost device which has a small storage and restricted computational capability. A tag stores its current state s_i in a non-volatile memory and can handle hashing.

- **Previous Owners** $\mathbb{O}_t^p$: $\mathbb{O}_t^p$ is a set of previous owners of the tag T_i. A previous owner is one who previously had a partial ownership of the tag or its full ownership if there is only one owner.

- **Current Owners** $\mathbb{O}_t^c$: $\mathbb{O}_t^c$ is a set of current owners of the tag T_i. Each current owner shares the same long-term authentication key K which allows $O_{t,i}^c$ to generate temporary keys and prove its partial ownership and authenticate tag T_i. All current owners can cooperatively transfer the full ownership to a new owner.

- **New Owners** $\mathbb{O}_t^n$: $\mathbb{O}_t^n$ is a set of new owners who intend to obtain the full ownership of tag T_i from the current owners. A new owner $O_{t,i}^n$ challenges all current owners to get the proof of ownership of T_i and requires the full ownership transfer. A new owner can update the long-term key once the transfer is completed.

A shared ownership transfer is run among the target tag, the current owners and the new owners. Current owners initiate a tag with the agreement of a shared state. A new owner sends the ownership transfer request and checks the current ownership prior to playing the transfer. Since owners are normally powerful roles, a secure communication channel is supported during the communication between owners. Our scheme consists of nine algorithms: system setup (Setup), key generation (KeyGen), tag initiation (TagInit), temporary key generation (TKGen), partial ownership prove (PartProve), full ownership prove (FullProve), ownership verify (Verify), ownership transfer (Transfer) and key update (KeyUpd). Current owners, new owners and the tag run the above algorithms as described in Figure 5.1. The definition of these algorithms are depicted as follows.

- $\mathsf{Setup}(\lambda) \rightarrow params$: On input a system security parameter λ, the algorithm outputs a set of public system parameters $params$.

- $\mathsf{KeyGen}(params, ID_t, \mathbb{O}_t^c) \rightarrow (K, K_u, \{sk_i\}, \{A_i\})$: On input public system parameters $params$, an identity ID_t of a tag and a set of current owners $\mathbb{O}_t^c$, the algorithm outputs a long-term authentication key K, an update key K_u, a set of private keys $\{sk_i\}$ and a set of auxiliary parameters $\{A_i\}$ of proof of ownership. It is run by the current owners.

- $\mathsf{TagInit}(T, K, K_u, ID_t) \rightarrow s$: On input a tag T, a long-term authentication key K, an update key K_u and an identity ID_t of T, the algorithm outputs and stores the state s on tag T. It is run by the current owners.

- $\mathsf{TKGen}(T, O^c_{t,i}, K, A_i) \rightarrow (L'_{i,1}, K_T)$: On input a tag T, a current owner $O^c_{t,i}$, a long-term authentication key K and auxiliary parameters A_i, the algorithm outputs a pair $(L'_{i,1}, K_T)$, where K_T is a temporary authentication key. It is run by the tag and a current owner.

- $\mathsf{PartProve}(T, O^c_{t,i}, \mathbb{O}^n, K, ID_t, sk_i, A_i) \rightarrow \sigma$: On input a tag T, a current owner $O^c_{t,i}$, a set of new owners $\mathbb{O}^n$, a long-term authentication key K, a tag's identity ID_t, a private key sk_i and auxiliary parameters A_i, the algorithm outputs a proof σ of partial ownership of T. It is run by a tag, a current owner and the new owners.

- $\mathsf{FullProve}(T, \mathbb{O}^c_t, \mathbb{O}^n, K, ID_t, \{sk_i\}, \{A_i\}) \rightarrow \Sigma$: On input a tag T, a set of current owners $\mathbb{O}^c_t$, a set of new owners $\mathbb{O}^n$, a long-term authentication key K, an identity ID_t of T, a set of private keys $\{sk_i\}$ and a set of auxiliary parameters $\{A_i\}$, the algorithm outputs a full ownership proof Σ of the tag. It is run by a tag, the current owners and the new owners.

- $\mathsf{Verify}(T, \mathbb{O}^n, \sigma, ID_t) \rightarrow \{0, 1\}$: On input a tag T, a proof σ of partial ownership of T, a set of new owners $\mathbb{O}^n$, an identity ID_t of tag, the algorithm outputs 1 if the proof is valid, outputs 0 otherwise. It is run by a tag and the new owners.

- $\mathsf{Transfer}(T, \mathbb{O}^c_t, \mathbb{O}^n, K, K_u, \{sk_i\}, \{A_i\}) \rightarrow (R, \mathbb{O}^n_t)$: On input a tag T, a set of current owners $\mathbb{O}^c_t$, a set of new owners $\mathbb{O}^n$, a long-term authentication key K, an update key K_u, a set of private keys $\{sk_i\}$ and a set of auxiliary parameters $\{A_i\}$ of the proof of ownership, the algorithm outputs a record of full ownership transfer R and transfers the ownership to new owners $\mathbb{O}^n_t$. It is run by a tag, the curent owners and new owners.

- $\mathsf{KeyUpd}(T, \mathbb{O}^c_t, K, K_u, \{sk_i\}, \{A_i\}) \rightarrow (K', K'_u, \{sk'_i\}, \{A'_i\})$: On input a tag T, a set of current owners $\mathbb{O}^c_t$, a long-term authentication key K, an update key K_u, a set of private keys $\{sk_i\}$ and a set of auxiliary parameters $\{A_i\}$ of the proof of ownership, the algorithm outputs a new long-term authentication key K', a new update key K'_u, a new set of private keys $\{sk'_i\}$ and a new set of auxiliary parameters $\{A'_i\}$ of proof of ownership. It is run by a tag and the current owners.

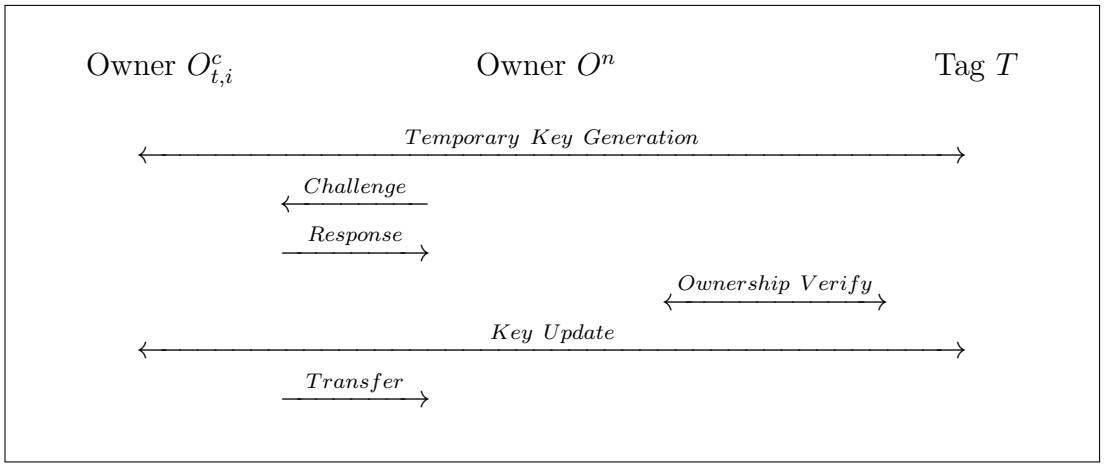

Figure 5.1: Ownership transfer scheme.

5.5 Proposed Protocol

The concrete construction of our protocol is presented in this section. The construction in Section 5.5.1 shows the situation where there are only two owners for a single tag, while our protocol is extendible to a current owner group which contains $1, 2, \cdots, n$ owners.

5.5.1 Constructions

- Setup: Select two cryptographic hash functions $H_1 : \{0,1\}^* \to \{0,1\}^l$ and $H_2 : \mathbb{G}_2 \times \mathbb{N} \times \{0,1\}^* \to \{0,1\}^l$. $H : \{0,1\}^* \to \mathbb{G}_1$ is a hash function defined in the BLS signature scheme. Let $\hat{e}$ be a bilinear map such that $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. P is a random generator of group $\mathbb{G}_2$. The public system parameters *params* are $(H_1, H_2, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, P)$.

- KeyGen: To simplify the algorithm, we show the key generation for two owners. Each current owner $O^c_{t,i}$ randomly chooses a key K_i, where $K_i \in \{0,1\}^l$, and run the algorithm $BLS.\mathsf{KeyGen}$ to generate a key pair $(a_i P, a_i)$, where $a_i \in \mathbb{Z}^*_p$. Sets the private key sk_i as (K_i, a_i) and the verification key $vk_i = a_i P$. $O^c_{t,i}$ computes two leaves $L_{i,1}$ and $L_{i,2}$, such that

$$L_{i,1} = H_1(K_i),\ L_{i,2} = H_2(vk_i, N_o, ID_t),$$

  where $N_o \in \mathbb{N}$ is the number of current owners and ID_t is the identity of tag. $O^c_{t,i}$ broadcasts a tuple $(vk_i, L_{i,1}, L_{i,2})$ to other owners. On receiving a broadcast, an owner verifies

$$L_{i,2} \stackrel{?}{=} H_2(vk_i, N_o, ID_t).$$

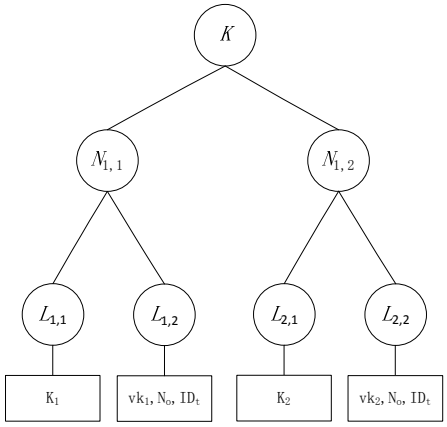

Figure 5.2: Two-owner Merkle tree of tag's long-term authentication key.

If the equation holds, the owner accepts the tuple, otherwise reject. Then, $O^c_{t,i}$ builds a Merkle tree $\mathcal{M}$ depicted in Figure 5.2, where $K = H_1(N_{1,1}||N_{1,2})$, and computes an update key $K_u = L_{1,1} \oplus \cdots \oplus L_{N_o,1}$. Let $\mathcal{P}_i$ be the sibling path of $L_{i,1}$ and set auxiliary parameter $A_i = (vk_i, N_o, \mathcal{P}_i, \mathcal{M})$. It outputs a tuple $(K, K_u, \{sk_i\}, \{A_i\})$. Although Figure 5.2 shows a key construction for two owners, the algorithm can generate a key for any number of owners. Depending on the value of N_o, there are two cases:

- $N_o = 2^n$, $n \in \mathbb{N}$: Each owner generates a Merkle tree on input a set of leaves $\{(L_{i,1}, L_{i,2})\}$, where $i = 1, 2, \cdots, 2^n$.
- $2^n < N_o < 2^{n+1}$, $n \in \mathbb{N}$: Each owner generates the common leaves as follows

$$L_{i,1} = H_1(ID_t),\ L_{i,2} = H_2(P, N_o, ID_t),$$

where $i = N_o + 1, \cdots, 2^{n+1}$. We call such a pair of common leaves $(L_{i,1}, L_{i,2})$ as an *ownership pad*. Note that there is no owner occupies an ownership pad.

- TagInit: Set the tag's state $s = (K, K_u, ID_t)$ and store s in the tag. Any current owner can initiate the tag while other owners need to verify the state. If any owner complains that s is not as expected, the initialization should be run again.

- TKGen: To generate a temporary authentication key K_T, a current owner sends an encrypted sibling path to tag T. If the provided sibling path $\mathcal{P}_i = \{L_{i,1}, L_{i,2}, N_{1,j}, K\}$, where $j = 1$ or $j = 2$, is valid, the tag challenges a random number to the owner. Then, the current owner creates and responds a new key K_T to the tag. T accepts the key if the response is authenticated. The algorithm outputs a pair $(L'_{i,1}, K_T)$. Details are depicted in Figure 5.3.

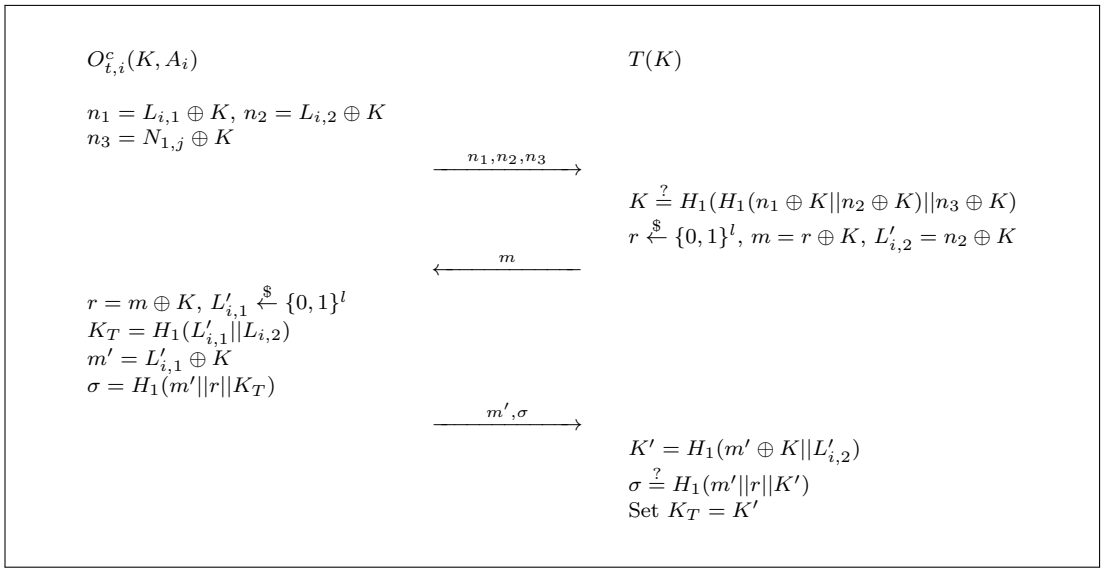

Figure 5.3: Temporary key generation.

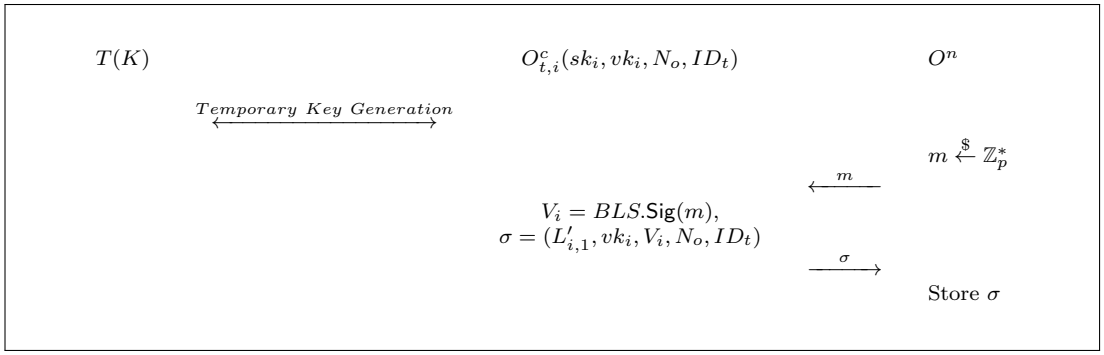

Figure 5.4: Partial ownership proof.

- PartProve: To prove a partial ownership of a tag, a new owner O^n_i challenges a current owner $O^c_{t,i}$. There are two phases in partial ownership proof. In phase 1, $O^c_{t,i}$ runs the algorithm TKGen with target tag T and outputs $(L'_{i,1}, K_T)$. K_T is a one-time authentication key which is erased from the tag after the next successful tag authentication. In phase 2, upon receiving a challenge m, $O^c_{t,i}$ generates a signature by using $BLS.\mathsf{Sign}$. The algorithm outputs a partial ownership proof σ. Details are depicted as in Figure 5.4.

- FullProve: To prove the full ownership of the tag, all owners individually prove the partial ownership by running PartProve algorithm.

- Verify: To verify a proof of partial ownership of a tag, the new owner firstly checks the validity of the signature. If it is valid, the new owner computes the key K_T and uses it to authenticate the tag. The partial ownership is accepted if the authentication succeeds and outputs 1; otherwise rejects and outputs 0. Details of partial ownership verification is described in Figure 5.5. If the new owner requests to check the full ownership, he should verify all the partial ownerships. If proofs

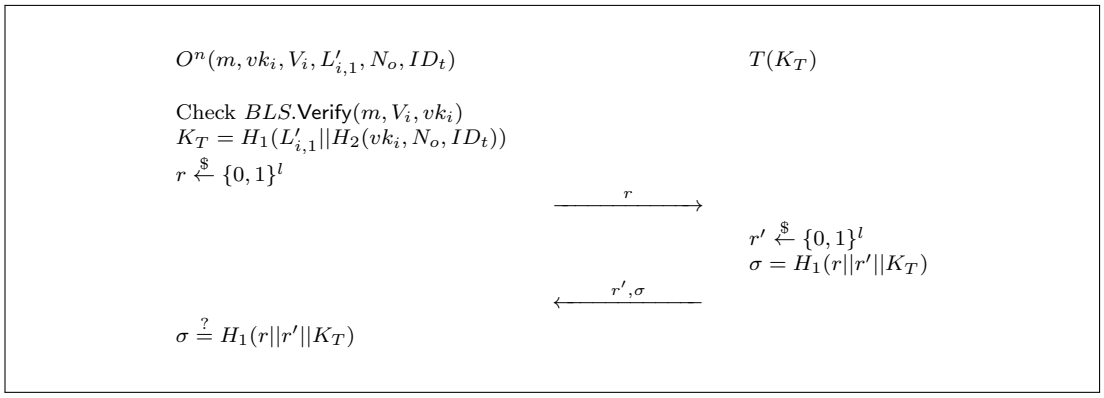

Figure 5.5: Partial ownership verification.

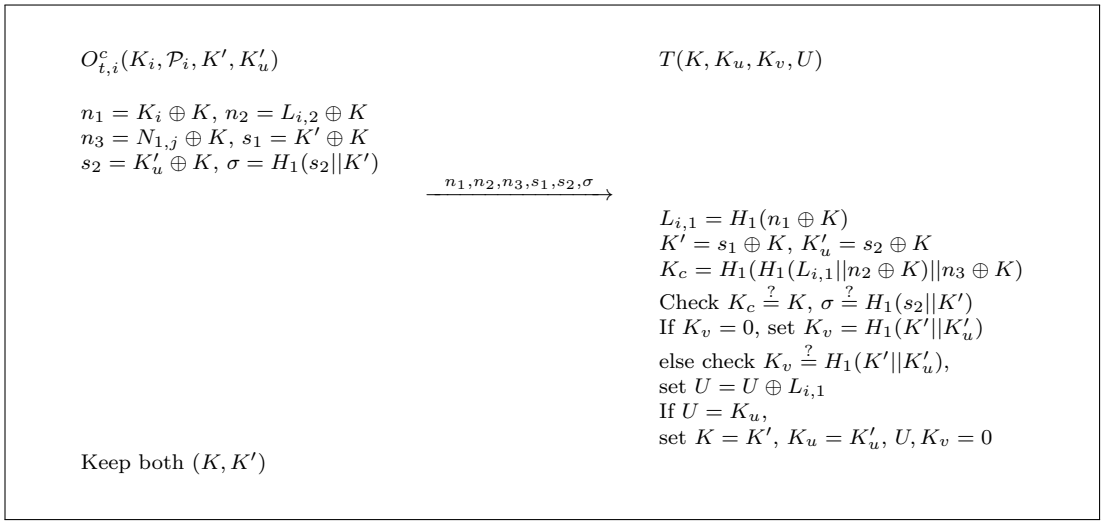

Figure 5.6: Key update protocol.

are valid, new owners accept and output 1, otherwise reject and output 0.

- Transfer: To transfer the full ownership to new owners, current owners run the algorithm KeyUpd on input $(K, K_u, \{K_i\}, \{A_i\})$ and collect outputs $(K', K'_u, \{K'_i\}, \{A'_i\})$. Then each current owner $O^c_{t,i}$ securely transfers a tuple (K', K'_u, K'_i, A'_i) to new owners. We denote R is a record of full ownership transfer, where $R = (K', K'_u, \{K'_i\}, \{A'_i\})$.

- KeyUpd: To update a long-term authentication key of a tag, all current owners need to respectively interact with the tag. During the key update, we assume that a communication between the tag and an owner is in an isolated environment so that no other current owners can eavesdrop the communication flows. First, current owners generate a new long-term authentication key K' and an update key K'_u by running the algorithm KeyGen. Then, they run the key update algorithm as shown in Figure 5.6.

A current owner maintains both the old key K and the new key K' since: 1) it allows the current owner to authenticate the tag during the period when other owners have not agreed the update and 2) it can also resist the desynchronized attack.

5.5.2 Discussion

In the temporary key generation, a current owner should present a valid sibling path to the tag. For n owners, a tag only needs to do $\lfloor \log_2 n \rfloor + 1$ hashing computations in the path checking. At the initiation, a tag stores two keys, namely a long-term authentication key K and an update key K_u. During the protocol execution, a tag stores two more keys K_v and U. Hence, there are at most four keys, which could be 128-bit [Sha08] each, stored in a non-volatile memory. Table 5.2 shows the computaional cost and the key size of proposed protocol.

We generate a public/private key pair in algorithm KeyGen, while an owner could use the same public/private key pair for all his tags. Hence, a record of ownership transfer does not contain the private key of owner. The applied BLS signature scheme is substitutable by any other secure digital signature schemes.

The ownership transfer can be cancelled if any owner denies the request. It is unnecessary to let all owners simultaneously transfer the ownership. Instead they could handle the transfer separately in any sequence.

Table 5.2: Storage and computational cost on tags.

H: hash computation; l: length of a bit-string.

Number of Owners	Temp. Key Generation	Part. Ownership Verification	Key Updating	Key Size
1 Owner	$3H$	$1H$	$4H$	$2l$
n Owners	$\lfloor \log_2 n \rfloor + 3H$	$1H$	$\lfloor \log_2 n \rfloor + 4H$	$2l$

5.6 Security Models of Shared Ownership Transfer Protocols

The security challenges of SROT protocols are different from the security challenges of traditional ownership transfer protocols, where each tag has a single owner at one time. A significant security concern beyond the normal (single-owner) RFID ownership transfer protocol is the *collusion attack*. That is, the attack is mounted by a subset of current owners. Since each owner possesses a part of the secret of tag, we argue that the collusion attack is stronger than attacks which are attempted by

adversaries who are not current owners. Thus, we only consider the *collusion-style* attacks in this section.

5.6.1 Adversaries and Oracles

We define a set of oracles and two collusion-style adversaries who respectively aim at two different goals. In the particular attack, the ability of an adversary is regarded as the actions executed by oracle calls.

Definition 5.1 (Oracles)**.** *The adversary plays with a challenger by given public information of the system and the following oracles.*

- $\mathsf{SetupOwner}(ID^o) \rightarrow O$*: Taking as input an owner's identity* ID^o*, it creates an owner* O*.*
- $\mathsf{TagInit}(ID^t, \{ID_i^o\}) \rightarrow (\{(vk_i, sk_i)\}, T)$*: Taking as input a tag's identity* ID^t *and a set of identities* $\{ID_i^o\}$ *of current owners, it runs the algorithm* KeyGen *to output a set of key pairs* $\{(vk_i, sk_i)\}$*, and run the algorithm* TagInit *to output the tag* T*.*
- $\mathsf{TKGen}(ID^t, ID^o) \rightarrow (L'_{i,1}, K_T)$*: Taking as input a tag's identity* ID^t *and a current owner* ID^o*, it outputs a pair* $(L'_{i,1}, K_T)$ *and stores the temporary authentication key* K_T *into the tag.*
- $\mathsf{CorruptTag}(ID^t) \rightarrow s$*: Taking as input a tag* T*'s identity* ID^t*, it outputs the complete internal state* s *of the tag. Note that the oracle does not destroy the tag and the tag is available in future communications.*
- $\mathsf{CorruptOwner}(ID^o, ID^t) \rightarrow (sk, A)$*: Taking as input an owner's identity* ID^o *and a tag's identity* ID^t*, it outputs the private key* sk *and the auxiliary parameters* A *of owner of tag* T*.*
- $\mathsf{PartProve}(ID^o, ID^t, m_i) \rightarrow \sigma$*: Taking as input an owner's identity* ID^o*, a tag's identity* ID^t *and a challenge* m_i*, it outputs a partial ownership proof* σ *of the tag.*
- $\mathsf{KeyUpd}(ID_i^t) \rightarrow \{0, 1\}$*: Taking as input a tag's identity* ID_i^t*, it updates the keys of the tag and the keys of tag's current owners.*
- $\mathsf{Transfer}(ID^t, \{ID^c\}, \{ID^n\}) \rightarrow \{ID^n\}$*: Taking as input a tag's identity* ID^t*, a set of identities* $\{ID^c\}$ *of current owners and a set of identities* $\{ID^n\}$ *of new owners, it transfers the full ownership of tag* T *to new owners.*

Definition 5.2 (Type I and Type II adversary)**.** *The adversary is defined by the oracle calls and the goal of the experiment.*

- Type I Adversary *($\mathcal{A}_I$): is allowed to query all above oracles except* PartProve. *This adversary is assumed that it has passed the full ownership verification. It aims at transferring the full ownership of the tag to new owners without agreement of all current owners.*

- Type II Adversary *($\mathcal{A}_{II}$): is allowed to query all above oracles except* Transfer. *It aims at outputting a partial ownership proof of a target tag that it cannot be detected during the ownership verification.*

5.6.2 Security Models

The security model of shared ownership transfer protocols is similar to the security model of conventional ownership transfer protocols defined in [LMSV13]. The main difference between the two models is that the adversary of our security model can launch collusion attacks when some of current owners might be dishonest. There are two security models defined in two different experiments, respectively. In an experiment, the ability of a particular adversary is captured by a set of oracle calls and rules of interactions. Each experiment is played between the adversary and the challenger where the adversary issues oracle queries and challenger outputs simulation of oracles. We denote the security parameters r, s, n and $\mathcal{S}$, such that r is the number of owner initiations, s is the number of oracle calls, n is the number of tag initiations and $\mathcal{S}$ is the shared RFID ownership transfer system. The security of a SROT protocol is evaluated under both experiments and we say it is insecure if any adversary succeeds in either experiment. Notice that the experiments in this section are defined for the protocols which do not employ trusted third parties.

5.6.2.1 Security Against Type I Attack

Type I adversary is a person who attempts to cooperate with at most $n-1$ current owners to transfer the full ownership of a tag without agreement of at least one honest owner. $\mathcal{A}_I$ interacts with the challenger via oracle calls and outputs a record R^* of the transfer. $\mathcal{A}_I$ succeeds if the record is valid. The details are described as experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_I,\mathcal{S}}[\lambda, r, s, n]$ in Figure 5.7.

Definition 5.3. *An SROT protocol is (r, s, n, ϵ)-secure against the Type I attack, if any PPT adversary $\mathcal{A}_I$ who succeeds in the experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_I,\mathcal{S}}[\lambda, r, s, n]$ has advantage*

$$\Pr[success \leftarrow \mathbf{Exp}^{secure}_{\mathcal{A}_I,\mathcal{S}}[\lambda, r, s, n]] \leq \epsilon,$$

where ϵ is negligible in λ.

Experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_I,\mathcal{S}}[\lambda, r, s, n]$:

- **Setup:** The challenger $\mathcal{C}$ runs the algorithm Setup to generate public parameters $params$ and returns to $\mathcal{A}_I$.
- **Phase 1(Learning):**
  - $\mathcal{A}_I$ can query oracles SetupOwner, TagInit, TKGen, CorruptTag, CorruptOwner, KeyUdp and Transfer to the challenger.
  - $\mathcal{A}_I$ outputs two sets $\mathsf{T} = \{T_1, \ldots, T_n\}$ and $\mathbb{O} = \{O_1, \ldots, O_r\}$, which are tags and owners.
- **Phase 2(Forge):**
  - $\mathcal{A}_I$ submits a target tag $T^* \in \mathsf{T}$, a set of current owners $\mathbb{O}_t^{c*}$ of T^*.
  - $\mathcal{A}_I$ queries oracles TKGen, CorruptTag, CorruptOwner, KeyUdp and Transfer to $\mathcal{C}$.
  - $\mathcal{A}_I$ outputs a record R^* of full ownership transfer.

Exp outputs *success* if it satisfies the following conditions:

- $\mathsf{Transfer}(T^*, \mathbb{O}_t^{c*}, \mathbb{O}^{n*}) \nrightarrow (\mathbb{O}_t^{n*}, R^*)$
- $\exists\ O_{t,i}^{c*} \in \mathbb{O}_t^{c*},\ s.t.\ \mathsf{CorruptOwner}(ID_i^*, T^*) \nrightarrow (sk_i^*, A_i^*)$
- $\mathsf{KeyUpd}(T^*, \mathbb{O}_t^{n*}, K^*, K_u^*, \{sk_i^*\}, \{A_i^*\}) \rightarrow (K^{*\prime}, K_u^{*\prime}, \{sk_i^{*\prime}\}, \{A_i^{*\prime}\})$

Figure 5.7: Type I security experiment of shared ownership transfer protocols.

5.6.2.2 Security Against Type II Attack

Type II adversary is a person who attempts to cooperate with at most $n-1$ current owners to prove partial ownership of a tag. $\mathcal{A}_{II}$ interacts with the challenger via oracle calls and outputs a partial proof σ^* of ownership of target tag. $\mathcal{A}_{II}$ succeeds if the proof is valid. The details are described as in experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_{II},\mathcal{S}}[\lambda, r, s, n]$ in Figure 5.8.

Definition 5.4. *An SROT protocol is (r, s, n, ϵ)-secure against the Type II attack, if any PPT adversary $\mathcal{A}_{II}$ who succeeds in the experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_{II},\mathcal{S}}[\lambda, r, s, n]$ has advantage*

$$\Pr[success \leftarrow \mathbf{Exp}^{secure}_{\mathcal{A}_{II},\mathcal{S}}[\lambda, r, s, n]] \leq \epsilon,$$

where ϵ is negligible in λ.

Experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_{II},\mathcal{S}}[\lambda, r, s, n]$:

- **Setup:** The challenger $\mathcal{C}$ runs the algorithm Setup to generate public parameters $params$ and returns to $\mathcal{A}_{II}$.
- **Phase 1(Learning):**
  - $\mathcal{A}_{II}$ can query oracles SetupOwner, TagInit, TKGen, CorruptTag, CorruptOwner, KeyUdp and PartProve to the challenger.
  - $\mathcal{A}_{II}$ outputs two sets $\mathsf{T} = \{T_1, \ldots, T_n\}$ and $\mathbb{O} = \{O_1, \ldots, O_r\}$, which are tags and owners.
- **Phase 2(Forge):**
  - $\mathcal{A}_{II}$ submits a target tag $T^* \in \mathsf{T}$, a set of current owners $\mathbb{O}^{c*}_t$ of T^*.
  - $\mathcal{A}_{II}$ queries oracles TKGen, CorruptTag, CorruptOwner, KeyUdp and PartProve to $\mathcal{C}$.
  - $\mathcal{A}_{II}$ outputs ID^*_i's partial proof σ^* of ownership.

Exp outputs *success* if it satisfies the following conditions:

- $\exists\ O^{c*}_{t,i} \in \mathbb{O}^{c*}_t$, $s.t.$ $\mathsf{CorruptOwner}(ID^{o*}_i, ID^{t*}_i) \nrightarrow (sk^*_i, A^*_i)$
- $\mathsf{PartProve}(ID^{o*}_i, ID^{t*}_i, m^*_i) \nrightarrow \sigma^*$
- $\mathsf{Verify}(T^*, \mathbb{O}^{n*}, \sigma^*, ID^{t*}_i) \rightarrow 1$

Figure 5.8: Type II security experiment of shared ownership transfer protocols.

5.7 Security Analysis

A SROT protocol is secure if it is against two types of attacks defined in Section 5.6. We analyse the security of our proposed protocol in both of above two experiments. Without loss of generality, the security proofs in this section is in the case where the number of shared owners is k, such that $k \geq 2$.

Theorem 5.1. *Our SROT protocol is (r, s, n, ϵ)-secure against the Type I attack if the hash function H_1 is preimege resistant.*

Proof. Suppose that there is a Type I adversary $\mathcal{A}_I$ who can (r, s, n, ϵ)-break our shared ownership transfer protocol in experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_I,\mathcal{S}}[r, s, n]$. We can construct an algorithm $\mathcal{B}$, which is run by the challenger, to break the preimege security of underlying hash function H_1 of the protocol. $\mathcal{B}$ sets up the Type I security experiment and interacts with $\mathcal{A}_I$. It answers the oracle queries as follows.

- Setup: $\mathcal{B}$ sets g as a generator of a group $\mathbb{G}_2$. Let two hash functions be $H_1 : \{0,1\}^* \rightarrow \{0,1\}^l$ and $H_2 : \mathbb{G}_2 \times \mathbb{N} \times \{0,1\}^* \rightarrow \{0,1\}^l$. It gives public parameters

params to the adversary. $\mathcal{B}$ maintains a database $D = \{< ID^o, ID^t, c, sk, A >\}$, which is initially empty.

- SetupOwner: On input query an identity ID_i^o, $\mathcal{B}$ creates an owner O_i if the item $< ID_i^o, \cdot, \cdot, \cdot, \cdot >$ is not in the database D and adds $< ID_i^o, \cdot, \cdot, \cdot, \cdot >$ into the database.

- TagInit: On input query a tag's identity ID_i^t and a set of identities $\{ID_i^o\}$ of owners, $\mathcal{B}$ runs the algorithm KeyGen to output $(K_i, K_{u_i}, \{sk_i\}, \{A_i\})$ and sets the tag's state as $c_i = (K_i, K_{u_i})$. It adds items $\{< ID_i^o, ID_i^t, c_i, sk_i, A_i >\}$ in to the database D.

- TKGen: On input query a tag's identity ID_i^t and an owner's identity ID_i^o, if the item $< ID_i^o, ID_i^t, \cdot, \cdot, \cdot >$ does not exist on the database, $\mathcal{B}$ returns $\perp$, otherwise retrieves the item. $\mathcal{B}$ randomly chooses $L'_{i,1} \in \{0,1\}^l$ and computes $K_T = H_1(L'_{i,1}||L_{i,2})$. $\mathcal{B}$ sets the tag's state $c_i = (K_i, K_{u_i}, K_T)$ and returns $(L'_{i,1}, K_T)$ to adversary and updates $< ID_i^o, ID_i^t, c_i, sk_i, A_i >$ in D.

- CorruptTag: On input query a tag's identity ID_i^t, $\mathcal{B}$ retrieves the item $< ID_i^o, ID_i^t, c_i, sk_i, A_i >$ if it appears in the database. Otherwise, $\mathcal{B}$ firstly runs the algorithm TagInit. It returns c_i to the adversary.

- CorruptOwner: On input query an owner's identity ID_i^o and a tag's identity ID_i^t, $\mathcal{B}$ returns $\perp$ if $< ID_i^o, ID_i^t, c_i, sk_i, A_i >$ is not in database, otherwise, $\mathcal{B}$ retrieves it and returns (sk_i, A_i) to adversary.

- KeyUpd: On input query a tag's identity ID_i^t, if ID_i^t does not exist on the database, $\mathcal{B}$ returns 0. Otherwise, $\mathcal{B}$ retrieves all items such that $< \cdot, ID_i^t, \cdot, \cdot, \cdot >$ from D. Then, it runs the algorithm KeyUdp to output new keys $(K', K', \{K'_i\}, \{A'_i\})$. $\mathcal{B}$ updates the items and returns 1.

- Transfer: On input query a tag's identity ID_i^t, a set of identities $\{ID_i^c\}$ of current owner and a set of identities ID^o of owners, $\mathcal{B}$ retrieves items $< ID_i^c, ID_i^t, \cdot, \cdot, \cdot >$ and sets the full ownership transfer record $R_i = (K_i, K_{u_i},$ $\{sk_i\}, \{A_i\})$, then deletes all such items. $\mathcal{B}$ runs the algorithm TagInit on input new owners' identities $\{ID_i^o\}$ and tag's identity ID_i^t and sets $\{ID_i^n\} = \{ID_i^o\}$. $\mathcal{B}$ returns $(R_i, \{ID_i^n\})$ to the adversary.

- Forge: In the end of the experiment, the adversary $\mathcal{A}_I$ succeeds if he outputs a target tag's full ownership transfer record $R^* = (K^*, U^*, \{sk_i^*\}, \{A_i^*\})$ which satisfies the conditions:

  1. Transfer$(T^*, \mathbb{O}_t^{c*}, \mathbb{O}^{n*}) \nrightarrow (\mathbb{O}_t^{n*}, R^*)$;

2. $\exists O_{t,i}^{c*} \in \mathbb{O}_t^{c*}, s.t.\mathsf{CorruptOwner}(ID_i^*, T^*) \nrightarrow (sk_i^*, A_i^*)$;
3. $\mathsf{KeyUpd}(T^*, \mathbb{O}_t^{n*}, K^*, K_u^*, \{sk_i^*\}, \{A_i^*\}) \rightarrow (K^{*\prime}, K_u^{*\prime}, \{sk_i^{*\prime}\}, \{A_i^{*\prime}\})$.

Then, $\mathcal{B}$ can use $\mathcal{A}_I$ to find the preimage of hash function H_1 since $\mathcal{A}_I$ success implies that he outputs a preimage $K_i^* \in sk_i^*$, such that $H_1(K_i^*) = L_{i,1}^*$, $L_{i,1}^* \in A_i^*$. It is contradicted with the preimage resistant assumption of H_1. Therefore, the proposed protocol is secure against the Type I attack.

□

Theorem 5.2. *Our SROT protocol is (r, s, n, ϵ)-secure against the Type II attack if the BLS signature is secure and the Merkle-tree lemma holds.*

Proof. Suppose that there is a Type II adversary $\mathcal{A}_{II}$ who can (r, s, n, ϵ)-break our shared ownership transfer protocol in experiment $\mathbf{Exp}_{\mathcal{A}_{II},\mathcal{S}}^{secure}[r, s, n]$. We can construct an algorithm $\mathcal{B}$, which is run by the challenger, to forge a valid instance of the BLS signature or to output a valid sibling path of the Merkle tree, such that it does not appear in the tree. $\mathcal{B}$ sets up the Type II security experiment and interacts with $\mathcal{A}_{II}$. It simulates and answers oracle queries as follows.

- Setup: $\mathcal{B}$ sets the simulation parameters as in the simulation of the BLS signature in [BLS04b]. Then, $\mathcal{B}$ chooses two hash functions, such that $H_1 : \{0,1\}^* \rightarrow \{0,1\}^l$, $H_2 : \mathbb{G}_2 \times \mathbb{N} \times \{0,1\}^* \rightarrow \{0,1\}^l$. It gives public parameters to the adversary and maintains a database $D = \{< ID^o, ID^t, c, sk, A, coin = \{0,1\} >\}$, which is initially empty.

- SetupOwner: On input query an identity ID_i^o, if ID_i^o does not appear in the database D, $\mathcal{B}$ creates an owner O_i and adds the item $< ID_i^o, \cdot, \cdot, \cdot, \cdot, \cdot >$ into D. Otherwise, $\mathcal{B}$ ignores the request.

- TagInit: On input query a tag's identity ID_i^t and a set of current identities $\{ID_i^o\}$ of current owners, $\mathcal{B}$ randomly picks an index $i \in n$, where n is the number of submitted current owners. For the ith owner, $\mathcal{B}$ firstly runs the algorithm $Oracle$.Setup to output a public/private key pair (vk_i, sk_i). Then, $\mathcal{B}$ follows the algorithm KeyGen, except the public/private key generation, to output $(K_i, K_{u_i}, \{sk_i\}, \{A_i\})$. $\mathcal{B}$ sets the tag's state $c_i = (K_i, K_{u_i})$, $coin_i = 0$, and adds $\{< ID_i^o, ID_i^t, c_i, sk_i, A_i, 0 >\}$ into D. For other $n-1$ owners, $\mathcal{B}$ runs the algorithm KeyGen to output $(K_j, K_{u_j}, \{sk_j\}, \{A_j\})$ and sets the tag's state $c_j = c_i$, $coin_j = 1$, where $j \in n$, $j \neq i$. It adds items such that $\{< ID_j^o, ID_j^t, c_j, sk_j, A_j, 1 >\}$ into the database D.

- TKGen: On input query a tag's identity ID_i^t and an owner's identity ID_i^o, if the item $< ID_i^o, ID_i^t, \cdot, \cdot, \cdot, \cdot >$ does not in D, $\mathcal{B}$ returns $\perp$, otherwise retrieves

the item. $\mathcal{B}$ then randomly picks $L'_{i,1} \in \{0,1\}^l$ and computes $K_T = H_1(L'_{i,1}||L_{i,2})$. Then, $\mathcal{B}$ sets the tag's state $c_i = (K_i, K_{u_i}, K_T)$ and returns $(L'_{i,1}, K_T)$ to adversary and updates $< ID_i^o, ID_i^t, c_i, \cdot, \cdot, \cdot >$ in D.

- CorruptTag: On input query a tag's identity ID_i^t, if ID_i^t is not in D, $\mathcal{B}$ runs the algorithm $TagInit$ first. Then, $\mathcal{B}$ retrieves the item $< ID_i^o, ID_i^t, c_i, sk_i, A_i, coin_i >$ and returns c_i to the adversary.

- CorruptOwner: On input query an owner's identity ID_i^o and a tag's identity ID_i^t, $\mathcal{B}$ returns $\perp$ if $< ID_i^o, ID_i^t, c_i, sk_i, A_i, coin_i >$ does not appear in the database D. If $coin_i = 0$, $\mathcal{B}$ aborts the simulation and returns $\perp$, otherwise, $\mathcal{B}$ retrieves the item and returns (sk_i, A_i) to adversary.

- KeyUpd: On input query a tag's identity ID_i^t, if ID_i^t does not exist on the database, $\mathcal{B}$ returns 0. Otherwise, $\mathcal{B}$ retrieves all items such that $< \cdot, ID_i^t, \cdot, \cdot, \cdot, \cdot >$ from D. Then, it runs the algorithm KeyUdp to output new keys $(K', K', \{K'_i\}, \{A'_i\})$. $\mathcal{B}$ updates the items and returns 1.

- PartProve: On input query an owner's identity ID_i^o, a tag's identity ID_i^t and a challenge m_i, $\mathcal{B}$ retrieves $< ID_i^o, ID_i^t, c_i, sk_i, A_i, coin_i >$ if it appears in D. If $coin = 1$, $\mathcal{B}$ runs the algorithm BLS.Sign to output a signature V_i, otherwise, $\mathcal{B}$ submits (m_i, ID_i^o) to the oracle $Oracle$.Sign and obtains an output signature V_i. Then, $\mathcal{B}$ calls the oracle TKGen to get $(L'_{i,1}, K_T)$. It sets $\sigma_i = (L'_{i,1}, vk_i, V_i, N_o, ID_i^t)$ as a partial ownership proof of ID_i^o and returns it to adversary.

- Forge: At the end of the experiment, the adversary $\mathcal{A}_{II}$ succeeds if he can output a target tag's partial ownership proof $\sigma^* = (L'^*_{i,1}, vk_i^*, V_i^*, N_o^*, ID_i^{t*})$ which satisfies the conditions:

  1. $\exists O_{t,i}^{c*} \in \mathbb{O}_t^{c*}$, $s.t.$CorruptOwner$(ID_i^{o*}, ID_i^{t*}) \nrightarrow (sk_i^*, A_i^*)$;
  2. PartProve$(ID_i^{o*}, ID_i^{t*}, m_i^*) \nrightarrow \sigma^*$;
  3. Verify$(T^*, \mathbb{O}^{n*}, \sigma^*, ID_i^{t*}) \rightarrow 1$.

  Then, $\mathcal{B}$ can use $\mathcal{A}_{II}$ to output a sibling path of Merkle tree $\mathcal{M}^*$ since a valid forgery of K_T^* implies that $\mathcal{A}_{II}$ can output a valid sibling path to convince the tag. Otherwise, it can output a valid forgery of BLS signature and solve its underlying problem if $O_{t,i}^{c*}$ is the owner who has the record $< ID_i^{o*}, ID_i^{t*}, c_i^*, sk_i^*, A_i^*, 0 >$ in the database. It is contradicted with the Merkle-tree lemma [HHPS11] and the security of BLS signature. Hence, the proposed protocol is secure against the Type II attack.

□

5.8 Conclusion

A shared ownership transfer protocol in the extended two-party system model has been proposed assuming the absence of a TTP. Our protocol only requires a tag to store keys with a constant size in order to prove the ownership of a group of owners. We provided the formal definition of shared ownership transfer and proved that our protocol is secure against various attacks including collusion attacks.

Chapter 6

Authorized RFID Authentication

In a traditional RFID application, the system is considered to be controlled by a single party who maintains all the secret information. However, in some practical scenarios, RFID tags, readers and servers could be operated by different parties. Although the private information should not be shared, the system should allow a valid tag to be authenticated by a legal reader. The challenge in designing the system is preserving the tag and the reader's privacy. In this chapter, we propose a novel concept of *authorized RFID authentication* (ARA), which was presented in [LMS+14]. The proposed protocols allow the tag to be merely identifiable by an authorized reader and the server cannot reveal the tag's identity during the reader-server interaction. We provide a formal definition of privacy and security models of authorized authentication protocols under the strong and weak notions and propose three provably secure protocols.

6.1 Introduction

A typical RFID system is established by a single party who initiates the secret keys. To identify a tag, a reader communicates with the tag and sends the tag's response to the back-end server. The server checks the tag's identity by using the shared keys and informs the reader whether the tag is valid.

Many RFID authentication protocols [SM08, Tsu06, JW05, HB01] have been proposed to preserve the tag privacy in conventional systems. These protocols assume that a reader and a server are held by a single entity. However, in some practical scenarios, we found that tag, reader and server are relatively independent, and hence, the existing solutions of RFID authentication protocols are deemed to be impractical. Consider the following scenario.

In an privileged membership club, there are sole facilities provided for their members exclusively, such as restaurant, massage and sauna. Each of these facilities is operated by different business owners, who are paid by the owner of the club, who is also taking membership fees from its members. Hence, these facilities will allow exclusive club members only to access them and enjoy the service provided. In order to provide this benefit to the members, the club issues a membership card that is used to identify each member's identity. Nevertheless, to ensure the privacy of each member, the member would like to ensure that his/her identity will remain private whenever he/she is enjoying those services. Otherwise, these

facilities will not be attractive to the members, if they have to sacrifice their privacy to trade for the facilities offered. In addition, the facilities are also expected to prevent the sensitive customer information from being exposed to the club, even though the members are indeed paying the membership fee to the club. The current solution may sound feasible to be implemented with an RFID system. Nevertheless, the requirement to maintain both privacy and accountability at the same time is seemingly contradictory.

The challenge in designing authentication protocols for the above scenario is the tag and the reader's privacy. A strong tag privacy prevents a tag being linked in two different sessions even if the tag is completely corrupted. Most previous protocols consider the tag untraceability under the assumption that the server is honest and the reader can authenticate all the tags. However, it is suitable to our scenario where the server and the reader are relatively independent. The adversary who plays as an authorized reader can attempt to disclose a tag which is not intended to be identifiable. The reader's privacy is considered as whether the back-end server can reveal the tag's identity during the protocol run. Specifically, the tag is merely identifiable by the authorized reader rather than the server; otherwise the server can obtain the merchant's (reader) client information and trace the tag. Unfortunately, to the best of our knowledge, existing protocols ignore this requirement and there is no protocol that cater the reader's privacy. Therefore, we need new models to evaluate a protocol's privacy and a novel protocol is desired.

Tag impersonation is one of crucial security problems of authentication protocols. Normally, it is hard to resist this attack if the tag is compromised. However, in our system, the untrusted server can cheat the reader without corrupting the tag by using the tag's shared secret. Hence, the protocol needs to prevent abuse of the shared information by the server.

Related Work. Vaudenay [Vau07] proposed a strong privacy model which is considered as the most complete one. The privacy of an RFID tag authentication protocol is classified in several levels which are strong, destructive, forward and weak. Each level is with respect to a different adversary with a set of oracle calls. A strong adversary is allowed to corrupt a tag and continues future interactions with the compromised tag.

Another strong privacy model was introduced by Juels and Weis [JW07]. The model is based on the IND-CCA2 experiment and the adversary of the experiment aims to distinguish two different tags. Later, Hermans, Pashalidis, Vercautern and Preneel [HPVP11] proposed a new practical RFID privacy model. They defined the "left" and "right" world that an adversary needs to decide which world is simulated in the experiment. Many other RFID privacy models (e.g., [CCEG10, BLdMT09, NSMSN08, DLYZ10]) are also presented in the literature.

Nithyanand, Tsudik and Uzun [NTU10, NTU11] considered the reader revocation problem in the public key infrastructure based RFID system. This problem is prominent as the (passive) tag could not check the time information during the protocol execution. The proposed solution requires a tag to equip a date display and a user checks during the certificate verification.

Organization of This Chapter. The rest of this chapter is organized as follows. In Section 6.2, we define the system model and Section 6.3 describes the concrete constructions of three ARA protocols. We introduce the privacy and security models in Section 6.4 and provide the formal privacy and security proof of proposed protocols in Section 6.5. We conclude the chapter in Section 6.6.

6.2 System Model

In this section, we describe the entities of the ARA system and the formal definition of ARA protocols. The system defines the following entities: Tags, Readers and Servers.

- **Tag** T_i: Has a small storage and is not temper-resistant. It stores the keys in a non-volatile memory and requires capabilities to perform hash computations and ECC computations depends on the protocols. It can be considered as a membership card held by the member who initiates the tag's secret key.
- **Reader** R_i: A powerful device which is authorized by a server to authenticate a group of tags with the given period key. R_i is controlled by a merchant who has an individual back-end server.
- **Server** S_i: S_i provides the membership registration for customers and aids the reader to authenticate a tag. The server can authorize the reader to authenticate a group of tags and revoke the reader when it is no longer qualified.

The ARA protocol is executed by tag, reader and server. In the system, a server creates a tag and publishes a set of public information, such as the public key of the server. The member initiates the tag with the server's information and the keys which are chosen by himself. The public key of the tag is given to the server when the card is activated, while the private key is unknown to the server. To authorize a reader, the server generates a period key for the reader. During the tag authentication, the reader needs to cooperate with the server. However, the server cannot discover the identity of the tag which is involved in the session. To revoke a reader, the server can let the reader's period key expire.

Our protocol consists of four algorithms: server key generation (ServerKeyGen), tag key generation (TagKeyGen), reader authorization (ReaderAuth) and tag authentication (Auth). The definition of algorithms are depicted as follows.

- $\mathsf{ServerKeyGen}(k) \rightarrow (PK, SK)$: Taking as input a security parameter k, it generates the server's public/private key pair (PK, SK).
- $\mathsf{TagKeyGen}(T, k) \rightarrow (pk, sk)$: Taking as input a security parameter k for the tag T, it outputs T's public key pk and private key sk.
- $\mathsf{ReaderAuth}(\{pk_i\}, \mathbb{T}_R, sk, R) \rightarrow (rsk, rpk)$: Taking as input a set of public keys $\{pk_i\}$ of tags $\mathbb{T}_R$, the server's private key SK and a reader R, it outputs a secret rsk and the reader's period key rpk. rpk is given to the reader and rsk is given to the server. For each run of this algorithm, the reader's current keys are revoked.
- $\mathsf{Auth}(sk, PK, rsk, rpk) \rightarrow \{T, \perp\}$: The tag takes as input a private key sk and a server's public key PK, a reader takes as input a period key rpk and the server takes as input a secret rsk, it outputs T if the tag is authenticated, $\perp$ otherwise.

6.3 Proposed Protocols

The concrete constructions of proposed ARA protocols are presented in this section. The protocol in Section 6.3.1 is based on symmetric-key cryptography and it achieves basic requirements of ARA protocols. Section 6.3.2 shows the drawbacks of protocol 1 and describe an ECC-based solution which the server handles most computations of the protocol execution. Optionally, Section 6.3.3 introduces a protocol which only requires constant communication cost during the authentication and provides the false output detection. As an overview, Table 6.1 summarizes the security and privacy properties of three protocols along with communication cost, computational efficiency and tag capabilities.

We define three cryptographic hash functions H_1, H_2, H_3, where $H_1 : \{0,1\}^* \rightarrow \{0,1\}^l$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, $H_3 : \{0,1\}^* \rightarrow \mathbb{G}$ and employ the pairing group $(P, Q, p, \hat{e}, \mathbb{G}, \mathbb{G}_T)$. $\mathbb{G}$ is an additive cyclic group and $\mathbb{G}_T$ is a multiplicative group of the same prime order p. P, Q are two generators of group $\mathbb{G}$. The map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a symmetric bilinear mapping.

6.3.1 Protocol 1

Our proposed protocol 1 is based on symmetric-key cryptography. It only requires a tag to compute hash values. The protocol achieves basic privacy requirements of ARA protocols with a relaxed condition. The protocol is presented in Figure 6.1.

	Forward Privacy	Backward Privacy	Reader Privacy	Tag Unforgeability	Constant Com. Cost	Constant Reader Auth.	Tag Cap.
P1	⧫	⧫	√	×	×	√	H
P2	√	√	√	√	×	√	H, PK
P3	√	√	√	√	√	×	H, PK

Table 6.1: Comparison of proposed protocols and tag capabilities.
√: the protocol achieves this property; ×: the protocol cannot provide this property; ⧫: the protocol achieves this property without tag corruption operations; H: requires hash computations; PK: requires ECC computations. Note that tag unforgeability is against a malicious server who cannot corrupt tags.

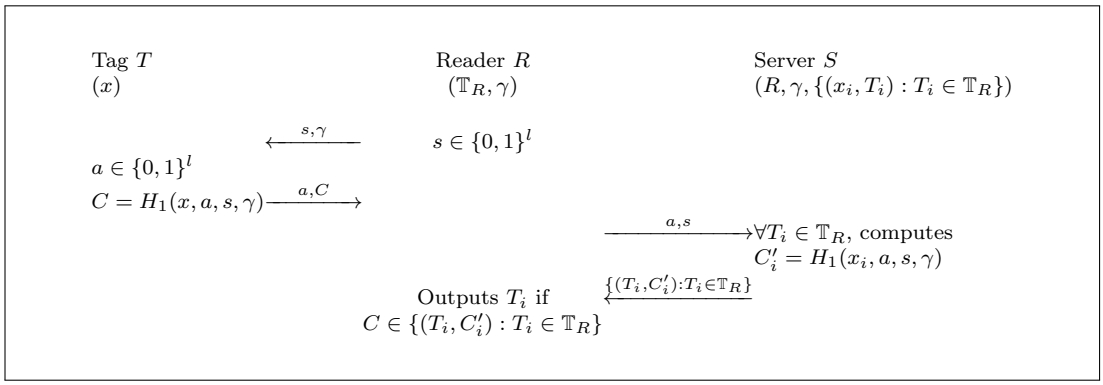

Figure 6.1: Authorized RFID authentication protocol 1.

- ServerKeyGen: The server generates a key space $\mathcal{K}$.
- TagKeyGen: The member randomly chooses $x \in \mathcal{K}$ and sets $(pk, sk) = (\cdot, x)$. The secret key sk is stored in the tag and given to the server.
- ReaderAuth: To authorize the reader R to identify a specified set of tags $\mathbb{T}_R$, the server randomly chooses $\gamma \in \{0,1\}^l$, and sets $(rpk, rsk) = (\gamma, \gamma)$.
- Auth:To authenticate a tag, the tag, reader and server interact as follows

  1. The reader randomly chooses $s \in \{0,1\}^l$ and send (s, γ) to the tag.
  2. Upon receiving (s, γ), the tag selects $a \in \{0,1\}^l$ and sends the reader the response (a, C), where $C = H_1(x, a, s, \gamma)$.
  3. Upon receiving the tag's response C, the reader sends (a, s) to the server.
  4. Upon receiving (a, s), the server retrieves $(\mathbb{T}_R, \gamma)$. For each $T_i \in \mathbb{T}_R$, the server computes $C' = H_1(x_i, a, s, \gamma)$ then sends the reader a set $\{(T_i, C'_i) : T_i \in \mathbb{T}_R\}$.
  5. Finally, the reader outputs T_i if $C \in \{(T_i, C'_i) : T_i \in \mathbb{T}_R\}$.

6.3.2 Protocol 2

In the ARA system model, tag, reader and server are relatively independent. The key of a tag is expected to be unknown by the server since the server could abuse

the key to forge the tag. It is difficult to prevent forging a tag by the server from using symmetric-key based protocols. A trivial solution may be that the tag sends a signed nonce to the reader. However, in this case, the tag's response is publicly verifiable that an adversary can identify the tag by exhaustive public key search. Thus, the tag's identity needs to be concealed and only an authorized reader is entitled to reveal. We then present Protocol 2 with ECC to tackle this issue. The protocol is presented in Figure 6.2.

- ServerKeyGen: The server randomly picks $\alpha \in \mathbb{Z}_p^*$, and sets the public/private key pair $(PK, SK) = (\alpha P, \alpha)$.
- TagKeyGen: The member randomly chooses $x \in \mathbb{Z}_p^*$, and computes the tag's public and private keys $(pk, sk) = (xP, x)$. (P, sk, pk, PK) are stored in the tag and pk is given to the server.
- ReaderAuth: To authorize the reader R to identify a specified set of tags $\mathbb{T}_R$, the server randomly chooses $\gamma \in \mathbb{Z}_p^*$, and sets (rpk, rsk) as

$$rpk = \{\gamma, (T_i, x_i P) : T_i \in \mathbb{T}_R\}, \quad rsk = (\gamma, \alpha).$$

  The server stores $(R, rsk, \mathbb{T}_R)$ and sends rpk to the reader R.
- Auth: To authenticate a tag, the tag, reader and server communicate as follows.
  1. The reader randomly selects $B \in \mathbb{G}$ and sends (B, γ) to the tag.
  2. Upon receiving (B, γ) from the reader, the tag randomly chooses $r \in \mathbb{Z}_p^*$, and computes (w, s, C_1, C_2, C_3). It sends (C_1, C_2, C_3) as the response to the reader. Note that C_1 is to assist reader identify a tag and hide the value s, otherwise the tag's response is publicly verifiable.
  3. Upon receiving (C_1, C_2, C_3), the reader forwards (B, C_3) to the server.
  4. Upon receiving the message (B, C_3) from the reader R, for each $T_i \in \mathbb{T}_R$, the server computes (w', v_i, U_i). Then the server replies $\{(T_i, v_i, U_i) : T_i \in \mathbb{T}_R\}$.
  5. Finally, the reader outputs T_i if $C_1 = U_i$ and $\hat{e}(C_2, (x_i + v_i)P) = \hat{e}(P, P)$, otherwise rejects.

6.3.3 Protocol 3

ARA protocol 2 engages the reader to perform constant computations during the tag authentication. Instead, the server needs to send the reader a set of possible values for tag identification. In some scenarios where the communication bandwith

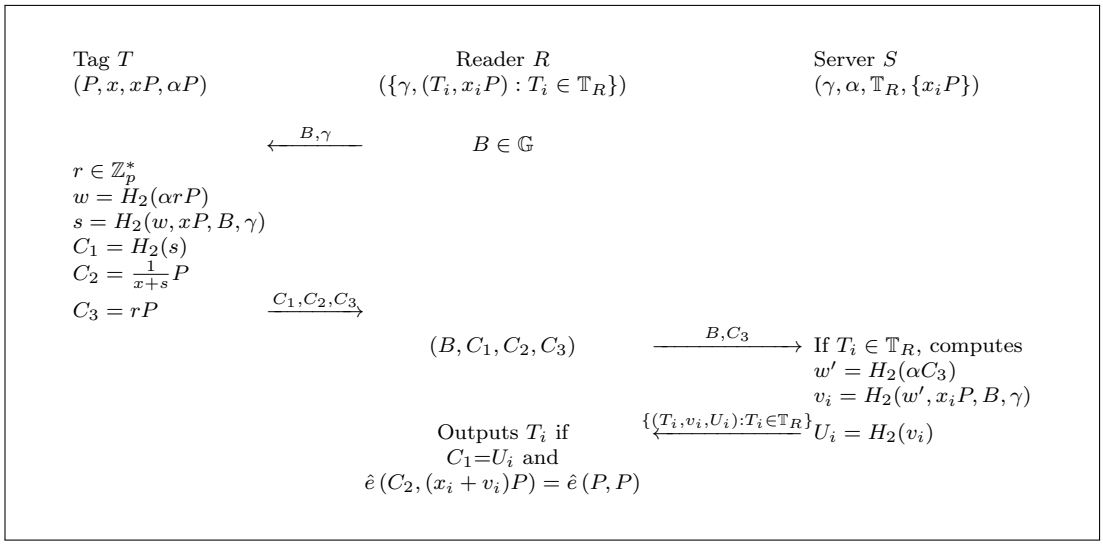

Figure 6.2: Authorized RFID authentication protocol 2.

is limited, it is desired to reduce the size of the set. Hence, we introduce a third protocol which only transfers one group element from the server to the reader. Additionally, we consider a new attack that the server may cheat a reader by replying a random value which is called *false output*. Then, the reader could not successfully authenticate a tag even the tag is valid. This attack cannot be detected in neither protocol 1 nor protocol 2. Fortunately, our protocol 3 below shows that it is able to determine whether the received value is a false output. The protocol is depicted as in Figure 6.3

- ServerKeyGen: The server picks α, where $\alpha \in \mathbb{Z}_p^*$, and sets the public key $PK = \alpha Q$ and the private key $SK = \alpha$.

- TagKeyGen: The member randomly chooses $x \in \mathbb{Z}_p^*$ and computes the tag's public and private keys $(pk, sk) = (xQ, xP)$. (P, sk, PK) are stored in the tag and pk is given to the server.

- ReaderAuth: To authorize the reader R to identify a specified set of tags $\mathbb{T}_R$, the server randomly chooses a secret $\gamma \in \mathbb{Z}_p^*$, and computes γP. For each tag $T_i \in \mathbb{T}_R$, the server computes $\alpha\gamma x_i Q$ and sets the reader's period key $rpk = \{\gamma P, (T_i, \alpha x_i \gamma Q) : T_i \in \mathbb{T}_R\}$ and the secret $rsk = (\gamma, \alpha)$. The server stores $(R, rsk, \mathbb{T}_R)$ and sends rpk to the reader R.

- Auth: To authenticate a tag, the following steps are implemented.

  1. The reader randomly selects $B \in \mathbb{G}$ and sends $(B, \gamma P)$ to the tag.
  2. Upon receiving $(B, \gamma P)$ from the reader, the tag chooses two random numbers $r, s \in \mathbb{Z}_p^*$, and computes a tuple $(C_1, C_2, C_3, C_4, C_5)$. It sends the tuple to the reader as a response.

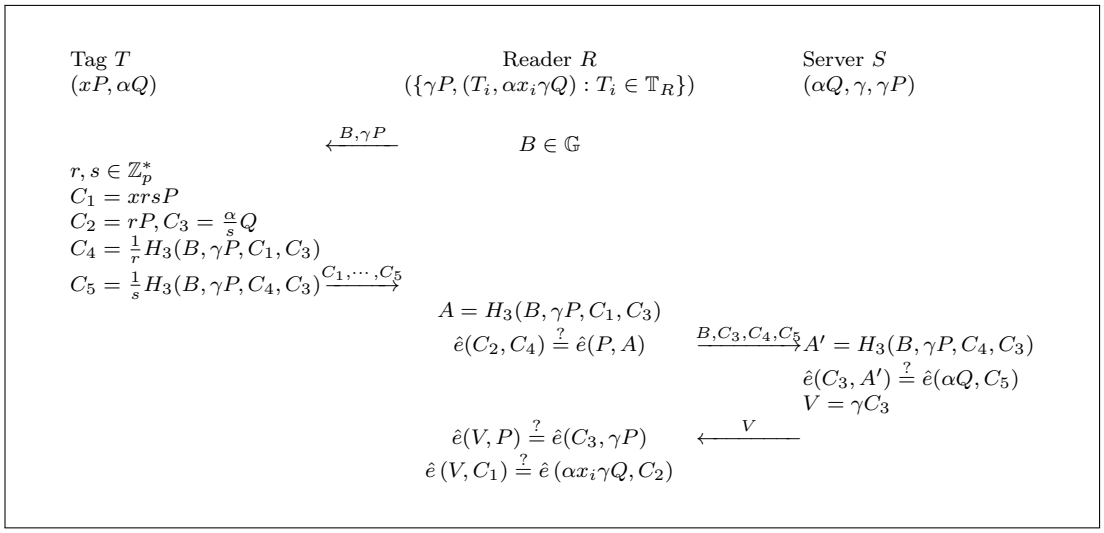

Figure 6.3: Authorized RFID authentication protocol 3.

3. Upon receiving the response, the reader checks $\hat{e}(C_2, C_4) \stackrel{?}{=} \hat{e}(P, A)$. If it holds, the reader forwards (B, C_3, C_4, C_5) to the server.
4. Upon receiving the message (B, C_3, C_4, C_5) from the reader R, the server retrieves $(\gamma, \gamma P)$ and check $\hat{e}(C_3, A') \stackrel{?}{=} \hat{e}(\alpha Q, C_5)$. If it holds, the server calculates and sends $V = \gamma C_3$ to the reader.
5. Finally, the reader authenticate the tag according to the server's response. Firstly, the checks the equation $\hat{e}(V, P) \stackrel{?}{=} \hat{e}(C_3, \gamma P)$. If the equation does not hold, the reader outputs false. After that, the reader computes $\hat{e}(V, C_1)$ and checks whether there exists a pair $(T_i, \alpha x_i \gamma Q) \in rpk$, such that $\hat{e}\,(V, C_1) = \hat{e}\,(\alpha x_i \gamma Q, C_2)$. The reader outputs T_i if the above equation holds, otherwise rejects.

6.3.4 Efficiency

We compare the efficiency in Table 6.2. In Protocol 2, C_1 is used for the reader to quickly identify the tag, and C_2 is for identity verification. The main computational cost of reader is dominated by computing $v_i P$ and $\hat{e}(C_2, x_i P + v_i P)$ for the verification, where $\hat{e}(P, P)$ can be pre-computed. This protocol requires the server to compute all potential hash values for the reader. The communication cost therefore is all v_i for each tag in $\mathbb{T}_R$. The server can send all hash values in sequence to eliminate sending T_i. In protocol 3, the communication cost and the computational cost of server is constant-size and independent of the size of $\mathbb{T}_R$. The price to pay of this protocol is a liner computation cost on the reader. The reader needs to identify the potential tag one by one until the correct one is found. The computation time therefore is linear in n for $|\mathbb{T}_R| = n$.

Note that Protocol 2 is suitable for computationally weak readers without bandwidth limitation, while Protocol 3 fits for scenarios of limited bandwidth.

Protocols	Reader Computation Cost	Communication Cost	Server Computation Cost
Protocol 2	$\mathbb{G}+\hat{e}$	$2n\|\mathbb{Z}_p\|$	$\mathbb{G}+2nH_2$
Protocol 3	$(n+1)\hat{e}$	$\|\mathbb{G}\|$	$\mathbb{G}+\hat{e}$

Table 6.2: Efficiency comparison between Protocol 2 and Protocol 3.

6.4 Privacy and Security Models

In this section, we consider the privacy and security models of authorized authentication protocols. We assume that the communication channel between the reader and the server is secure.

6.4.1 Adversaries and Oracles

We define a set of oracles and four attacks which respectively aim at different goals. In the particular attack, the ability of an adversary is regarded as the actions executed by oracle calls.

Definition 6.1 (Oracles)**.** *The adversary plays with the challenger by given public information of the system and the following oracles.*

- $\mathsf{TagCorrupt}(T) \rightarrow sk$*: On input a tag* T*, it outputs the tag's private key* sk*.*
- $\mathsf{ReaderAuth}(\mathbb{T}_R) \rightarrow rpk$*: On input a set of tags* $\mathbb{T}_R$*, it outputs the reader's period key* rpk*.*
- $\mathsf{SendTag}(T, m, \pi) \rightarrow m'$*: On input a tag* T*, a message* m *and a session* π*, it sends the message* m *to the tag and receives the tag's response* m'*.*
- $\mathsf{SendServer}(m) \rightarrow m'$*: On input a message* m*, it sends the message* m *to the server and receives the response* m'*.*
- $\mathsf{Challenge}(m^*, T^*) \rightarrow C^*$*: On input a message* m^* *and a target tag* T^* *which is not issued to* ReaderAuth *oracle, it flips a coin* b *and outputs a response* C^* *regarding to the tag* T^* *(if* $b = 1$*) or a random tag* $T^{*\prime}$ *(if* $b = 0$*). This oracle can be called at most once of a game.*

Definition 6.2 (Strong and weak adversaries)**.** *We define four types of attacks as follows.*

- Forward *attack: The adversary plays as a malicious reader who attempts to trace the tags' previous communications after it has been authorized by the server.*
- Backward *attack: The adversary plays as a malicious reader who attempts to trace the tags' future communications after it has been revoked by the server.*
- Outside *attack: The adversary plays as a dishonest server who attempts to discover the tag which is authenticating by the reader.*
- Impersonation *attack: The adversary plays as an impersonator who is not the tag holder attempts to impersonate the tag which is not compromised without being detected.*

A strong adversary can access all above oracles and launch all above attacks while a weak adversary cannot access the TagCorrypt$(\cdot)$ *oracle.*

6.4.2 Privacy and Security Models

Forward Privacy. The forward privacy game allows the adversary $\mathcal{A}$ to launch the forward attack. In the ARA system, a reader R may be authorized to authenticate a tag T in a certain period P of time. However, R shall not be able to interpret T's sessions prior to P since R is unauthorized to authenticate T outside the time P. In the forward privacy game, $\mathcal{A}$ is given the reader's current period key and attempts to decide whether the tag which can be authenticated currently was involved in the *previous* interactions.

The forward privacy game is defined in two phases, which are Forward Phase and Backward Phase. $\mathcal{A}$ plays with the challenger as follows.

- Setup: The challenger runs the algorithms ServerKeyGen and TagKeyGen to generate the server and tags' public/private keys (PK, SK) and $\{(pk_i, sk_i)\}$, respectively. The challenger gives public keys to $\mathcal{A}$.
- Forward Phase: The challenger sets the reader's period key and $\mathcal{A}$ can query Challenge$(\cdot)$ for the challenge. $\mathcal{A}$ interacts with the challenger through the oracles which can be accessed by the classified type of $\mathcal{A}$.
- Backward Phase: The challenger refreshes the reader's period key and $\mathcal{A}$ interacts with the challenger through the oracles which can be accessed by the classified type of $\mathcal{A}$.
- Guess: $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$.

Definition 6.3. *An authorized authentication scheme provides forward privacy if there is no $\mathcal{A}$ who wins the above game with the probability* $\Pr[b' = b] \geq \frac{1}{2} + \epsilon$*, where ϵ is negligible in k.*

Backward Privacy. The backward privacy game allows the adversary $\mathcal{A}$ to launch the backward attack. It is different from the forward attack that a reader R attempts to trace the tag after R has been revoked. In the backward privacy game, $\mathcal{A}$ is given the reader's current period key to authenticate the tags, while $\mathcal{A}$ needs to decide whether a tag involves in the *future* interactions after the reader was revoked.

The backward privacy game is defined in two phases, which are Forward Phase and Backward Phase. $\mathcal{A}$ plays with the challenger as follows.

- Setup: The challenger runs the algorithms ServerKeyGen and TagKeyGen to generate the server and tags' public/private keys (PK, SK) and $\{(pk_i, sk_i)\}$, respectively. The challenger gives public keys to $\mathcal{A}$.
- Forward Phase: The challenger sets the reader's period key and $\mathcal{A}$ interacts with the challenger through the oracles which can be accessed by the classified type of $\mathcal{A}$.
- Backward Phase: The challenger refreshes the reader's period key and $\mathcal{A}$ can query Challenge$(\cdot)$ for the challenge. $\mathcal{A}$ interacts with the challenger through the oracles which can be accessed by the classified type of $\mathcal{A}$.
- Guess: $\mathcal{A}_{\mathcal{F}}$ outputs a bit b' and wins the game if $b' = b$.

Definition 6.4. *An authorized authentication scheme provides backward unlinkability if there is no $\mathcal{A}$ who wins the above game with the probability* $\Pr[b' = b] \geq \frac{1}{2} + \epsilon$, *where ϵ is negligible in k.*

Reader Privacy. The reader privacy game allows the adversary $\mathcal{A}$ to launch the outside attack. Conventionally, the reader and the server are mutually trusted in RFID systems. However, the reader's privacy is needed to be considered in ARA protocols. For instance, the server may intend to learn the identity of the tag which is authenticating by the reader. Since the reader and the server are operated by different parties, the reader/tag interaction should be invisible to the server. In the reader privacy game, $\mathcal{A}$ is given the secret of the server and attempts to distinguish the tags during the server/reader interactions. $\mathcal{A}$ interacts with the challenger as follows.

- Setup: The challenger runs the algorithms ServerKeyGen, TagKeyGen and ReaderAuth to respectively generate the server's public and private keys (PK, SK), tag's public and private keys (pk, sk) and reader's keys (rpk, rsk). The challenger gives the server and tags' public/private keys and the reader's period key rpk to $\mathcal{A}$.
- Query: The adversary is allowed to make queries to the oracle SendServer$(\cdot)$.

- Challenge: The adversary outputs two tags T_0 and T_1 to the challenger. The challenger randomly chooses a bit $b \in \{0, 1\}$. Let M_t be the output of SendTag$(\cdot)$ with respect to the tag T_b and M_s be the corresponding query to SendServer$(\cdot)$. The challenger sends M_s to the adversary.

- Guess: $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$.

Definition 6.5. *An authorized authentication scheme provides reader privacy if there is no $\mathcal{A}$ who wins the above game with the probability* $\Pr[b' = b] \geq \frac{1}{2} + \epsilon$*, where ϵ is negligible in k. We say that it unconditionally preserves the reader privacy if $\epsilon = 0$.*

Tag Unforgeability. The tag unforgeability is with respect to the security of the protocol and the attacker is referred to a malicious sever. This game allows an adversary $\mathcal{A}$ to launch the impersonation attack. Clearly, it is hard to prevent the impersonation attack if the tag is corrupted. Symmetry-key based protocols are not secure against this attack as a server obtains secret keys of tags during the system setup. Hence, TagCorrupt$(\cdot)$ oracle cannot be queried during the game. $\mathcal{A}$ attempts to forge a tag's response to pass the authentication. It allows $\mathcal{A}$ to access the secret of the server and the reader. $\mathcal{A}$ interacts with the challenger as follows.

- Setup: The challenger runs the algorithms ServerKeyGen, TagKeyGen and ReaderAuth to respectively generate the server's public and private keys (PK, SK), tag's public and private keys (pk, sk) and reader's keys (rpk, rsk). The challenger gives the server's private key, tags' public key and reader's keys to $\mathcal{A}$.

- Query: The adversary can query the oracle SendTag$(\cdot)$ to the challenger.

- Forgery: $\mathcal{A}$ outputs a valid session π which is not queried to the SendTag$(\cdot)$ oracle.

Definition 6.6. *An authorized authentication scheme provides tag unforgeability if there is no $\mathcal{A}$ who can outputs a valid forgery of the tag with the non-negligible advantage ϵ in k.*

6.5 Privacy and Security Analysis

6.5.1 New Complexity Assumptions

Definition 6.7 (EDBDH Assumption)**.** *Let $(P, p, \hat{e}, \mathbb{G}, \mathbb{G}_T)$ be a pairing group. Given (P, aP, bP, cP, tP), the Extended Decisional Bilinear Diffie-Hellman problem is to determine whether $tP = abcP$. We say that the EDBDH assumption holds, if no PPT algorithm $\mathcal{A}$ can solve the problem with non-negligible advantage in k.*

Definition 6.8 (V-l-wDBDHI Assumption)**.** *Let* $(P, Q, p, \hat{e}, \mathbb{G}, \mathbb{G}_T)$ *be a pairing group. Given* $(P, Q, aP, a^2P, \cdots, a^lP, aQ, a^2Q, \cdots, a^lQ, tP)$*, the Various l-weak Decisional Bilinear Diffie-Hellman Inversion problem is to determine whether* $tP = a^{2l+1}P$*. We say that the V-l-wDBDHI assumption holds, if no PPT algorithm* $\mathcal{A}$ *can solve the problem with non-negligible advantage in* k*.*

We show that the security of EDBDH assumption is related to the security of Decisional Bilinear Diffie-Hellman (DBDH) assumption.

Lemma 6.1. *The EDBDH assumption holds if the DBDH assumption holds.*

Proof. Suppose that there is a PPT algorithm $\mathcal{A}$ who can break the EDBDH assumption. Given an instance (P, aP, bP, cP, tP), $\mathcal{A}$ can output whether $tP = abcP$ in polynomial time with non-negligible advantage. It implies that $\mathcal{A}$ decides whether $\hat{e}(P, tP) = \hat{e}(P, abcP)$ which is a solution of DBDH problem. Therefore, if DBDH problem is intractable then the EDBDH assumption holds. □

To analyse the security of V-l-wDBDHI assumption, we show that its security is similar to the security of l-weak Decisional Bilinear Deffie-Hellman Inversion (l-wDBDHI) assumption. According Lemma 6.1, a solution of V-l-wDBDHI problem also implies that the algorithm $\mathcal{A}$ can decide whether $\hat{e}(P, tP) = \hat{e}(P, a^{2l+1}P)$. Since that l-wDBDHI is proved secure in the generic group model, the complexity of V-l-wDBDHI can be bounded by using the similar proof of general Diffie-Hellman Exponent problem [BBG05].

6.5.2 Privacy Proofs

Theorem 6.1. *Our ARA protocol 1 provides forward privacy and backward privacy against the weak adversary if* H_1 *is pre-image resistant.*

Proof. Suppose a weak adversary can distinguish the tags from their previous (resp. future) interactions. Given an instance (a, b, γ, C), the adversary aims to determine whether it associates with the tag T_0 or T_1. Due to the input a, b, γ are known and $C = H_1(x_b, a, b, \gamma)$, where $b = 0$ or $b = 1$, the adversary has to distinguish the tag's secret keys x_0 and x_1 which are unknown to adversary. This implies that the adversary is able to compute x_b ($b = 0$ or $b = 1$) since the weak adversary cannot query the oracle $\mathsf{TagCorrupt}(\cdot)$. It contradicts to the pre-image assumption of the hash function H_1. Therefore, the proposed protocol 1 preserves forward privacy (resp. backward privacy) against the weak adversary. □

Theorem 6.2. *Our ARA protocol 2 provides forward privacy against the strong adversary if the ODH assumption holds.*

Proof. Suppose a strong adversary $\mathcal{A}$ who can break the forward privacy of our protocol 2. We can construct an algorithm $\mathcal{B}$ to use $\mathcal{A}$ to violate the ODH assumption. $\mathcal{B}$ is given an instance (aP, Pb, t), a hash function H_2 and an oracle $\mathcal{O}$, the goal is to output a decision whether $t = H_2(abP)$. $\mathcal{B}$ sets up the forward privacy game and interacts with $\mathcal{A}$ as follows.

Setup: Let P be a generator of group $\mathbb{G}$ and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^l$ be a cryptographic hash function. $\mathcal{B}$ sets the servers public and private key pair (PK, SK) as (bP, b), where b is unknown to $\mathcal{B}$. $\mathcal{B}$ picks $x_i \in \mathbb{Z}_p^*$, where $i = 1, \cdots, q$, and sets the tag T_i's public and private keys (pk, sk) as (x_iP, x_i). Then, for each tag, $\mathcal{B}$ stores (x_i, P, x_iP, bP). $\mathcal{B}$ maintains the list $L_T = \{< T, sk, pk >\}$ and adds all tags $< T_i, x_iP, x_i >$ into L_T.
Forward Phase: $\mathcal{B}$ randomly chooses $\gamma \in \mathbb{Z}_p^*$ and provides the following oracles to $\mathcal{A}$.

- TagCorrupt: $\mathcal{A}$ issues a tag corruption query on input a tag T_i. If T_i exists in the list L_T, $\mathcal{B}$ outputs x_iP, x_i, otherwise $\mathcal{B}$ outputs $\perp$.

- ReaderAuth: $\mathcal{A}$ issues a reader registration query on input a set of tags $\mathbb{T}_R$. If any $T_i \in \mathbb{T}_R$ does not appear in L_T, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ sets

$$rpk = \{\gamma, (T_i, x_iP) : T_i \in \mathbb{T}_R\}$$

  and outputs rpk.

- SendTag: $\mathcal{A}$ issues a tag T_i, a message $m_i = (B_i, \gamma_i)$, $\mathcal{B}$ retrieves $< T_i, x_iP, x_i >$ from the list L_T. $\mathcal{B}$ randomly selects $r \in \mathbb{Z}_p^*$ and computes

$$w_i = H_2(ar_iP),\ s_i = H_2(w_i, x_iP, B_i, \gamma_i),\ C_{i,1} = H_2(s_i),$$

$$C_{i,2} = \frac{1}{x_i + s_i}P,\ C_{i,3} = r_iP.$$

  $\mathcal{B}$ sets $C_i = (C_{i,1}, C_{i,2}, C_{i,3})$ and outputs C_i.

- SendServer: $\mathcal{A}$ issues a query on input a message $m_i = (B_i, C_{i,3})$. For each tag $T_i \in \mathbb{T}_R$, $\mathcal{B}$ looks up x_iP from L_T and queries the oracle $\mathcal{O}$ on input $C_{i,3}$. On receiving $\mathcal{O}$'s output $H_2(bC_{i,3})$, $\mathcal{B}$ computes

$$V_i = H_2(H_2(bC_{i,3}), x_iP, B_i, \gamma),\ U_i = H_2(V_i),$$

  and outputs $\{(T_i, V_i, U_i) : T_i \in \mathbb{T}_R\}$.

- Challenge: $\mathcal{A}$ issues a challenge query on input a message $m^* = (B^*, \gamma^*)$ and a target tag $T^* \in L_T$. $\mathcal{B}$ retrieves T^*'s private key x^* and randomly chooses

$coin \in \{0,1\}$. If $coin = 1$, $\mathcal{B}$ computes $w^* = t$, otherwise $w^* = y$, where $y \in \mathbb{Z}_p^*$. Then $\mathcal{B}$ computes

$$s^* = H_2(w^*, x^*P, B^*, \gamma^*),\ C_1^* = H_2(s^*),\ C_2^* = \frac{1}{x^* + s^*}P,\ C_3^* = aP.$$

$\mathcal{B}$ sets $C^* = (C_1^*, C_2^*, C_3^*)$ and outputs C^*.

Backward Phase: $\mathcal{B}$ randomly selects $\gamma' \in \mathbb{Z}_p^*$. It provides the following oracles to $\mathcal{A}$.

- TagCorrupt: Same as in forward phase.
- ReaderAuth: Same as in forward phase except that $\mathcal{B}$ replaces γ by γ'.
- SendTag: Same as in forward phase.
- SendServer: Same as in forward phase except that $\mathcal{B}$ replaces γ by γ'.

Guess: $\mathcal{A}$ outputs a guess $coin' \in \{0,1\}$ and wins the game if $coin' = coin$. Then, $\mathcal{B}$ concludes the game and outputs $t = H_2(abP)$ if $coin' = 1$, otherwise outputs $t \neq H_2(abP)$.

□

Theorem 6.3. *Our ARA protocol 2 provides backward privacy against the strong adversary if the ODH assumprion holds.*

Proof. The proof of the theorem is similar to the proof of Theorem 6.2 (PT2). To simplify the proof, we only show the differences from PT2. Given an ODH instance (aP, bP, t), a hash function H_2 and an oracle $\mathcal{O}$, $\mathcal{B}$ can output whether $t = H_2(abP)$ if ODH assumption is broken. We use the same Setup, Forward Phase and Backward Phase of PT2, except that Challenge is provided in Backward Phase instead of Forward Phase. Clearly, if the adversary $\mathcal{A}$ outputs a guess $coin' \in \{0,1\}$, $\mathcal{B}$ can conclude that $t = H_2(abP)$ (if $coin' = 1$) or $t \neq H_2(abP)$ (if $coin' = 0$). □

Theorem 6.4. *Our ARA protocol 2 unconditionally provides reader privacy.*

Proof. We show that our protocol 2 unconditionally preserves the reader's privacy against the adversary $\mathcal{A}$. During the challenge phase, $\mathcal{A}$ submits two tags T_0, T_1 to $\mathcal{B}$. $\mathcal{B}$ randomly picks $b \in \{0,1\}$ and runs the tag authentication to obtain the tag T_b's response (C_1, C_2, C_3). According to the protocol, we have $C_3 = rP$ and $\mathcal{B}$ gives B, C_3 to $\mathcal{A}$. Since that $B \in \mathbb{G}$, $r \in \mathbb{Z}_p^*$ are random values, it obviously gives no information about the corresponding tag T_b. $\mathcal{A}$ has no advantage to determine whether $T_b = T_0$ or $T_b = T_1$. Therefore, we have that Protocol 2 unconditionally provides the reader privacy.

□

Theorem 6.5. *Our ARA protocol 3 provides forward privacy against the strong adversary if the V-l-wDBDHI assumption holds.*

Proof. Suppose a strong adversary $\mathcal{A}$ who can break the forward privacy of our protocol 3. We can construct an algorithm $\mathcal{B}$ to use $\mathcal{A}$ to solve the V-l-wDBDHI problem. Let $l = 3$, $\mathcal{B}$ is given an instance $(P, Q, aP, a^2P, a^3P, aQ, a^2Q, a^3Q, tP)$, the goal is to output a decision whether $tP = a^7P$. $\mathcal{B}$ sets up the forward privacy game and interacts with $\mathcal{A}$ as follows.

Setup: $\mathcal{B}$ sets $P_0 = aP, Q_0 = aQ$ as two generators of group $\mathbb{G}$ and the server's public key $PK = a^2Q$, private key $SK = a$, where a is unknown to $\mathcal{B}$. $\mathcal{B}$ creates q tags for the game and randomly selects a tag T_c as a challenge tag. $\mathcal{B}$ picks $x_i \in \mathbb{Z}_p^*$, where $i = 1, 2, \cdots, q$. For each tag $T_i \neq T_c$, $\mathcal{B}$ computes the tag's public/private keys $(pk_i, sk_i) = (x_iaQ, x_iaP)$. For the tag T_c, $\mathcal{B}$ sets the public/private key pair as $(pk_c, sk_c) = (x_ca^3Q, x_ca^3P)$. Then, $\mathcal{B}$ stores (P_0, sk_i, a^2Q) into the tag T_i, where $i = 1, 2, \cdots, q$. $\mathcal{B}$ maintains four lists $L_T = \{< T, sk, pk, x >\}$, $L_{H3} = \{< B, W, U, S, z, Z >\}$ and $L_C = \{< S, s >\}$, then adds all tags $< T_i, sk_i, pk_i, x_i >$ into L_T.

Forward Phase: $\mathcal{B}$ randomly selects $k \in \mathbb{Z}_p^*$ and sets $P_0^{\gamma^*} = ka^2P$ and provides the following oracles to $\mathcal{A}$.

- H_3 Oracle: $\mathcal{A}$ issues a hash query on input a message m_i, where $m_i = (B_i, W_i, U_i, S_i)$, $\mathcal{B}$ outputs Z_i if $< B_i, W_i, S_i, z_i, Z_i >$ exits. Otherwise, $\mathcal{B}$ checks whether S_i appears in the list L_C. If it exists, $\mathcal{B}$ computes $Z_i = z_iaQ$, otherwise $Z_i = kz_ia^3Q$, where $z_i \in \mathbb{Z}_p^*$, and sets $H_3(B_i, W_i, U_i, S_i) = Z_i$. Then, $\mathcal{B}$ adds $< B_i, W_i, U_i, S_i, z_i, Z_i >$ into the list L_{H3} and outputs Z_i.

- TagCorrupt: $\mathcal{A}$ issues a tag corruption query on input a tag T_i. If T_i does not exist in the list L_T, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ retrieves $< T_i, sk_i, pk_i, x_i >$ and returns sk_i.

- ReaderAuth: $\mathcal{A}$ issues a reader authorization query on input a set of tags $\mathbb{T}_R$. If $T_c \in \mathbb{T}_R$, $\mathcal{B}$ aborts the simulation, otherwise, for each $T_i \in \mathbb{T}_R$, $\mathcal{B}$ retrieves $< T_i, sk_i, pk_i, x_i >$ and computes kx_ia^3Q. Then $\mathcal{B}$ sets

$$rpk = \{ka^2P, (T_i, kx_ia^3Q) : T_i \in \mathbb{T}_R\}$$

  and outputs rpk.

- SendTag: $\mathcal{A}$ issues a tag T_i, a message $m_i = (B_i, \gamma_iP_0)$, $\mathcal{B}$ retrieves $< T_i, sk_i, pk_i, \cdot >$ from the list L_T. $\mathcal{B}$ picks $s_i \in \mathbb{Z}_p^*$ and adds $< \frac{a^2}{s_i}Q, s_i >$ in to the list L_C. $\mathcal{B}$ ran-

domly selects $r_i \in \mathbb{Z}_p^*$ and calculates

$$C_{i,1} = r_i s_i sk_i,\ C_{i,2} = r_i aP,$$

where sk_i is in the list L_T. $\mathcal{B}$ calls H_3 oracle on input $(B_i, \gamma_i P_0, C_{i,1}, \frac{a^2}{s_i}Q)$ and obtains the output Z_i. Then, $\mathcal{B}$ issues $(B_i, \gamma_i P_0, \frac{1}{r_i}Z_i, \frac{a^2}{s_i}Q)$ to H_3 oracle and receives the output Z_i'. $\mathcal{B}$ sets

$$C_{i,3} = s_i aP,\ C_{i,4} = \frac{1}{r_i}Z_i,\ C_{i,5} = \frac{1}{s_i}Z_i',$$

$m_i' = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}, C_{i,5})$ and outputs m_i'.

- SendServer: $\mathcal{A}$ issues a query on input a message $m_i = (B_i, C_{i,3}, C_{i,4}, C_{i,5})$. If $< B_i, \gamma^* P_0, C_{i,4}, C_{i,3}, z_i, Z_i >$ exists and $\hat{e}(C_{i,3}, Z_i) = \hat{e}(a^2Q, C_{i,5})$ holds, then $\mathcal{B}$ computes $V_i = \frac{k}{s_i}a^3Q$ If $< C_{i,3}, s_i >$ appears in L_C, or computes $V_i = \frac{1}{z_i}C_{i,5}$. Then, $\mathcal{B}$ sets $m' = V_i$ and outputs m'.

- Challenge: $\mathcal{A}$ issues a challenge query on input a message $m^* = (B^*, \gamma^* P_0)$ and a target tag T^*, such that T^* is not queried to the ReaderAuth oracle and $T^* = T_c$, $\mathcal{B}$ flips a coin $b \in \{0, 1\}$. If $b = 1$, $\mathcal{B}$ chooses $u, v \in \mathbb{Z}_p^*$ and computes

$$C_1^* = x_c uvtP,\ C_2^* = ua^3P,\ C_3^* = \frac{1}{v}Q.$$

If $b = 0$, $\mathcal{B}$ computes $C_1^* = yuvaP$, where $y \in \mathbb{Z}^*$. $\mathcal{B}$ issues the H_3 oracle call on input $(B^*, \gamma^* P, C_1^*, C_3^*)$ and receives the output $< B^*, \gamma^* P_0, C_3^*, z^*, Z^* >$. $\mathcal{B}$ computes $C_4^* = \frac{z^* k}{u}aQ$ and $\mathcal{B}$ queries H_3 oracle on input $(B^*, \gamma^* P_0, C_4^*, C_3^*)$ then obtains the output $< B^*, \gamma^* P_0, C_3^*, z'^*, Z'^* >$. $\mathcal{B}$ computes $C_5^* = \frac{z'^* k}{v}aQ$ and sets $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$. Finally, $\mathcal{B}$ outputs C^*.

Backward Phase: $\mathcal{B}$ randomly selects $k' \in \mathbb{Z}_p^*$, and sets $\gamma' P_0 = k'P$. It provides the following oracles to $\mathcal{A}$.

- H_3 Oracle: $\mathcal{A}$ issues a hash query on input a message $m_i = (B_i, W_i, U_i, S_i)$, $\mathcal{B}$ outputs Z_i if $< B_i, W_i, U_i, S_i, z_i, Z_i >$ exits. $\mathcal{B}$ computes $Z_i = z_i k' aQ$, where $z_i \in \mathbb{Z}_p^*$, if S_i does not exist in the list L_C, otherwise, $\mathcal{B}$ computes $Z_i = z_i aQ$. $\mathcal{B}$ sets $H_3(B_i, W_i, U_i, S_i) = Z_i$ and outputs Z_i. It adds $< B_i, W_i, U_i, S_i, z_i, Z_i >$ into the list L_{H3} and outputs Z_i.

- TagCorrupt: Same as in forward phase.

- ReaderAuth: $\mathcal{A}$ issues a reader authorization query on input a set of tag $\mathbb{T}_R$. For each $T_i \in \mathbb{T}_R$, $\mathcal{B}$ retrieves $< T_i, sk_i, pk_i, x_i >$ and computes $Y_i = x_i k' aQ$ if $T_i \neq T_c$,

otherwise computes $Y_i = x_c k' a^3 Q$. Then $\mathcal{B}$ sets $rpk' = \{k'P, (T_i, Y_i) : T_i \in \mathbb{T}_R\}$ and outputs rpk'.

- SendServer: $\mathcal{A}$ issues a query on input a message $m_i = (B_i, C_{i,3}, C_{i,4}, C_{i,5})$. If $< B_i, W_I, C_{i,4}, C_{i,3}, z_i, Z_i >$ exists and $\hat{e}(C_{i,3}, Z_i) = \hat{e}(a^2 Q, C_{i,5})$ holds, then $\mathcal{B}$ computes $V_i = \frac{k'}{s_i} Q$ If $< C_{i,3}, s_i >$ appears in L_C, or computes $V_i = \frac{1}{z_i} C_{i,5}$. Then, $\mathcal{B}$ sets $m' = V_i$ and outputs m'.

- SendTag: Same as in forward phase.

Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins the game if b'=b. $\mathcal{B}$ concludes the game and outputs $tP = a^7 P$ if $b' = 1$, otherwise outputs $tP \neq a^7 P$.

□

Theorem 6.6. *Our ARA protocol 3 provides backward privacy if the V-l-wDBDHI assumption holds.*

Proof. The proof of the theorem is similar to the proof of Theorem 6.5 (PT5). To simplify the proof, we only show the differences from PT5. Suppose a strong adversary $\mathcal{A}$ who can break the backward privacy of our protocol. We can construct an algorithm $\mathcal{B}$ to use $\mathcal{A}$ to violate the V-l-wDBDHI assumption. Given an instance of V-l-wDBDHI and let $l = 3$, we use the same Setup of PT5. However, we swap the simulation of Forward Phase and Backward Phase of PT5. The oracle Challenge$(\cdot)$ is provided in the Backward Phase. Then, based on the output $b' \in \{0, 1\}$ of $\mathcal{A}$, $\mathcal{B}$ can conclude the game and output $tP = a^7 P$ if $b' = 1$ or $tP \neq a^7 P$.

□

Theorem 6.7. *Our ARA protocol 3 provides reader privacy if the EDBDH assumption holds.*

Proof. Suppose that a strong adversary $\mathcal{A}$ who can break the reader privacy of our protocol 3. We can construct an algorithm $\mathcal{B}$ to use $\mathcal{A}$ to solve the EDBDH problem. $\mathcal{B}$ is given an instance (P, aP, bP, cP, tP), it aims to determine whether $tP \stackrel{?}{=} abcP$. $\mathcal{B}$ sets up the game and interacts with $\mathcal{A}$ as follows.
Setup: $\mathcal{B}$ sets $P_1 = aP, Q = cP$ as two generators of the group $\mathbb{G}$. $\mathcal{B}$ creates q tags for the game and selects a challenge tag T_c. $\mathcal{B}$ randomly chooses $x_1, \cdots, x_q, \alpha, \gamma \in \mathbb{Z}_p^*$ and computes $x_i Q, x_i P_1, \alpha Q, \gamma P_1$ and $x_i \alpha \gamma Q$. $\mathcal{B}$ sets the server's public/private keys $(PK, SK) = (\alpha Q, \alpha)$, tag T_i's public/private keys $(pk, sk) = (x_i Q, x_i P_1)$ and the reader's period keys

$$(rpk, rsk) = (\{P_1, \gamma P_1, T_i, x_i \alpha \gamma Q : i = 1, \cdots, q\}, (\gamma, \alpha)).$$

$\mathcal{B}$ gives generated public and private keys to the adversary $\mathcal{A}$. $\mathcal{B}$ maintains a filled list $L_T = \{< T, sk, pk, x >\}$ and $L_{H3} = \{< B, W, U, S, z, Z >\}$, which is initially empty.

H_3 Oracle: $\mathcal{A}$ issues a H_3 hash query on input a message $m_i = (B_i, W_i, U_i, S_i)$, if Z_i appears in the list L_{H3}, $\mathcal{B}$ outputs Z_i. For the query from Challenge oracle, $\mathcal{B}$ chooses $z_i \in \mathbb{Z}_p^*$ and computes $Z_i = z_i cP$ if U_i exists in L_{H_3}, otherwise computes $Z_i = z_i bP$. For other queries, $\mathcal{B}$ selects $z_i \in \mathbb{Z}$ and computes $Z_i = z_i P$. $\mathcal{B}$ sets $H(B_i, W_i, U_i, S_i) = Z_i$ and adds $< B_i, W_i, U_i, S_i, z_i, Z_i >$ into the list L_{H3}. $\mathcal{B}$ outputs Z_i.

Challenge: $\mathcal{A}$ issues a challenge query on tags T_0, T_1. $\mathcal{B}$ randomly chooses $coin \in \{0, 1\}$ and retrieves x_{coin} for the tag T_{coin}. $\mathcal{B}$ selects $B \in \mathbb{G}$, $u \in \mathbb{Z}_p^*$ and sets $r = b$, $s = cu$. $\mathcal{B}$ computes

$$C_1^* = tux_{coin}P,\ C_3^* = \frac{\alpha}{u}P.$$

Then, $\mathcal{B}$ calls H_3 query on input $(B, \gamma P_1, C_1^*, C_3^*)$ to obtain the output Z_i and calculates $C_4^* = z_i P$, where $z_i \in L_{H3}$. After that, $\mathcal{B}$ calls H_3 query on input $(B, \gamma P_1, C_4^*, C_3^*)$ to obtain the output Z_i' and computes $C_5^* = z_i' P$. Finally, $\mathcal{B}$ outputs (B, C_3^*, C_4^*, C_5^*) to $\mathcal{A}$.

Guess: $\mathcal{A}$ outputs a guess $coin' \in \{0, 1\}$ and wins the game if $coin' = coin$. Then, $\mathcal{B}$ can conclude the game and outputs $tP = abcP$ if $coin' = 1$, otherwise output $tP \neq abcP$.

□

6.5.3 Security Proofs

Theorem 6.8. *Our ARA protocol 2 provides tag unforgeability if BB signature [BB04] is secure.*

Proof. In ARA protocol 2, a reader challenges the tag on a message $m = (B, \gamma)$. The response of the tag consists of three values (C_1, C_2, C_3). We consider that $C_2 = \frac{1}{x+s}P$, where $s = H_2(H_2(\alpha r P), xP, B, \gamma)$, as a BB signature of the signing message s under the private key x. Although s is not sent to the reader, an adversary $\mathcal{A}$ has to forge a BB signature to make a valid response. The security of ARA protocol 2 depends on the security of BB signature and the latter is provably unforgeable if the private key x is unknown. Hence, the protocol 2 provides tag unforgeability.

□

Theorem 6.9. *Our ARA protocol 3 provides the tag unforgeability if the k+1-Exponent assumption holds.*

Proof. Suppose an adversary $\mathcal{A}$ who can break the tag unforgeability of our protocol 3. We can construct an algorithm $\mathcal{B}$ to use $\mathcal{A}$ to solve the k+1-Exponent problem.

$\mathcal{B}$ is given an instance (P, aP, a^2P, a^3P) where $k = 3$, it aims to find a^4P. $\mathcal{B}$ sets up the game and interacts with $\mathcal{A}$ as follows.

Setup: $\mathcal{B}$ sets $P_1 = a^2P$ and $Q_1 = aP$ as two generators of group $\mathbb{G}$, where a is unknown to $\mathcal{B}$. $\mathcal{B}$ randomly picks $\alpha, \gamma \in \mathbb{Z}_p^*$ and computes $\alpha Q_1, \gamma P_1$. Let the tag T's public key be $XQ_1 = a^3P$, where $x = a^2$ is unknown to $\mathcal{B}$. Then, $\mathcal{B}$ computes $\alpha\gamma a^3P$, the server's public/private key pair $(PK, SK) = (\alpha Q_1, \alpha)$, reader's period key pair $(rpk, rsk) = (\{\gamma P_1, T, \alpha\gamma a^3P\}, (\gamma, \alpha))$, and gives (PK, SK, rpk, rsk, xQ_1) to $\mathcal{A}$. $\mathcal{B}$ maintains a list $L_{H3} = \{< B, W, U, S, z, Z >\}$, which is initially empty.

- H_3 Oracle: $\mathcal{A}$ issues a hash query on input a message $m_i = (B_i, W_i, U_i, S_i)$, $\mathcal{B}$ outputs Z_i if $< B_i, W_i, S_i, z_i, Z_i >$ exits. $\mathcal{B}$ selects $z_i \in \mathbb{Z}_p^*$ and computes $Z_i = z_iP$ if the query is submitted by the SendTag oracle, otherwise $\mathcal{B}$ computes $Z_i = z_iU_i$. $\mathcal{B}$ sets $H(B_i, W_i, U_i, S_i) = Z_i$ and outputs Z_i. Then, $\mathcal{B}$ adds $< B_i, W_i, U_i, S_i, z_i, Z_i >$ into the list L_{H3}.

- SendTag: $\mathcal{A}$ issues a message $m_i = (B_i, \gamma_iP)$, $\mathcal{B}$ randomly chooses $r_i, s_i \in \mathbb{Z}_p^*$ and computes $C_{i,1} = r_iP_1, S_i = s_i\alpha a^3P$. $\mathcal{B}$ submits a query $(B_i, \gamma_iP, C_{i,1}, S_i)$ to the H_3 oracle and obtains the output $< B_i, W_i, U_i, S_i, z_i, Z_i >$. Then, $\mathcal{B}$ queries H_3 oracle on input $(B_i, \gamma_iP, \frac{z_i}{r_i,s_i}P, S_i)$ to H_3 oracle and receives the output $< B_i, W_i, U_i, S_i, z_i', Z_i' >$. $\mathcal{B}$ sets C_i as

$$C_{i,1} = r_ia^2P,\ C_{i,2} = r_is_ia^2P,\ C_{i,3} = s_i\alpha a^3P,\ C_{i,4} = \frac{z_i}{r_is_i}P,\ C_{i,5} = s_iz_i'a^2P.$$

  Finally, $\mathcal{B}$ outputs C_i.

Forgery: Eventually, $\mathcal{A}$ outputs a forgery $C* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ on message $m^* = (B^*, \gamma^*P)$ of the tag T. $\mathcal{A}$ wins the game if the pair (m^*, C^*) is valid and the algorithm Auth accepts the tag T, where (m^*, C^*) is not an output of SendTag query. $\mathcal{B}$ retrieves z_i, z_j from the list L_{H3}, where $z_iP = H_3(B^*, \gamma^*P, C_1^*, C_3^*)$ and $z_jP = H_3(B^*, \gamma^*P, C_4^*, C_3^*)$ from the list L_{H3}. Then, $\mathcal{B}$ computes

$$a^4P = \frac{1}{z_iz_j}C_5^*$$

as the solution of the k+1-Exponent problem.

□

6.6 Conclusion

In this chapter, we introduced a novel concept of authorized RFID authentication protocols. The reader's privacy is considered as a new issue that it prevents the

server disclosing the identity of the tag which is authenticated by the reader. Two protocols were proposed based on the different efficiency requirements. We provided the formal definition of privacy and security models of authorized authentication protocols and proved that our protocols are secure against the various adversaries.

Chapter 7

Anonymous Yoking-Group Proofs

Yoking-proofs show an interesting application in RFID that a verifier can check whether two tags are simultaneously scanned by a reader. We consider a scenario in which multi-group of tags can be proved to be scanned simultaneously. Grouping-proof, which is an extension of yoking-proofs, allows multiple tags to be proved together, while existing protocols *cannot* support multiple groups. In this chapter, we introduce a novel concept called "yoking-group proofs", which was proposed in [LMSV15a]. Additionally, we propose an anonymous yoking-proof protocol and an anonymous yoking-group proof protocol and prove their security in Universal Composability framework.

7.1 Introduction

A conventional RFID system usually assumes that a reader is authorized and the the server is online. In some specific application, both the readers and the server are also operated by a trusted party who initiates the tags. Moreover, a usual RFID tag authentication can only check the validity of a single tag. These limitations restrict the adoption of RFID in many applications. Consider that we need to prove the simultaneous presence of tags, it is insufficient if the tags are authenticated individually. Furthermore, if this task is delegated to an untrusted third party, preventing tags' privacy and security become a challenging task.

Juels [Jue04] introduced the notion of yoking-proofs which guarantee that two tags have been scanned simultaneously. In the scenario, a verifier who maintains a server and all secret information of the tags is considered as a TTP. Yoking-proofs allow an untrusted party to control a reader which can connect to the verifier after a proof is compiled. Two tags interact with the reader individually and assume that the tags cannot be directly linked during the protocol run. The reader obtains a proof at the end of the protocol and issues it to the verifier who can be either offline or online.

The concept of yoking-proofs have been extended to grouping-proofs by Saito and Sakurai [SS05] in 2005. Grouping-proof protocols allow multiple tags cooperatively generate a proof of simultaneous presence to the reader. During the proof, any message from a tag shall be delivered by a reader to another tag. An important task of a reader is to follow a specific logical structure to communicate with tags as different structures may significantly influence the security of protocol.

Tag anonymity is an essential consideration of most RFID authentication protocols [SM08, BM11, LWD08, LBSV10a, LBSV10b]. In grouping-proof protocols, preventing the tag or group's identity from leaking to an adversary is a challenging task if the protocol intends to employ symmetric-key cryptography. Usually, a symmetric-key based RFID authentication protocol requires exhaustive key search during the tag authentication. To check the validity of multiple tags without providing identities, efficiency problems will ensue as there are many possible combinations of tags and each tag cannot be considered individually. Public key based protocols [BLS+10, BLS+12, HP12] enable the verifier to verify arbitrary anonymous groups without overwhelming computational cost. However, low-cost passive tags usually cannot afford these protocols due to the extensive computation requirements and therefore, we do not consider this type of solutions as it is rather expensive and impractical.

Motivation. Although the grouping-proof can show the simultaneous presence of a group of tags, in some scenarios, single grouping-proof is insufficient. For example, a warehouse which stores a number of vehicles intends to prove the presence of all goods to the owner. For each car, several tags are attached to core components, such as the engine, brake and lock system. A prover should show the presence of cars as well as their integrity. Swapping components between vehicles is prevented, otherwise the product is no longer original. Unfortunately, the grouping-proof can only show the presence of either all components or an individual car. Note that each car is in a different group since it intends to be sold separately. In addition, a car should be untraceable after it has been sold. Thus, the authentication should be anonymous that an unauthorized third party cannot obtain the tags' identity during the protocol execution.

For another instance, a hospital employs an RFID system to manage medications for inpatients. A medicine could be prescribed to different patients with various instructions. It is important to keep the safety of medicines. Meanwhile, a nurse needs to store a group of medicines (for an individual person) in a separate place, such as a locker. However, both yoking-proof and grouping-proof provide only the presence of a pair or a single group of tags. The information is insufficient to monitor the storage for all patients.

In above scenarios, a protocol which provides untraceability and simultaneous presence of multi-group is necessary. In this chapter, we propose a solution that it not only supports a single group verification but also generates an anonymous proof for multiple groups. Note that a trivial solution from grouping-proof protocols is highly unlikely due to the lack of inter-group communication. In the literature, grouping-proof protocols usually interact with the tags in a specific order which constructs a logical structure. To the best of our knowledge, there are two main

structures: ring and yoking-style. Ring-based protocols [BR05, BdMM08, DK09, LLM+09, BM13] start from an initiator tag, then followed by the other tags and complete the proof at the initiator. Regardless whether tags share a group key or not, once an initiator has generated a proof, it cannot interact with initiators of other groups. In other words, we need to significantly modify the protocol to achieve the requirement. Existing protocols [SS05, LLMC08, CYH+08, HK09] which adopt yoking-style proof generation can also hardly accomplish a multi-group proof. These protocols usually employ a pallet tag to collect responses from a bunch of tags, then generate a proof by collaborating with another tag. Eventually, the group proof is formed similarly as in yoking proof. Therefore, to perform the multi-group proof, an upper layer must be constructed.

Related Work. Two yoking-proofs were proposed in [Jue04], while the protocol using minimalist MACs has been attacked by Saito and Sakurai [SS05]. Later on, a tree-based protocol was introduced by Chien and Liu [CL09] that the computational cost of identifying a tag is constant. However, the protocols requires that a reader holds some shared secrets. Indeed, the protocol violates the rule of yoking-proofs where a reader is assumed to be untrusted and cannot handle any secret. The concept of anonymous grouping-proof was firstly presented by Bolotnyy and Robins [BR05]. Unfortunately, the protocol statement is ambiguous that how to link tags via their pseudonyms.

Huang and Ku [HK09] introduced a lightweight grouping-proof which employs a pseudorandom number generator in passive tags, however, the proposal is insecure as described in [CYWL11]. Burmester, Medeiros and Motta [BdMM08] presented two grouping-proof protocols which support the anonymity and forward security. The protocols have been proved secure in universal composability framework, while some flaws were illustrated later by Peris-Lopez, Orfila, Hernandez-Castro and Lubbe [PLOHCvdL11]. Subsequently, some other anonymous grouping-proof protocols were proposed in [PHER07, LNZ+13, BLS+10].

Recently, Burmester and Munilla [BM13] proposed an anonymous grouping-proof protocol which prevents an adversary from linking two sessions of a target group. The protocol considered that a group of tags is defined as in a logical ring. A specific framework was described. It allows a reader to concurrently interact with tags. The proof of simultaneous presence is generated by a tag labeled with index 1 and the proof is in constant size.

In grouping-proof, a reader is normally assumed as untrusted and the verifier is offline. If an online server is available, the grouping-proof is not hard since the verifier can setup a timer for a challenge. Even so, there are some protocols [LLMC08, DK09, LLM+09] proposed for an online verifier. Note that, we only consider a system where online verifier is unavailable.

Organization of This Chapter. The rest of this chapter is organized as follows. In Section 7.2, we describe an overview of our anonymous yoking-group proof system and define the notations of symbols. Section 7.3 describes the security and privacy models with some attacks. Section 7.4 proposes an anonymous yoking-proof. Section 7.5 gives the concrete constructions of our anonymous yoking-group proof protocol. Section 7.6 proves the privacy and the security of proposed protocols. Finally, we conclude the chapter in Section 7.7.

7.2 Preliminaries

In this section, we describe the overview of our anonymous yoking-group proofs and the notions which are used throughout this chapter.

7.2.1 System Overview

The yoking-group proof system consists of three entities: RFID tags, reader and a verifier.

- Tag T_i: A low-cost device which has a small storage and limited computational capabilities. In this chapter, we only consider (passive) tags which cannot afford the cost of public key cryptography. To construct a group, each tag is assigned distinct keys and a unique sequential index starts from 1. We assume that the keys of each tag refer to an unique identity and a tag is initiated by the verifier.

- Reader R_i: A powerful device which is controlled by an untrusted third party. Tag and verifier's secret information is unknown to R_i. The reader can maintain links, deliver messages between tags and provide to the verifier a yoking-group proof according to the collections.

- Verifier V_i: A TTP which maintains all the keys and identities of groups. If V_i is not the owner of tags, it needs to share the same secret of tags with the owner and V_i can initiate the system. Note that V_i is normally an offline party and a yoking-group proof is verifiable at some time later.

In an anonymous yoking-group proof, a verifier V creates a group of tags where each tag shares a part of secret keys with other tags. Each group has an individual group identification key which indicates the group's identity. Note that no individual tag holds the key of a group, instead, the key is reconstructed during the protocol run. In the protocol execution, a reader firstly interrogates all the tags and collects their indexes. Then, the reader constructs a logical key structure based on the tag indexes and interacts with tags in the specific order. Finally, the reader obtains a

$\mathcal{K}$:	a space of bit strings where each element is k bits.
$H(\cdot)$:	a cryptographic collision-resistant hash function that $H : \{0,1\}^* \rightarrow \{0,1\}^k$.
N_i:	label of a node of a Merkle tree.
Σ:	a yoking-proof of two tags.
Σ_G	a yoking-group proof of multiple groups.
K_N^i:	a key of a node N_i.
K_I^i:	an identification key of T_i.
K_A^i:	an authentication key of T_i.
K_D^i:	a decryption key of T_i.
$\|\|$:	concatenation of two bit-strings.

Table 7.1: Notations.

proof that multiple groups are presented simultaneously and issues the proof to the verifier. However, the reader cannot extract any private information, such as a tag's identity and the group's key.

7.2.2 Notations

To describe the concrete construction of our protocols, we give the notations in Table 7.1.

7.3 Security and Privacy Models

In this section, we define the security requirements and assumptions. A yoking-group proof can be considered as an extension of yoking-proof and grouping-proof, thus the security requirements are basically close to the security definitions of them. However, to estimate the security of an anonymous yoking-group proof protocol, it is necessary to describe a new security model.

7.3.1 Adversaries and Attacks

A yoking-group proof algorithm proves that pre-defined groups of tags have been simultaneously scanned. To evaluate the security of our protocol, we shall define the capabilities of an adversary as follows.

Definition 7.1 (Adversary)**.** *We define three adversaries as follows.*

- *An adversary $\mathcal{A}$ is allowed to capture all the interactions between the interrogating reader and participating tags.*
- *$\mathcal{A}$ can corrupt at most $n-1$ tags where n is the number of tags. In this case, the anonymity of compromised tags are ignored. Note that the corrupted tags are not*

destructive that they can be involved in the future communications.

- *$\mathcal{A}$ is allowed to interact with both tags and a reader in any order and messages can be altered by $\mathcal{A}$ in some manners.*

In terms of security, $\mathcal{A}$ can launch various attacks on the protocol to output a valid yoking-group proof. $\mathcal{A}$ has an ability to perform attacks based on the above definition. Informally, we described the following attacks as examples.

Replay Attack: $\mathcal{A}$ responds a reader's challenge to a target tag T by sending a message σ which is T's previous response.

Impersonation Attack: $\mathcal{A}$ attempts to forge a valid response σ of the target tag T without the knowledge of T's private key.

Collusion Attack: $\mathcal{A}$ intends to output a valid yoking-group proof with only a part of legitimate groups. $\mathcal{A}$ is allowed to interact with all the tags, while at least one tag is exclusive in each session. Then, $\mathcal{A}$ generates a proof according to the collections.

Trace Attack: $\mathcal{A}$ attempts to trace groups of tags in distinct sessions. We assume that target groups have the same quantity of tags, otherwise it is trivial to distinguish groups.

7.3.2 Ideal Functionalities

The universal composability (UC) framework [Can01, Can06] supports composable security even in complex systems. Most RFID applications are assumed to be deployed in a large and scalable system, thus we apply the UC model to analyze proposed protocols. In UC framework, it introduces models in ideal world which expresses robust protocol execution and real world where a PPT adversary is allowed to control adversarial parties and arranges the order of activation for parties. In the UC model, it defines a simulator $\mathcal{S}$ which emulates the real protocol ρ execution and translates runs of ρ with real world adversary $\mathcal{A}$ into runs of ρ which is executed by ideal functionality $\mathcal{F}$ with ideal world adversary $\widehat{\mathcal{A}}$. An adversary of UC model in both ideal world and real world shall interact with a PPT called *environment* $\mathcal{Z}$ which distinguishes whether the interactions collected are from $\mathcal{F}$ and $\widehat{\mathcal{A}}$ or from ρ and $\mathcal{A}$. To analyze the security of a protocol ρ, we say that ρ UC-*realizes* $\mathcal{F}$ if there is no $\mathcal{Z}$ can distinguish between ideal and real world with non-negligible probability.

In UC framework, a protocol execution is captured by a session identifier *sid* which is created by $\mathcal{Z}$. In each protocol run, all parties use the same *sid* and any input/output of the protocol is represented associated with *sid*. Specifically,

Functionality $\mathcal{F}_{ayp}$

Upon receiving input Initiate at reader R: It ignores the request if R is compromised. Otherwise, it generates a unique subsession identifier s and replaces any existing init record by $\mathsf{init}(s, R)$, and then outputs init.

Upon receiving input Initiate at tag T_i: It ignores the request if T_i is compromised. Otherwise, it generates a unique subsession identifier s'_i and replaces any existing init records by $\mathsf{init}(s'_i, T_i)$ and deletes any record link, and then outputs init.

Upon receiving input Link (s'_i, s'_j) at tag T_i: If there are records $\mathsf{init}(s'_j, T_j)$ or $\mathsf{link}(s'_j, T_j)$ and $\mathsf{init}(s'_i, T_i)$, and then it deletes $\mathsf{init}(s'_i, T_i)$, records and outputs $\mathsf{link}(s'_i, T_i)$.

Upon receiving input $\mathsf{Prove}(s, s'_i, s'_j)$: If there are records $\mathsf{init}(s, R)$, $\mathsf{link}(s'_i, T_i)$ and $\mathsf{link}(s'_j, T_j)$, then it records $\mathsf{proof}(s, s'_i, s'_j)$, deletes all link records and outputs proof.

Upon receiving input $\mathsf{Impersonate}(s, s'_i, T_j)$: If there are records $\mathsf{init}(s, R)$, $\mathsf{init}(s'_i, T_i)$ and T_j is a tag controlled by an adversary, and then it records and outputs $\mathsf{proof}(s, s'_i, ps'_j)$, where ps'_j is a pseudo-subsession.

Upon receiving input Verify(proof) at verifier: If there exists a record $\mathsf{proof}'(s, s'_i, s'_j)$= proof, and then it outputs valid, otherwise invalid.

Figure 7.1: Ideal functionality $\mathcal{F}_{ayp}$ for anonymous yoking-proof.

The ideal functionality $\mathcal{F}_{ayp}$ of anonymous yoking proof is depicted as in Figure 7.1 and the ideal functionality $\mathcal{F}_{aygp}$ of anonymous yoking-group proof is depicted as in Figure 7.2.

We describe parties and sessions involved in the protocol execution for both of proposed protocols as follows.

Sessions. The entire life-time of a protocol is represented by a single session. For each session *sid*, it consists of multiple subsessions which are initiated by protocol parties. Particularly, a unique subsession identifier *ssid* is assigned when a protocol party receives an input Initiate from $\mathcal{Z}$. Note that all parities in different subsessions share the same *sid*.

Parties. There are three types of parties: tag, reader and verifier. In each subsession *ssid*, there are arbitrarily many of tags that different instances of type tag are allowed. However, there is only one instance type of reader in a subsession. As UC entities, such as $\mathcal{Z}$ and $\mathcal{A}$, are not parties of a protocol, an adversary is allowed to control multiple protocol parties.

Yoking-proof. A successful yoking proof in the real world implies that the output of each tag contains the other tag's input which is generated by a piece of secret. An adversary is entitled to fully control the network and can selects involved

partners. In the ideal world, the protocol execution is emulated by the calls of Link and Prove activities.

Yoking-group proof. A successful yoking-group proof in the real world implies that yoking proof of each pair of child nodes is successful, correctness of group secret reconstruction and the output of one group involves the secrets of another group. As above, the adversary can choose participating partners and control the network. In the ideal world, the protocol execution is emulated by the calls of Link, Build and Prove activities.

Anonymity. In the ideal functionality $\mathcal{F}_{ayp}$ of anonymous yoking proof, it only leaks a party's type information which is "tag" or "reader". In the ideal functionality $\mathcal{F}_{aygp}$ of anonymous yoking-group proof, it discloses the type information of involved parties as well as the index when the party is type tag. Note that an index of a tag indicates the position of a tag in a group other than the identity of a tag. Clearly, the tags and readers are distinguishable as they proceed distinct tasks.

7.4 Building Block

A novel anonymous yoking-proof protocol is proposed in this section. It is an essential building block of our yoking-group proofs described in Section 7.5. The proposed yoking-proof is a lightweight symmetric-key based protocol which only requires one hash computation to each tag. The untrusted third party who controls the reader has an obligation to ensure the sequence of interactions and collects responses of tags without obtaining any privacy of target tags. Then, it compiles the proof and submits it to the verifier who is referred to a TTP. During the reader-tag communication, the TTP is unnecessary to be online and all proofs can be verified latter.

Our protocol is described as in Figure 7.3 that any two legal tags are eligible to be presented in the proof. For each tag, the TTP stores two keys in its non-volatile memory, namely identification key K_I and authentication key K_A, such that $K_I, K_A \in \mathcal{K}$, where $\mathcal{K}$ is the defined key space. The protocol consists of two rounds of challenge-response interactions that the proof process is initiated by the reader R. Firstly, the reader randomly chooses $a \in \{0,1\}^k$, where k is a security parameter, and sends a as a challenge to both of tags T_l and T_r. The nonce a aims at resisting the replay attack which could be launched by an adversary who attempts to impersonate the tag. Upon receiving the challenge, T_l selects a random number $b_l \in \{0,1\}^k$ and computes (σ_l, Δ_l) that b_l is considered as an ephemeral key to protect the authentication key K_A^l. The tag T_r executes the same as T_l and sends challenge (σ_r, Δ_r) to the reader. Then, the reader swaps two responses and sends (σ_r, Δ_r), (σ_l, Δ_l) to T_l, T_r, respectively. Upon receiving the challenge, the tag

Functionality $\mathcal{F}_{aygp}$

Upon receiving input Initiate at reader R: It ignores the request if R is compromised. Otherwise, it generates a unique subsession identifier s and replaces any existing init record by $\mathsf{init}(s, R)$, and then outputs init.

Upon receiving input Initiate at tag T_i, where $i \geq 1$: It ignores the request if T_i is compromised. Otherwise, it generates a unique subsession identifier s'_i and replaces any existing init records by $\mathsf{init}(s'_i, T_i, i)$ and deletes any record link, and then outputs init.

Upon receiving input Link (s'_i, s'_j) at tag T_i: If there are records $\mathsf{init}(s'_j, T_j, j)$ or $\mathsf{link}(s'_j, T_j, j)$ and $\mathsf{init}(s'_i, T_i, i)$, then it deletes $\mathsf{init}(s'_i, T_i, i)$, records and outputs $\mathsf{link}(s'_i, T_i, i)$.

Upon receiving input $\mathsf{Build}(s'_i, s'_j, s'_k)$ at tag T_k: If there are records $\mathsf{link}(s'_i, T_i, i)$, $\mathsf{link}(s'_j, T_j, j)$ and $\mathsf{init}(s'_k, T_k, k)$, then it deletes $\mathsf{init}(s'_k, T_k, k)$, records and outputs $\mathsf{build}(s'_k, T_k, k)$.

Upon receiving input $\mathsf{Build}(s'_i, s'_j, s'_z, s'_k)$ at tag T_k: If there are records $\mathsf{link}(s'_i, T_i, i)$, $\mathsf{link}(s'_j, T_j, j)$, $\mathsf{init}(s'_z, T_z, z)$ and $\mathsf{init}(s'_k, T_k, k)$, then it deletes $\mathsf{init}(s'_k, T_k, k)$, records and outputs $\mathsf{build}(s'_k, T_k, k)$.

Upon receiving input $\mathsf{Prove}(s, \{s'_m\})$, where $m \geq 1$: If there is a record $\mathsf{init}(s, R)$; for all $s'_i \in \{s'_m\}$, there exists $\mathsf{build}(s'_i, T_i, i)$, then it records $\mathsf{proof}(s, \{s'_m\})$, deletes all build records and outputs proof.

Upon receiving input $\mathsf{Impersonate}(s, \{s'_m\} \setminus \{s'_j\}, T_j)$: If there is a record $\mathsf{init}(s, R)$; for all $s'_i \in \{s'_m\} \setminus \{s'_j\}$ and T_j is a tag controlled by an adversary, then it records and outputs $\mathsf{proof}(s, (\{s'_m\} \setminus \{s'_j\}) \cup \{ps'_j\})$, where ps'_j is a pseudo-subsession.

Upon receiving input Verify(proof) at verifier: If there exists a record $\mathsf{proof}'(s, \{s'_m\}) = \mathsf{proof}$, and then it outputs valid, otherwise invalid.

Figure 7.2: Ideal functionality $\mathcal{F}_{aygp}$ for anonymous yoking-group proof

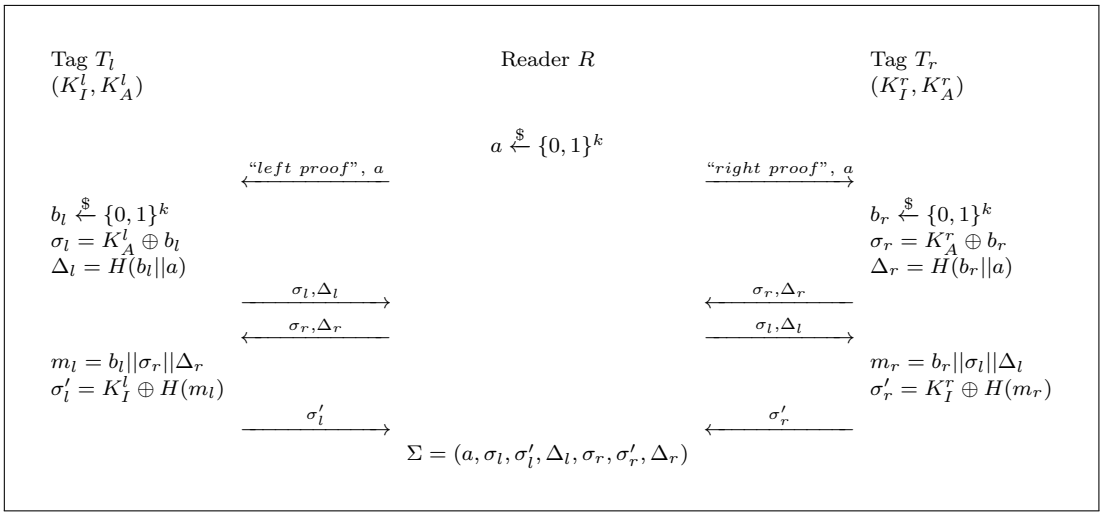

Figure 7.3: Anonymous yoking-proof protocol.

generates a message m and encrypts the message by using the identification key K_I. At last, the reader compiles the responses received from two tags and generates a yoking proof Σ. To verify the proof, the verifier who maintains pairs of keys (K_A, K_I) searches pairs (K_A^l, K_I^l) and (K_A^r, K_I^r) and checks the proof as follows,

$$\Delta_l \stackrel{?}{=} H(\sigma_l \oplus K_A^l||a),\ \sigma'_l \stackrel{?}{=} K_I^l \oplus H(\sigma_l \oplus K_A^l||\sigma_r||\Delta_r)$$

and

$$\Delta_r \stackrel{?}{=} H(\sigma_r \oplus K_A^r||a),\ \sigma'_r \stackrel{?}{=} K_I^r \oplus H(\sigma_r \oplus K_A^r||\sigma_l||\Delta_l).$$

If and only if the above equations hold, the proof Σ is accepted, otherwise it is rejected.

In this protocol, the identity of a tag is considered as a piece of sensitive information which shall be prevented from disclosing to the reader. We assume that each tag T_i is assigned a distinct key pair (K_A^i, K_I^i) which indicates its unique identity in the database and only the verifier can reveal the keys.

The proposed protocol also employs the timeout mechanism which guarantees the interaction is completed in a small period of time. The tag and reader abort the protocol if the response received later than the defined time window. To simplify the description, we assume that there is a timeout for each round where the recipient is awaiting any response. For the rest of this chapter, we will adopt the same definition.

7.5 Proposed Protocol

In this section, we demonstrate an anonymous yoking-group proof protocol which allows an untrusted third party to prove that two groups of tags have been scanned

simultaneously. Intuitively, the protocol guarantees plenty of tags have been presented during the protocol run. However, the further requirement of the protocol is that the verifier needs to identify which groups of tags have been scanned other than the validity of tags. The proposed protocol is based on the building block introduced in Section 7.4. Although a tag may handle more computations during the authentication, the minimum computational cost remains the same as in our yoking-proof protocol. In addition, the computational cost to the verifier is irrelevant to the number of tags.

7.5.1 Anonymous Yoking-Group Proof

An anonymous yoking-group protocol is described as in two phases: the grouping phase and the yoking phase. To show the presence of distinct groups, a reader firstly requests tags to generate a grouping-proof and then yokes groups to generate a yoking-proof. We call the proof created by the protocol as a yoking-group proof Σ_G.

In some previous grouping-proof protocols, the tags are arranged in a specific logical structure. For instance, Burmester and Munilla [BM13] assume that a group of tags form a logical ring and the tags are labelled from 1 to n, where n is the quantity of tags. Some other protocols [BLS+12, SS05] also consider a group of tags as a ring or a chain during the authentication. They also consecutively assign each tag a distinct number. Usually, the protocols which apply symmetric-key cryptography pre-define a group by sharing a group key, identity or distinctive counters. Similar method is employed, especially in some protocols which intend to protect the tag's identity and provide anonymity proofs. One reason for this is that the symmetric-key based protocols normally require exhaustive key search to check the validity of proofs and undefined groups may cause efficiency problems as the verifier has to test all the possible combinations of tags. Hence, in our protocol, we assume that the groups are defined prior to being authenticated.

An important consideration to a grouping-proof protocol which needs to provide group anonymity is that the computational cost to a verifier shall be lower than the way of exhaustive key search for each tag. As one phase of yoking-group proof protocol, the group authentication in our protocol only requires constant computational cost $O(1)$ in verifying the validity of a group. Moreover, the total cost of proof verification equals to the cost of authenticating our yoking proof. To reduce the computational cost and guarantee the integrity of a group, we employ a binary tree as a logical structure to represent tags and each tag is assigned a different index in the specific group. The verifier only needs to check the validity of a proof generated by the root tag instead of all the tags.

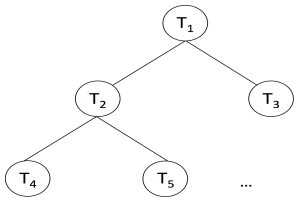

Figure 7.4: Labeled tree for a group of tags.

The size of a proof influences the performance of a protocol. Normally, a larger size indicates that more components should be checked during the verification. Regardless whether the proof is a compressed message, it is undesirable if the tags are verified individually. As a feature of our anonymous yoking-group proof protocol, the length of a proof remains the same size of our yoking-proof.

7.5.2 Constructions

Our protocol is described in Figure 7.5 that it shows the proof for two groups. To simplify our protocol, we only consider that each group consist of three tags. It is a typical case which demonstrates the essential process of most situations, while the protocol can be extended to groups which contain any number of tags.

Without loss of generality, we firstly introduce the binary tree which represents the logical structure of a group of tags. According to Figure 7.4, each node, including leaves and root, represents a tag. From top to down and left to right in each level, a tag T is assigned to an index i and the root is labelled as 1. For a tag T_i, it stores three keys which are identification key K_I^i, authentication key K_A^i and decryption key K_D^i. Initially, the tag owner sets these keys as follows.

$$K_I^i \xleftarrow{\$} \mathcal{K},\ K_D^i \xleftarrow{\$} \mathcal{K},\ K_A^i = K_D^p,$$

where $K_D^p \in \mathcal{K}$ is the decryption key of T_i's parent. In case that if T_i is the root of a tree, the owner randomly chooses a key K_A^i from $\mathcal{K}$. The group key K_G is constructed as in a Merkle tree that for each non-leaf node N_i, the key K_N^i is calculated as

$$K_N^i = H(K_N^l || K_N^r || K_I^i),$$

where K_N^l, K_N^r are node keys of N_i's left and right child respectively and K_I^i is the identification key of T_i in N_i. If T_i is in a leaf of a tree, we set $K_N^i = K_I^i$. Clearly, apart from the leaves, a tag does not keep a node key, instead, the node keys are built during the protocol execution. Indeed, the protocol rebuilds the Merkle tree and outputs the group key K_N^1 at the root T_1. The verifier maintains the group key and it indicates the identity of a specific group of tags.

In the grouping phase of our protocol, tags generate a proof of group. As an example, we show the grouping for a group which has three tags T_1, T_2 and T_3. Note that the grouping-proof is close to yoking-group proof if there are only three tags in a group. Hence, we show a single round of grouping-proof via our yoking-group proof protocol and the grouping-proof can be accomplished recursively for a larger group. We denote that T_1 is a root and T_2, T_3 are respectively left and right child of T_1. Without loss of generality, we refer tags T_2 and T_3 to T_l and T_r, respectively.

To start with, the reader R broadcasts a request along with a nonce a. A tag sends its index as a part of the first response to R. According to our system model, the reader builds a hash tree structure based on the indexes and maintains a link to each tag. In terms of one group, T_l, T_r and R firstly run the yoking-proof protocol illustrated in Section 7.4. The only difference is that the reader sets the message

$$\sigma_l = \sigma_l||\sigma_i, \quad \sigma_r = \sigma_r||\sigma_i,$$

where σ_i is generally the response of challenge a from the tags' parent. The reader collects a yoking-proof Σ and T_i's response (σ_i, Δ_i). Concurrently, the reader obtains the reply (σ_j, Δ_j) form T_j, which is the root of another group.

After compiling proofs, the reader sends $m = (\sigma_l, \sigma'_l, \Delta_l, \sigma_r, \sigma'_r, \Delta_r, \sigma_j, \Delta_j)$ to the root T_i and starts the yoking phase. Upon receiving the message m, T_i extracts the node keys K_N^l, K_N^r of two tags and computes the node key K_N^i. In a case that if node N_i of a tree only has the left child, the reader sets $(\sigma_r, \sigma'_r) = (\sigma_i, 0)$, where σ_i is a response of N_i. Accordingly, the tag T_i ignores a message σ'_r and simply sets $K_N^r = 0$. We should note that this step does not guarantee the correctness of node keys as a tag handles the received message without authentication. Since the goal of our protocol is to check the simultaneous scanning of groups and the hash tree ensures the integrity of a group, there is no need to verify messages during the proof. The verifier is able to verify the group's integrity when it checks the validity of a yoking-group proof.

Finally, T_i outputs a reply σ'_i by using K_N^i and T_j outputs σ'_j via the same approach. Upon receiving two responses $(\sigma_i, \sigma'_i, \Delta_i)$ and $(\sigma_j, \sigma'_j, \Delta_j)$, the reader compiles a yoking-group proof Σ_G. The verifier checks the validity of proofs by searching key pairs (K_A^i, K_N^i) and (K_A^j, K_N^j).

The extension of the protocol for multi-group proof is as follows. Assume that there are n groups. In the grouping phase, a reader collects each root tag's response (σ_i, Δ_i), where $i = 1, 2, \cdots, n$. For example, (σ_1, Δ_1) is the response of the root tag of group G_1. In the yoking phase, the reader transmits (σ_k, Δ_k), where $k = 1, 2, \cdots, n-1$, from group G_k to G_{k+1}, then transmits (σ_n, Δ_n) to G_1. Note that this step can be performed concurrently and the following steps are the same as in

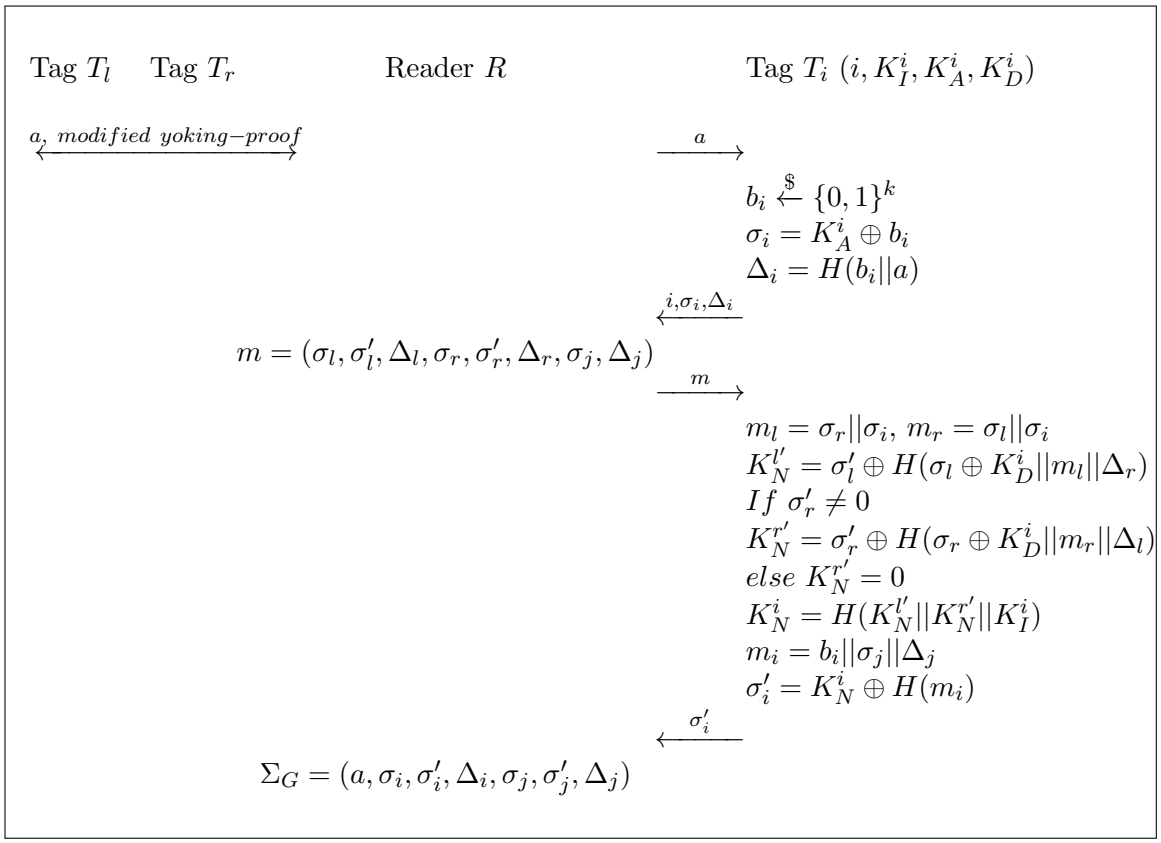

Figure 7.5: Yoking-group proof with group anonymity for two groups.

two-group version.

Informally, the protocol protects the group's privacy as its identity is concealed by a random number b and a collision-resistant cryptographic hash function H. We assume that an adversary cannot corrupt a tag to obtain its internal states. Without given the value of b, the tag's keys which are also considered as an identity can be hidden perfectly during the communication. It can only be authenticated by the verifier through searching possible keys in its database. The formal security proof of our protocol is given in Section 7.6.

7.6 Security Analysis

In this section, we analyze the security of proposed protocols. We show that both our anonymous yoking-proof and anonymous yoking-group proof are secure under the UC framework.

Theorem 7.1. *Our anonymous yoking-proof protocol UC-realizes the functionality $\mathcal{F}_{ayp}$ with random oracles.*

Proof. To simulate the real world protocol execution, we firstly activate the environment $\mathcal{Z}$, create protocol parties of anonymous yoking proof that a reader, some

tags and a verifier, and the adversary $\mathcal{A}$ is activated. To simplify the description, we now refer G as a function, where $G : b \xleftarrow{\$} \{0,1\}^k$. The same notion is applied in the proof of Theorem 7.2. The simulator $\mathcal{S}_{ayp}$ provides the following actions to emulate protocol run in the real world.

- $\mathcal{S}_{ayp}$ simulates copies of tag $\widehat{T}_i$, reader $\widehat{R}$, verifier $\widehat{V}$ and an ideal world adversary $\widehat{\mathcal{A}}$. Then, it activates $\widehat{\mathcal{A}}$.
- $\mathcal{S}_{ayp}$ initiates a database $\widehat{D}$ for the verifier $\widehat{V}$ that it adds or removes keys of both honest tags and adversary controlled tags to $\widehat{D}$.
- $\mathcal{S}_{ayp}$ collects the requests of all real world parties and forwards them without any modification to the functionality $\mathcal{F}_{ayp}$ as in Figure 7.6. Then $\mathcal{S}_{ayp}$ returns the outputs to $\widehat{\mathcal{A}}$.
- $\mathcal{S}_{ayp}$ simulates all the interactions between $\mathcal{Z}$ and external visible parties of the protocols. Specifically, $\mathcal{S}_{ayp}$ calls Initiate when there is a request to the reader or tag initiation. In the ideal world, $\mathcal{F}_{ayp}$ calls Link, Prove, Impersonate and Verify when received from $\mathcal{S}_{ayp}$ the input request linking two tags, forwarding unmodified messages between tags and the reader, impersonating a simulated tag which is controlled by an adversary and translating unmodified messages between the reader and the verifier, respectively.
- If the input of honest tags to the Prove in the ideal world has been modified, $\mathcal{S}_{ayp}$ ignores the request unless it is from an adversarial controlled reader.

Clearly, the main difference of the protocol execution between ideal world and real world is related to the functions applied. In the ideal world, the true random functions which output uniformly independent values are employed. However, in the real world, a hash function H and the random number generator G are used.

We remark the requirements of $\mathcal{F}_{ayp}$ for anonymous yoking proof. It provides the unforgeability that an honest tag cannot be impersonated by an adversary. The random values generated during the protocol execution is unknown to the adversary. A tag's identity is perfectly protected that no deterministic information of a tag disclosed to an adversary. $\mathcal{F}_{ayp}$ resists replay, collusion, impersonation and other attacks even the adversary is allowed to interact with $\mathcal{Z}$ in arbitrary fashion.

Let E be the event that the simulation fails. If we assume that the output of function H and G are both truly random and independent, the simulation is perfect unless E occurs when $\mathcal{S}_{ayp}$ ignores the Prove request as the message of an honest tag has been altered by an adversary. That is, the adversary $\mathcal{A}$ forges random numbers and outputs a valid proof in the real world. It occurs only if $\mathcal{A}$ could guess the random number b generated by G and find a collision of hash function H.

Simulator $\mathcal{S}_{ayp}$

Upon receiving input Initiate at reader $\widehat{R}$ from $\widehat{\mathcal{A}}$: It delivers the request to $\mathcal{F}_{ayp}$. On output init(s, R), it selects a new random number a from a list $\widehat{L}$ and outputs (s, a) to $\widehat{\mathcal{A}}$.

Upon receiving input Initiate at tag $\widehat{T}_i$ from $\widehat{\mathcal{A}}$: It delivers the request to $\mathcal{F}_{ayp}$. On output init(s'_i, T_i), it assigns keys (K_I^i, K_A^i) which are from the database $\widehat{D}$ to tag $\widehat{T}_i$ and sends s'_i to $\widehat{\mathcal{A}}$.

Upon receiving input Link(s'_i, s'_j) at tag $\widehat{T}_i$ from $\widehat{\mathcal{A}}$: It delivers the request to $\mathcal{F}_{ayp}$. On output link(s'_i, T_i), $\widehat{T}_i$ runs the anonymous yoking proof, except that it replaces H by a hash function and G by a random number generator, and outputs $(\sigma_i, \sigma'_i, \Delta_i)$ to $\widehat{R}$ and $\widehat{\mathcal{A}}$.

Upon receiving input Prove(s, s'_i, s'_j) at $\widehat{R}$ from $\widehat{\mathcal{A}}$: It delivers the request to $\mathcal{F}_{ayp}$. On output proof(s, s'_i, s'_j), $\widehat{R}$ outputs Σ to $\widehat{\mathcal{A}}$.

Upon receiving input Impersonate(s, s'_i, T_j) at tag T_j from $\widehat{\mathcal{A}}$: It delivers the request to $\mathcal{F}_{ayp}$. On output proof(s, s'_i, ps'_j), it outputs proof to $\widehat{\mathcal{A}}$.

Upon receiving input Verify(proof) at verifier $\widehat{V}$: It delivers the request to $\mathcal{F}_{ayp}$. On output valid/invalid, it send this to $\widehat{\mathcal{A}}$.

Figure 7.6: Simulator $\mathcal{S}_{ayp}$ for anonymous yoking-proof.

Given a random a which indicates a subsession of the reader, the probability $\Pr[E]$ that event E occurs is at most $2^{-2k}tn$, where k is the security parameter as well as the length of hash value, t is the number of tags in the database and n is the interactions placed during the subsession. Hence, the environment $\mathcal{Z}$ can distinguish the simulation in polynomial time whether happened in the ideal world or the real world with probability $\Pr[E]$, which is negligible. □

Theorem 7.2. *Our anonymous yoking-group proof protocol UC-realizes the functionality $\mathcal{F}_{aygp}$.*

Proof. To emulate the real world protocol run, we activate the environment $\mathcal{Z}$. Then, we create anonymous yoking-group proof protocol parties, such as a reader, multiple tags, a verifier and a real world adversary $\mathcal{A}$ which is activated immediately. The simulator $\mathcal{S}_{aygp}$ which is described as in Figure 7.7 provides actions to simulate the protocol execution in the real world. The actions supported by $\mathcal{S}_{aygp}$ are similar to the actions described in the proof of Theorem 7.1. Additionally, in the ideal world, if $\mathcal{F}_{aygp}$ receives from $\mathcal{S}_{aygp}$ the request of building a node key of several tags, it calls Build to generate corresponding responses.

The main differences of the protocol execution between the ideal world and the real world are these: true random functions which output uniformly independent

Simulator $\mathcal{S}_{aygp}$

Upon receiving input Initiate at reader $\widehat{R}$ from $\widehat{\mathcal{A}}$: It delivers the request to $\mathcal{F}_{aygp}$. On output init(s, R), it selects a new random number a from a list $\widehat{L}$ and outputs (s, a) to $\widehat{\mathcal{A}}$.

Upon receiving input Initiate at tag $\widehat{T}_i$ from $\widehat{\mathcal{A}}$, where $1 \geq i$: It delivers the request to $\mathcal{F}_{aygp}$. On output init$(s'_i, T_i$,i), it assigns keys (K_I^i, K_A^i, K_D^i) which are from the database $\widehat{D}$ to tag $\widehat{T}_i$ and sends (s'_i, i) to $\widehat{\mathcal{A}}$.

Upon receiving input Link(s'_i, s'_j) at tag $\widehat{T}_i$ from $\widehat{\mathcal{A}}$: It delivers the request to $\mathcal{F}_{aygp}$. On output link(s'_i, T_i, i), $\widehat{T}_i$ runs the anonymous yoking-group proof, except that it replaces H by a hash function and G by a random number generator, and outputs $(\sigma_i, \sigma'_i, \Delta_i)$ to $\widehat{R}$ and $\widehat{\mathcal{A}}$.

Upon receiving input Build(s'_i, s'_j, s'_k) at tag $\widehat{T_k}$ from $\widehat{\mathcal{A}}$: It delivers the request the $\mathcal{F}_{aygp}$. On output build(s'_k, T'_k, k), $\widehat{T}_j$ runs the anonymous yoking-group proof, except that it replaces H by a hash function and G by a random number generator and outputs σ'_k to $\widehat{R}$ and $\widehat{A}$.

Upon receiving input Build(s'_i, s'_j, s'_z, s'_k) at tag $\widehat{T_k}$ from $\widehat{\mathcal{A}}$: It delivers the request the $\mathcal{F}_{aygp}$. On output build(s'_k, T'_k, k), $\widehat{T_k}$ runs the anonymous yoking-group proof, except that it replaces H by a hash function and G by a random number generator and outputs σ'_k to $\widehat{R}$ and $\widehat{A}$.

Upon receiving input Prove(s, s'_i, s'_j) at $\widehat{R}$ from $\widehat{\mathcal{A}}$: It delivers the request to $\mathcal{F}_{aypg}$. On output proof(s, s'_i, s'_j), $\widehat{R}$ outputs Σ_G to $\widehat{\mathcal{A}}$.

Upon receiving input Impersonate(s, s'_i, T_j) at tag T_j from $\widehat{\mathcal{A}}$: It delivers the request to $\mathcal{F}_{aygp}$. On output proof(s, s'_i, ps'_j), it outputs proof to $\widehat{\mathcal{A}}$.

Upon receiving input Verify(proof) at verifier $\widehat{V}$: It delivers the request to $\mathcal{F}_{aygp}$. On output valid/invalid, it send this to $\widehat{\mathcal{A}}$.

Figure 7.7: Simulator $\mathcal{S}_{aygp}$ for anonymous yoking-proof.

values are employed, while the H is a hash function and G is a random number generator in the real world.

The Functionality $\mathcal{F}_{aygp}$ for anonymous yoking-group proof provides unforgeability which prevents the impersonation of an honest tag, untraceability that an adversary cannot link tags or groups in different sessions, anonymity that an untrusted third party cannot disclose the identity of a group and any random numbers generated during the protocol run is invisible to an adversary. In any order of activation between an adversary and $\mathcal{Z}$, $\mathcal{F}_{aygp}$ prevents attacks, such as replay, collusion and impersonation.

Let E be the event that the simulation fails. E occurs if $\mathcal{S}_{aygp}$ ignores the Prove request from $\mathcal{A}$ when we consider the functions H and G are true random. As the yoking-group proof is generated by two root tags as in yoking-group proof, the probability $\Pr[E]$ can be analyzed similarly to the previous proof that the environment $\mathcal{Z}$ can determine whether the simulation is occurred in the real world or and ideal world with negligible probability at most $2^{-2k}mn$, where m is the number of groups and n is the interactions placed during the subsession.

□

7.7 Conclusion

In this chapter, we introduced a novel concept of anonymous yoking-group proofs. The proposed anonymous yoking-proof protocol achieves computational efficiency on tags and remains the required security properties. Our anonymous yoking-group proofs convince a verifier that multiple groups of tags are simultaneously scanned. We discussed some common attacks to RFID protocols and the security and privacy were formally proved in the UC framework.

Chapter 8

Conclusion

The privacy and security are two important requirements of RFID systems. A challenging problem in RFID research is the tradeoff among security, privacy, efficiency and hardware cost. Low-cost RFID tags can normally carry lightweight operations such as CRC and PRNG, which are too weak to deal with cryptographic operations, in order to construct secure protocols. Because cryptographic calculations require a high level of computational capability, the cost of tags will be increased. In this thesis, we focused on various security and privacy problems of RFID protocols, considering the computation requirement of tags.

In Chapter 3, we focused on the wide-strong privacy of RFID tag authentication protocols. We showed the vulnerabilities of a recent protocol and provided some solutions. We then presented a repaired protocol which is proven secure. We also proposed a new ECC-based RFID authentication protocol which allows a reader to extract a tag's signature. The proposed protocol is private against a wide-strong adversary.

In Chapter 4, we considered the security of ownership transfer protocols against strong attacks such as the replacement attack. In our solution, we adopted an ownership chain to preserve the authenticity of tags. It guaranteed a valid ownership even if the internal state of a tag is disclosed. The proposed protocol prevented an unauthorized user from transferring ownership of tags. We defined a formal security model for ownership transfer protocols and proved that our protocol is secure.

In Chapter 5, we proposed the first secure shared RFID ownership transfer protocol without TTP. We gave a formal definition of shared ownership transfer and defined a security model, which captures active attacks such as the collusion attacks and corruption attacks. Some other attacks in collusion fashion were also illustrated. The proposed protocol only needs constant tag-key size and provides partial ownership verification.

In Chapter 6, we introduced a novel concept of authorized RFID authentication protocols. We reviewed the relationship among RFID tags, RFID readers and backend server. It shows that we need to address some new issues if they are independent. We then proposed three ARA protocols with distinct features and security levels. Formal models of security and privacy were defined against both strong and weak adversaries. Our symmetric-key based protocol achieves weak privacy and two ECC-based protocols are secure in strong model.

In Chapter 7, we introduced the yoking-group proof which is an extension of the

grouping proof. It allows a verifier to check the simultaneous presence of multiple groups of tags. We firstly proposed a lightweight anonymous yoking proof which only requires one hash computation on tag. We then used it as a building block to construct our anonymous yoking-group proof protocol. The protocol guarantees the existence of multiple tag groups and the size of proof is constant. Two proposed protocols were also proven secure in universal composability framework.

Future Work. Scalability is an important requirement of RFID systems. Symmetric key based RFID authentication protocols usually require exhaustive key search which reduces efficiency. Many solutions have been proposed to address this problem. Public key based protocols can achieve constant tag authentication time. However, we demonstrated that some cases, such as ARA protocols, still need new solutions. Our ARA protocol 2 has a linear communication cost and the ARA protocol 3 requires exhaustive key search. In our future work, we will explore solutions to address the scalability issue in public key based RFID authentication protocols. We will also consider solutions to avoid exhaustive key search in symmetric-key based anonymous yoking-group proof.

Bibliography

[Aba09] Jemal H. Abawajy. Enhancing RFID tag resistance against cloning attack. In Yang Xiang, Javier Lopez, Haining Wang, and Wanlei Zhou, editors, *Third International Conference on Network and System Security, NSS 2009, Gold Coast, Queensland, Australia, October 19-21, 2009*, pages 18–23. IEEE Computer Society, 2009.

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of dhies. In David Naccache, editor, *CT-RSA*, volume 2020 of *LNCS*, pages 143–158. Springer, 2001.

[Aby12] Mohammad Reza Sohizadeh Abyaneh. On the privacy of two tag ownership transfer protocols for RFIDs. *CoRR*, abs/1202.4663, 2012.

[AdM04] Giuseppe Ateniese and Breno de Medeiros. Identity-based chameleon hash and applications. In Ari Juels, editor, *Financial Cryptography*, volume 3110 of *LNCS*, pages 164–180. Springer, 2004.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005.

[BC93] Stefan Brands and David Chaum. Distance-bounding protocols (extended abstract). In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *LNCS*, pages 344–359. Springer, 1993.

[BCD06] Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. HB^{++}: a lightweight authentication protocol secure against some attacks. In *SecPerU*, pages 28–33. IEEE Computer Society, 2006.

[BCI08a] Julien Bringer, Hervé Chabanne, and Thomas Icart. Cryptanalysis of EC-RAC, a RFID identification protocol. In Matthew K. Franklin, Lucas Chi Kwong Hui, and Duncan S. Wong, editors, *CANS*, volume 5339 of *LNCS*, pages 149–161. Springer, 2008.

[BCI08b] Julien Bringer, Hervé Chabanne, and Thomas Icart. Improved privacy of the tree-based hash protocols using physically unclonable function. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *Security and Cryptography for Networks, 6th International Conference, SCN 2008, Amalfi, Italy, September 10-12, 2008. Proceedings*, volume 5229 of *Lecture Notes in Computer Science*, pages 77–91. Springer, 2008.

[BdMM08] Mike Burmester, Breno de Medeiros, and Rossana Motta. Provably secure grouping-proofs for RFID tags. In Gilles Grimaud and François-Xavier Standaert, editors, *CARDIS*, volume 5189 of *LNCS*, pages 176–190, 2008.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

[BJRS13] Kevin D. Bowers, Ari Juels, Ronald L. Rivest, and Emily Shen. Drifting keys: Impersonation detection for constrained devices. In *Proceedings of the IEEE INFOCOM 2013, Turin, Italy, April 14-19, 2013*, pages 1025–1033. IEEE, 2013.

[BLdMT09] Mike Burmester, Tri Van Le, Breno de Medeiros, and Gene Tsudik. Universally composable RFID identification and authentication protocols. *ACM Trans. Inf. Syst. Secur.*, 12(4):1–33, 2009.

[BLS04a] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Efficient implementation of pairing-based cryptosystems. *J. Cryptology*, 17(4):321–334, 2004.

[BLS04b] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

[BLS+10] Lejla Batina, Yong Ki Lee, Stefaan Seys, Dave Singelée, and Ingrid Verbauwhede. Privacy-preserving ECC-based grouping proofs for RFID. In Mike Burmester, Gene Tsudik, Spyros S. Magliveras, and Ivana Ilic, editors, *ISC*, Lecture Notes in Computer Science, pages 159–165. Springer, 2010.

[BLS+12] Lejla Batina, Yong Ki Lee, Stefaan Seys, Dave Singelée, and Ingrid Verbauwhede. Extending ECC-based RFID authentication protocols to privacy-preserving multi-party grouping proofs. *Personal and Ubiquitous Computing*, 16(3):323–335, 2012.

[BM11] Mike Burmester and Jorge Munilla. Lightweight RFID authentication with forward and backward security. *ACM Trans. Inf. Syst. Secur.*, 14(1):11, 2011.

[BM13] Mike Burmester and Jorge Munilla. Distributed group authentication for RFID supply management. *IACR Cryptology ePrint Archive*, 2013:779, 2013.

[BMV13a] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. Practical & provably secure distance-bounding. *IACR Cryptology ePrint Archive*, 2013:465, 2013.

[BMV13b] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. Secure and lightweight distance-bounding. In Gildas Avoine and Orhun Kara, editors, *Lightweight Cryptography for Security and Privacy - Second International Workshop, LightSec 2013, Gebze, Turkey, May 6-7, 2013, Revised Selected Papers*, volume 8162 of *Lecture Notes in Computer Science*, pages 97–113. Springer, 2013.

[BMV13c] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. Towards secure distance bounding. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *LNCS*, pages 55–67. Springer, 2013.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 390–399. ACM, 2006.

[BP09] Jens-Matthias Bohli and Andreas Pashalidis. Relations among privacy notions. In Roger Dingledine and Philippe Golle, editors, *Financial Cryptography*, volume 5628 of *LNCS*, pages 362–380. Springer, 2009.

[BP11] Jens-Matthias Bohli and Andreas Pashalidis. Relations among privacy notions. *ACM Trans. Inf. Syst. Secur.*, 14(1):4, 2011.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR97] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *LNCS*, pages 470–484. Springer, 1997.

[BR05] Leonid Bolotnyy and Gabriel Robins. Generalized "yoking-proofs" for a group of RFID tags. In *International conference on mobile and ubiquitous systems: networking and service, MobiQuitous*, pages 1–4. IEEE Computer Society, 2005.

[BR07] Leonid Bolotnyy and Gabriel Robins. Physically unclonable function-based security and privacy in RFID systems. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2007), 19-23 March 2007, White Plains, New York, USA*, pages 211–220. IEEE Computer Society, 2007.

[BSSV11] Lejla Batina, Stefaan Seys, Dave Singelée, and Ingrid Verbauwhede. Hierarchical ECC-based RFID authentication protocol. In Ari Juels and Christof Paar, editors, *RFIDSec*, volume 7055 of *LNCS*, pages 183–201. Springer, 2011.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.

[Can06] Ran Canetti. Security and composition of cryptographic protocols: A tutorial. *IACR Cryptology ePrint Archive*, 2006:465, 2006.

[CCEG10] Sébastien Canard, Iwen Coisel, Jonathan Etrog, and Marc Girault. Privacy-preserving RFID systems: Model and constructions. *IACR Cryptology ePrint Archive*, 2010:405, 2010.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CL09] Hung-Yu Chien and Shih-Bin Liu. Tree-based RFID yoking proof. In *Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC '09. International Conference on*, volume 1, pages 550–553, April 2009.

[CLL05] Eun Young Choi, Su-Mi Lee, and Dong Hoon Lee. Efficient RFID authentication protocol for ubiquitous computing environment. In Tomoya Enokido, Lu Yan, Bin Xiao, Daeyoung Kim, Yuan-Shun Dai, and Laurence Tianruo Yang, editors, *Embedded and Ubiquitous Computing - EUC 2005 Workshops, EUC 2005 Workshops: UISW, NCUS, SecUbiq, USN, and TAUES, Nagasaki, Japan, December 6-9, 2005, Proceedings*, volume 3823 of *LNCS*, pages 945–954. Springer, 2005.

[CM13] Iwen Coisel and Tania Martin. Untangling RFID privacy models. *Journal of Computer Networks and Communications*, 2013:710275:1–710275:26, 2013.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *LNCS*, pages 13–25. Springer, 1998.

[CYH+08] Jung-Sik Cho, Sang-Soo Yeo, Suchul Hwang, Sang-Yong Rhee, and Sung Kwon Kim. Enhanced yoking proof protocols for RFID tags and tag groups. In *AINA*, pages 1591–1596. IEEE Computer Society, 2008.

[CYK11] Jung-Sik Cho, Sang-Soo Yeo, and Sung Kwon Kim. Securing against brute-force attack: A hash-based RFID mutual authentication protocol using a secret value. *Computer Communications*, 34(3):391–397, 2011.

[CYWL11] Hung-Yu Chien, Chia-Chuan Yang, Tzong-Chen Wu, and Chin-Feng Lee. Two RFID-based solutions to enhance inpatient medication safety. *J. Medical Systems*, 35(3):369–375, 2011.

[Dam87] Ivan Damgård. Collision free hash functions and public key signature schemes. In *EUROCRYPT*, pages 203–216, 1987.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DK09] Dang Nguyen Duc and Kwangjo Kim. Grouping-proof protocol for RFID tags: Security definition and scalable construction. *IACR Cryptology ePrint Archive*, 2009:609, 2009.

[DLYZ10] Robert H. Deng, Yingjiu Li, Moti Yung, and Yunlei Zhao. A new framework for RFID privacy. In *ESORICS*, volume 6345 of *LNCS*, pages 1–18. Springer, 2010.

[EBM11] Kaoutar Elkhiyaoui, Erik-Oliver Blass, and Refik Molva. Rotiv: RFID ownership transfer with issuer verification. In *RFIDSec*, volume 7055 of *LNCS*, pages 163–182. Springer, 2011.

[FA07] Sepideh Fouladgar and Hossam A. An efficient delegation and transfer of ownership protocol for RFID tags. In *First International EURASIP Workshop on RFID Technology*. ACM, 2007.

[FDW04] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *LNCS*, pages 357–370. Springer, 2004.

[FHV10] Junfeng Fan, Jens Hermans, and Frederik Vercauteren. On the claimed privacy of EC-RAC III. In Siddika Berna Ors Yalcin, editor, *RFIDSec*, volume 6370 of *LNCS*, pages 66–74. Springer, 2010.

[FMR99] Gerhard Frey, Michael Müller, and Hans-Georg Rück. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Transactions on Information Theory*, 45(5):1717–1719, 1999.

[FMTRCRDF11a] Albert Fernàndez-Mir, Rolando Trujillo-Rasua, Jordi Castellà-Roca, and Josep Domingo-Ferrer. A scalable RFID authentica-

tion protocol supporting ownership transfer and controlled delegation. In Ari Juels and Christof Paar, editors, *RFIDSec*, volume 7055 of *LNCS*, pages 147–162. Springer, 2011.

[FMTRCRDF11b] Albert Fernàndez-Mir, Rolando Trujillo-Rasua, Jordi Castellà-Roca, and Josep Domingo-Ferrer. A scalable RFID authentication protocol supporting ownership transfer and controlled delegation. In Ari Juels and Christof Paar, editors, *RFIDSec*, volume 7055 of *LNCS*, pages 147–162. Springer, 2011.

[GRS05] Henri Gilbert, Matthew J. B. Robshaw, and Hervé Sibert. An active attack against HB+ - A provably secure lightweight authentication protocol. *IACR Cryptology ePrint Archive*, 2005:237, 2005.

[GRS08] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. $HB^{\#}$: Increasing the security and efficiency of HB^{+}. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *LNCS*, pages 361–378. Springer, 2008.

[HB01] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *LNCS*, pages 52–66. Springer, 2001.

[HHPS11] Shai Halevi, Danny Harnik, Benny Pinkas, and Alexandra Shulman-Peleg. Proofs of ownership in remote storage systems. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*, pages 491–500. ACM, 2011.

[HK05] Gerhard P. Hancke and Markus G. Kuhn. An RFID distance bounding protocol. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks, SecureComm 2005, Athens, Greece, 5-9 September, 2005*, pages 67–73. IEEE, 2005.

[HK09] Hsieh-Hong Huang and Cheng-Yuan Ku. A RFID grouping proof protocol for medication safety of inpatient. *J. Medical Systems*, 33(6):467–474, 2009.

[HP12] Jens Hermans and Roel Peeters. Private yoking proofs: Attacks, models and new provable constructions. In Jaap-Henk Hoepman

and Ingrid Verbauwhede, editors, *RFIDSec*, LNCS, pages 96–108. Springer, 2012.

[HPP14] Jens Hermans, Roel Peeters, and Bart Preneel. Proper RFID privacy: Model and protocols. *IEEE Trans. Mob. Comput.*, 13(12):2888–2902, 2014.

[HPVP11] Jens Hermans, Andreas Pashalidis, Frederik Vercauteren, and Bart Preneel. A new RFID privacy model. In Vijay Atluri and Claudia Díaz, editors, *ESORICS*, volume 6879 of *LNCS*, pages 568–587. Springer, 2011.

[HWF08] Daniel M. Hein, Johannes Wolkerstorfer, and Norbert Felber. ECC is ready for RFID - a proof in silicon. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography*, volume 5381 of *LNCS*, pages 401–413. Springer, 2008.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In Ueli M. Maurer, editor, *EUROCRYPT*, volume 1070 of *LNCS*, pages 143–154. Springer, 1996.

[Jue04] Ari Juels. "yoking-proofs" for RFID tags. In *PerCom Workshops*, pages 138–143. IEEE Computer Society, 2004.

[Jue06] Ari Juels. RFID security and privacy: a research survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, 2006.

[JW05] Ari Juels and Stephen A. Weis. Authenticating pervasive devices with human protocols. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *LNCS*, pages 293–308. Springer, 2005.

[JW07] Ari Juels and Stephen A. Weis. Defining strong privacy for RFID. In *PerCom Workshops*, pages 342–347. IEEE Computer Society, 2007.

[Khe13] Walid I. Khedr. SRFID: A hash-based security scheme for low cost RFID systems. *Egyptian Informatics Journal*, 14(1):89 – 98, 2013.

[KP12] Gaurav Kapoor and Selwyn Piramuthu. Single RFID tag ownership transfer protocols. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 42(2):164–173, 2012.

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS*, 2000.

[KSLE09] Selim Volkan Kaya, Erkay Savas, Albert Levi, and Özgür Erçetin. Public key cryptography based privacy preserving multi-context RFID infrastructure. *Ad Hoc Networks*, 7(1):136–152, 2009.

[KW05] Ziv Kfir and Avishai Wool. Picking virtual pockets using relay attacks on contactless smartcard. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks, SecureComm 2005, Athens, Greece, 5-9 September, 2005*, pages 47–58. IEEE, 2005.

[KYWG10] Lars Kulseng, Zhen Yu, Yawen Wei, and Yong Guan. Lightweight mutual authentication and ownership transfer for RFID systems. In *INFOCOM*, pages 251–255. IEEE, 2010.

[KZP11] Gaurav Kapoor, Wei Zhou, and Selwyn Piramuthu. Multi-tag and multi-owner RFID ownership transfer in supply chains. *Decision Support Systems*, 52(1):258–270, 2011.

[LBSV10a] Yong Ki Lee, Lejla Batina, Dave Singelée, and Ingrid Verbauwhede. Low-cost untraceable authentication protocols for RFID. In Susanne Wetzel, Cristina Nita-Rotaru, and Frank Stajano, editors, *WISEC*, pages 55–64. ACM, 2010.

[LBSV10b] Yong Ki Lee, Lejla Batina, Dave Singelée, and Ingrid Verbauwhede. Wide-weak privacy-preserving RFID authentication protocols. In Periklis Chatzimisios, Christos V. Verikoukis, Ignacio Santamaría, Massimiliano Laddomada, and Oliver Hoffmann, editors, *MOBILIGHT*, volume 45 of *LNCS, Social Informatics and Telecommunications Engineering*, pages 254–267. Springer, 2010.

[LBV08] Yong Ki Lee, L. Batina, and I. Verbauwhede. EC-RAC (ECDLP based randomized access control): Provably secure RFID authentication protocol. In *RFID, 2008 IEEE International Conference on*, pages 97 –104, 2008.

[LBV09] Yong Ki Lee, L. Batina, and I. Verbauwhede. Untraceable RFID authentication protocols: Revision of EC-RAC. In *RFID, 2009 IEEE International Conference on*, pages 178 –185, 2009.

[LDL10] Junzuo Lai, Robert H. Deng, and Yingjiu Li. Revisiting unpredictability-based RFID privacy models. In Jianying Zhou and Moti Yung, editors, *ACNS*, volume 6123 of *LNCS*, pages 475–492, 2010.

[Lee10] Kaleb Lee. A two-step mutual authentication protocol based on randomized hash-lock for small RFID networks. In Yang Xiang, Pierangela Samarati, Jiankun Hu, Wanlei Zhou, and Ahmad-Reza Sadeghi, editors, *Fourth International Conference on Network and System Security, NSS 2010, Melbourne, Victoria, Australia, September 1-3, 2010*, pages 527–533. IEEE Computer Society, 2010.

[LH13] Yi-Pin Liao and Chih-Ming Hsiao. A secure ECC-based RFID authentication scheme integrated with id-verifier transfer protocol. *Ad Hoc Networks*, page http://dx.doi.org/10.1016/j.adhoc.2013.02.004, 2013.

[LIM]

[LK06a] Chae Hoon Lim and Tymur Korkishko. mcrypton a lightweight block cipher for security of low-cost RFID tags and sensors. In Joo-Seok Song, Taekyoung Kwon, and Moti Yung, editors, *Information Security Applications*, volume 3786 of *LNCS*, pages 243–258. Springer, 2006.

[LK06b] Chae Hoon Lim and Taekyoung Kwon. Strong and robust RFID authentication enabling perfect ownership transfer. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *ICICS*, volume 4307 of *LNCS*, pages 1–20. Springer, 2006.

[LLM+09] Xuefei Leng, Yuanhung Lien, Keith Mayes, Konstantinos Markantonakis, and Jung-Hui Chiu. Select-response grouping proof for RFID tags. In Ngoc Thanh Nguyen, Huynh Phan Nguyen, and Adam Grzech, editors, *First Asian Conference on Intelligent Information and Database Systems, ACIIDS 2009, Dong hoi, Quang binh, Vietnam, April 1-3, 2009*, pages 73–77. IEEE Computer Society, 2009.

[LLMC08] Yuanhung Lien, Xuefei Leng, Keith Mayes, and Jung-Hui Chiu. Reading order independent grouping proof for RFID tags. In *IEEE International Conference on Intelligence and Security Informatics, ISI 2008, Taipei, Taiwan, June 17-20, 2008, Proceedings*, pages 128–136. IEEE, 2008.

[LMS+13] Nan Li, Yi Mu, Willy Susilo, Fuchun Guo, and Vijay Varadharajan. On RFID authentication protocols with wide-strong privacy. In Changshe Ma and Jian Weng, editors, *RFIDsec'13 Asia*, volume 11 of *Cryptology and Inforamtion Security Series*, pages 3–16. IOS Press, 2013.

[LMS+14] Nan Li, Yi Mu, Willy Susilo, Fuchun Guo, and Vijay Varadharajan. Privacy-preserving authorized RFID authentication protocols. In Nitesh Saxena and Ahmad-Reza Sadeghi, editors, *Radio Frequency Identification: Security and Privacy Issues - 10th International Workshop, RFIDSec 2014, Oxford, UK, July 21-23, 2014, Revised Selected Papers*, volume 8651 of *LNCS*, pages 108–122. Springer, 2014.

[LMS+15] Nan Li, Yi Mu, Willy Susilo, Fuchun Guo, and Vijay Varadharajan. Vulnerabilities of an ECC-based RFID authentication scheme. *Security and Communication Networks*, 2015.

[LMSV13] Nan Li, Yi Mu, Willy Susilo, and Vijay Varadharajan. Secure RFID ownership transfer protocols. In RobertH. Deng and Tao Feng, editors, *ISPEC*, volume 7863 of *LNCS*, pages 189–203. Springer Berlin Heidelberg, 2013.

[LMSV15a] Nan Li, Yi Mu, Willy Susilo, and Vijay Varadharajan. Anonymous yoking-group proofs. In Feng Bao, Steven Miller, Jianying Zhou, and Gail-Joon Ahn, editors, *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '15, Singapore, April 14-17, 2015*, pages 615–620. ACM, 2015.

[LMSV15b] Nan Li, Yi Mu, Willy Susilo, and Vijay Varadharajan. Shared RFID ownership transfer protocols. *Computer Standards & Interfaces*, 42(0):95 – 104, 2015.

[LNZ+13] Hong Liu, Huansheng Ning, Yan Zhang, Daojing He, Qingxu Xiong, and Laurence T. Yang. Grouping-proofs-based authen-

tication protocol for distributed RFID systems. *IEEE Trans. Parallel Distrib. Syst.*, 24(7):1321–1330, 2013.

[LOIM09] Mikko Lehtonen, Daniel Ostojic, Alexander Ilic, and Florian Michahelles. Securing RFID systems by detecting tag cloning. In Hideyuki Tokuda, Michael Beigl, Adrian Friday, A. J. Bernheim Brush, and Yoshito Tobe, editors, *Pervasive Computing, 7th International Conference, Pervasive 2009, Nara, Japan, May 11-14, 2009. Proceedings*, volume 5538 of *LNCS*, pages 291–308. Springer, 2009.

[LSBV08] Yong Ki Lee, Kazuo Sakiyama, Lejla Batina, and Ingrid Verbauwhede. Elliptic-curve-based security processor for RFID. *IEEE Trans. Computers*, 57(11):1514–1527, 2008.

[LWD08] Tieyan Li, Guilin Wang, and Robert H. Deng. Security analysis on a family of ultra-lightweight RFID authentication protocols. *JSW*, 3(3):1–10, 2008.

[Mer89] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *LNCS*, pages 218–238. Springer, 1989.

[MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[MSW05] David Molnar, Andrea Soppera, and David Wagner. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In Bart Preneel and Stafford E. Tavares, editors, *SAC*, volume 3897 of *LNCS*, pages 276–290. Springer, 2005.

[MV10] Roel Maes and Ingrid Verbauwhede. Physically unclonable functions: A study on the state of the art and future research directions. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security - Foundations and Practice*, Information Security and Cryptography, pages 3–37. Springer, 2010.

[NSMSN08] Ching Yu Ng, Willy Susilo, Yi Mu, and Reihaneh Safavi-Naini. RFID privacy models revisited. In *ESORICS*, volume 5283 of *LNCS*, pages 251–266. Springer, 2008.

[NSMSN11] Ching Yu Ng, Willy Susilo, Yi Mu, and Reihaneh Safavi-Naini. Practical RFID ownership transfer scheme. *Journal of Computer Security*, 19(2):319–341, 2011.

[NTU10] Rishab Nithyanand, Gene Tsudik, and Ersin Uzun. Readers behaving badly - reader revocation in PKI-based RFID systems. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *ESORICS*, volume 6345 of *LNCS*, pages 19–36. Springer, 2010.

[NTU11] Rishab Nithyanand, Gene Tsudik, and Ersin Uzun. User-aided reader revocation in PKI-based RFID systems. *Journal of Computer Security*, 19(6):1147–1172, 2011.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43. ACM, 1989.

[OF09] Yossef Oren and Martin Feldhofer. A low-resource public-key identification scheme for RFID tags and sensor nodes. In David A. Basin, Srdjan Capkun, and Wenke Lee, editors, *WISEC*, pages 59–68. ACM, 2009.

[One12] Cristina Onete. Key updates for RFID distance-bounding protocols: Achieving narrow-destructive privacy. *IACR Cryptology ePrint Archive*, 2012:165, 2012.

[OSK03] Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic approach to "privacy-friendly tags. In *RFID Privacy Workshop*, 2003.

[OSK04] Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Efficient hash-chain based RFID privacy protection scheme. In *International Conference on Ubiquitous Computing Ubicomp, Workshop Privacy: Current Status and Future Directions*, 2004.

[OTYT06] Kyosuke Osaka, Tsuyoshi Takagi, Kenichi Yamazaki, and Osamu Takahashi. An efficient and secure RFID security method with ownership transfer. In Yuping Wang, Yiu ming Cheung, and

Hailin Liu, editors, *CIS*, volume 4456 of *LNCS*, pages 778–787. Springer, 2006.

[PH12] Roel Peeters and Jens Hermans. Wide strong private RFID identification based on zero-knowledge. *IACR Cryptology ePrint Archive*, 2012:389, 2012.

[PH13] Roel Peeters and Jens Hermans. Attack on Liao and Hsiao's secure ECC-based RFID authentication scheme integrated with id-verifier transfer protocol. *IACR Cryptology ePrint Archive*, 2013:399, 2013.

[PHER07] Pedro Peris-Lopez, Julio C. Hernandez-Castro, Juan M. Estévez-Tapiador, and Arturo Ribagorda. Solving the simultaneous scanning problem anonymously: Clumping proofs for RFID tags. In Panagiotis Georgiadis, Javier Lopez, Stefanos Gritzalis, and Giannis F. Marias, editors, *Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, SECPerU 2007, Istanbul, Turkey, July 19, 2007*, pages 55–60. IEEE Computer Society, 2007.

[PHF13] Roel Peeters, Jens Hermans, and Junfeng Fan. IBIHOP: Proper privacy preserving mutual RFID authentication. In Changshe Ma and Jian Weng, editors, *RFIDsec'13 Asia*, volume 11 of *Cryptology and Inforamtion Security Series*, pages 3–16. IOS Press, 2013.

[Pir06] Selwyn Piramuthu. On existence proofs for multiple RFID tags. In *Proceedings of the ACS/IEEE International Conference on Pervasive Services 2006, ICPS '06, 26-29 June 2006, Lyon, France*, pages 317–320. IEEE Computer Society, 2006.

[Pir11] Selwyn Piramuthu. RFID mutual authentication protocols. *Decision Support Systems*, 50(2):387–393, 2011.

[PLHCT+10] Pedro Peris-Lopez, Julio C. Hernandez-Castro, Juan E. Tapiador, Tieyan Li, and Yingjiu Li. Vulnerability analysis of RFID protocols for tag ownership transfer. *Computer Networks*, 54(9):1502–1508, 2010.

[PLOHCvdL11] Pedro Peris-Lopez, Agustín Orfila, Julio C. Hernandez-Castro, and Jan C. A. van der Lubbe. Flaws on RFID grouping-proofs.

guidelines for future sound protocols. *J. Network and Computer Applications*, 34(3):833–845, 2011.

[PPH11] Christian Pendl, Markus Pelnar, and Michael Hutter. Elliptic curve cryptography on the WISP UHF RFID tag. In *RFID. Security and Privacy - 7th International Workshop, RFIDSec 2011, Amherst, USA, June 26-28, 2011, Revised Selected Papers*, volume 7055 of *LNCS*, pages 32–47. Springer, 2011.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Advances in Cryptology - Proceedings of EUROCRYPT '96*, volume 1070 of *LNCS*, pages 387–398. Springer, 1996.

[PV08] Radu-Ioan Paise and Serge Vaudenay. Mutual authentication in RFID: security and privacy. In Masayuki Abe and Virgil D. Gligor, editors, *ASIACCS*, pages 292–299. ACM, 2008.

[RG12] Panagiotis Rizomiliotis and Stefanos Gritzalis. GHB #: A provably secure HB-like lightweight authentication protocol. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *ACNS*, volume 7341 of *LNCS*, pages 489–506. Springer, 2012.

[Riv92] R. Rivest. The MD5 message-digest algorithm. RFC 1321, 1992.

[RRG09] Panagiotis Rizomiliotis, Evangelos Rekleitis, and Stefanos Gritzalis. Security analysis of the song-mitchell authentication protocol for low-cost RFID tags. *Comm. Letters.*, 13(4):274–276, 2009.

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *LNCS*, pages 371–388. Springer, 2004.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *LNCS*, pages 239–252. Springer, 1989.

[SD07] G. Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th Design Automation Conference, DAC 2007,*

San Diego, CA, USA, June 4-8, 2007, volume 3860 of *LNCS*, pages 9–14. Springer, 2007.

[Sha08] Adi Shamir. SQUASH - a new MAC with provable security properties for highly constrained devices such as RFID tags. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *LNCS*, pages 144–157. Springer, 2008.

[SIS05] 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, *EUC Workshops*, volume 3823 of *LNCS*, pages 1303–1312. Springer, 2005.

[SM08] Boyeon Song and Chris J. Mitchell. RFID authentication protocol for low-cost tags. In Virgil D. Gligor, Jean-Pierre Hubaux, and Radha Poovendran, editors, *WISEC*, pages 140–147. ACM, 2008.

[Son08] Boyeon Song. RFID tag ownership transfer. In *4th Workshop on RFID Security - RFIDSec*, 2008.

[SS05] Junichiro Saito and Kouichi Sakurai. Grouping proof for RFID tags. In *AINA*, pages 621–624. IEEE Computer Society, 2005.

[SSAQ02] David Samyde, Sergei P. Skorobogatov, Ross J. Anderson, and Jean-Jacques Quisquater. On a new way to read data from memory. In *IEEE Security in Storage Workshop*, pages 65–69, 2002.

[TB06] Pim Tuyls and Lejla Batina. RFID-tags for anti-counterfeiting. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *LNCS*, pages 115–131. Springer, 2006.

[Tsu06] Gene Tsudik. Ya-trap: Yet another trivial RFID authentication protocol. In *PerCom Workshops*, pages 640–643. IEEE Computer Society, 2006.

[Vau07] Serge Vaudenay. On privacy models for RFID. In *ASIACRYPT*, volume 4833 of *LNCS*, pages 68–87. Springer, 2007.

[Vau10] Serge Vaudenay. Privacy models for RFID schemes. In *RFIDSec*, volume 6370 of *LNCS*, page 65. Springer, 2010.

[vDMRV09] 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, *ESORICS*, volume 5789 of *LNCS*, pages 637–654. Springer, 2009.

[vDR08] Ton van Deursen and Saša Radomirović. Attacks on RFID protocols. *IACR Cryptology ePrint Archive*, 2008:310, 2008.

[vDR09] Ton van Deursen and Saša Radomirović. Untraceable RFID protocols are not trivially composable: Attacks on the rvision of EC-RAC. *IACR Cryptology ePrint Archive*, 2009:332, 2009.

[vDR10] Ton van Deursen and Saša Radomirović. EC-RAC: Enriching a capacious RFID attack collection. In Siddika Berna Ors Yalcin, editor, *RFIDSec*, volume 6370 of *LNCS*, pages 75–90. Springer, 2010.

[vDR11] Ton van Deursen and Saša Radomirović. Insider attacks and privacy of RFID protocols. In Svetla Petkova-Nikova, Andreas Pashalidis, and Günther Pernul, editors, *EuroPKI*, volume 7163 of *LNCS*, pages 91–105. Springer, 2011.

[Wei00] Steve H. Weingart. Physical security devices for computer subsystems: A survey of attacks and defences. In Çetin Kaya Koç and Christof Paar, editors, *CHES*, volume 1965 of *LNCS*, pages 302–317. Springer, 2000.

[WLHL07] Weijia Wang, Yong Li, Lei Hu, and Li Lu. Storage-awareness: RFID private authentication based on sparse tree. In Panagiotis Georgiadis, Javier Lopez, Stefanos Gritzalis, and Giannis F. Marias, editors, *Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, SECPerU 2007, Istanbul, Turkey, July 19, 2007*, pages 61–66. IEEE Computer Society, 2007.

[WSRE03] 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, *SPC*, volume 2802 of *LNCS*, pages 201–212. Springer, 2003.

[ZCJ13] Davide Zanetti, Srdjan Capkun, and Ari Juels. Tailing RFID tags for clone detection. In *20th Annual Network and Distributed*

System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013. The Internet Society, 2013.

[ZPS92] Yuliang Zheng, Josef Pieprzyk, and Jennifer Seberry. HAVAL - a one-way hashing algorithm with variable length of output. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT*, volume 718 of *LNCS*, pages 83–104. Springer, 1992.

[ZZLW10] Shijie Zhou, Zhen Zhang, Zongwei Luo, and Edward C. Wong. A lightweight anti-desynchronization RFID authentication protocol. *Information Systems Frontiers*, 12(5):521–528, 2010.

Index