Security and Privacy in RFID-Enabled Supply Chains

by

Shaoying Cai

Submitted to School of Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Information Systems

Dissertation Committee:

Yingjiu LI (Supervisor / Chair)
Associate Professor of Information Systems
Singapore Management University

Robert DENG Huijie (Co-supervisor)
Professor of Information Systems
Singapore Management University

Xuhua DING
Associate Professor of Information Systems
Singapore Management University

Jianying ZHOU
Head of Infocomm Security Department
Institute for Infocomm Research

Singapore Management University
2014

Security and Privacy in RFID-Enabled Supply Chains

Shaoying Cai

Abstract

Supply chain is a network involving multiple parties such as suppliers, transporters, storage facilities, distributors, and retailers that participate in the production, delivery, and sale of a product. It is difficult to monitor a supply chain since the involving parties are distributed at multiple locations or even across countries. RFID technology, when combined with networking technology, enables product information to be collected, integrated, shared, and queried in supply chains at various levels (e.g., item, pallet, case, and container) in real time manner.

While RFID technology has greatly facilitated the supply chain management, it is still challenging to design a secure, privacy-preserving, and efficient RFID-enabled supply chain system. The wireless communications between RFID readers and tags are subject to a variety of attacks. An adversary may eavesdrop, replay, and manipulate RFID communications to obtain tag identifier, track tag location, impersonate tag and reader, and trigger denial of service. This dissertation focuses on secure and privacy-preserving tag authentication in various supply chain application scenarios.

Our first work is on attacks and improvements of an existing mutual authentication protocol and a tag secret update protocol for RFID-enabled supply chains. Our second work improves the efficiency of an RFID-enabled supply chain system by designing the system in two security modes. In the weak security mode, the tagged products can be processed in a highly efficient way. In the strong security mode, our system guarantees a high level of security, while its efficiency is lower than that in the weak security mode. Our third work addresses the tag authentication problem in the scenario of third-party logistics(3PL). We firstly formalize the security

and privacy requirements of RFID systems for 3PL supply chains considering the existence of the internal adversaries as well as the external adversaries. We propose two different protocols, one is based on aggregate message authentication codes, the other is based on aggregate signature scheme. Our solutions enable a third-party to check tag existence without knowing tag secrets. Our fourth work focuses on path authentication in RFID-enabled supply chains. We propose a single-game-based privacy notion for RFID-enabled path authentication which has been proven to be stronger than existing privacy notions for path authentication. We also propose t-wo new path authentication schemes, one for closed supply chains, and another for dynamic supply chains.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This dissertation would not have been possible without the help of many people.

Firstly, my utmost gratitude goes to my advisors Professor Yingjiu Li and Professor Robert Deng. I appreciate all their time and efforts in training me to become a better researcher. Their enthusiasms in research will always encourage me. And their scientific rigor will always serve as an example to me.

Secondly, I would like to express my gratitude to other dissertation commitee members: Professor Xuhua Ding and Dr. Jianying Zhou for their valuable comments on improving my dissertation.

I am also thankful to my collaborators: Prof. Yunlei Zhao, Dr. Tieyan Li, Dr. Chunhua Su and Dr. Changshe Ma. I have learned a lot from them through the collaborations. Their contributions have become an indispensable part of my dissertation.

I would like to thank my labmates and friends: Cane LEUNG Wing Ki, Lucia, Li Yan, Lo Swee Won, Yazhe Wang, Jundi Ding, Zhiyang Yu, Siyang Xiong, Shi Jie and many others, for the joyful time we have spent together.

Finally, I would like to express my appreciation to my parents for their unconditional love.

Dedication

I dedicate my dissertation to my parents.

Chapter 1

Introduction

RFID technology has been widely envisioned to have significant impact on the economy world-wide as an inevitable replacement of barcodes in the near future. With RFID technology, product information can be efficiently collected, tracked, shared, and managed in a real time manner. This helps reduce the overall cost of supply chain management and benefits all involved parties [43].

A typical RFID system consists of tags, readers, and a back-end server. An RFID tag is a radio transponder that is composed of an integrated circuit for storing and processing identification information, as well as an antenna for communicating with RFID readers. An RFID reader is a radio transceiver that can be used to query RFID tags through wireless channel for the identifying information about the objects to which the tags are attached [36]. An RFID reader is usually connected to a back-end server via a secure channel. The back-end server has a database of tags that can be used to retrieve detailed information about a tag according to the tag's response.

Like a double-edged sword, RFID technology has also triggered significant security and privacy issues. An RFID reader and an RFID tag communicate via wireless radios which can be eavesdropped and manipulated without the tag owner's concern. Also, due to the limited storage and computational power [6], a low-cost tag could be compromised by an adversary (e.g., via side-channel attack [4]), and as

a result, all the contents stored on the tag will be revealed. The weaknesses of RFID systems may result in inventory information leakage and tag spoofing/cloing, and cause significant loss to supply chain partners. A lot of RFID reader-tag communication protocols have been proposed, however, only few of them are designed for supply chain applications. Meanwhile, different supply chain application scenarios will impose different sets of security requirements and require different treatments to design tag-reader communication protocols. Existing solutions cannot cover all the supply chain application scenarios. In this thesis, we focus on the security and privacy issues in a variety of RFID-enabled supply chains.

1.1 Problem Statement and Contributions

1.1.1 Mutual Authentication

We first consider the security and privacy issues in generic supply chains. Generic supply chains refer to supply chains that meet the following criteria: 1) the partners are well-connected so that any parter can transfer bulky information about products to its downstream partners; 2) all the partners can connect to the same trusted authority (if there is a trusted authority involved in the supply chain management).

Figure 1.1 illustrates a sectional view of two interacting partners within a generic supply chain. Only minimum amount of information such as product IDs and lightweight security primitives is stored in RFID tags. Bulky data about product details are stored in savant computers. When partner A hands over a batch of products to partner B, the information about the products will be transferred to partner B as well.

Visibility, efficiency, and security are three basic requirements for RFID-based supply chain management systems. The manager of a supply chain or any authorized party should be able to track the movement of RFID tags. Also, RFID readers should be able to identify/authenticate RFID tags in a high speed to cater for the

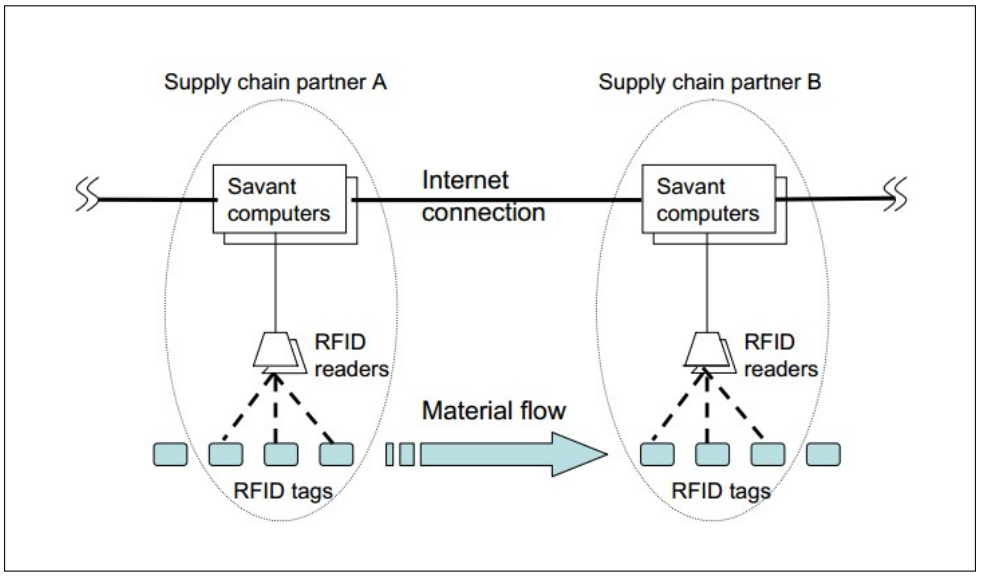

Figure 1.1: Sectional View of Generic Supply Chains [46]

high throughput property of supply chains. Besides, any adversary should not be able to collect information about tagged products by probing the tags, update tag contents, etc. The security requirements of RFID-tagged supply chains are summarized in [43]. We increase the requirements to include forward and backward secrecy and de-synchronization resilience.

Though many solutions have been proposed to protect RFID communications (see section 2.3.1 for more details about the related work), few of them address all of the security and privacy concerns. Our goal is to propose secure and privacy-preserving tag-reader communication protocols to support high-level supply chain visibility, efficiency, and security.

We first analyze two typical protocols [61, 60] that are asserted to have the most desired security properties for RFID communications. We discover that these protocols are vulnerable to a series of active attacks including server impersonation, tag impersonation, and de-synchronization. We propose the revised protocols to eliminate the vulnerabilities without violating any other security properties. The storage and computational requirements are comparable to the existing solutions. The revised protocols could leverage on the standard EPCglobal network to share information about products. Each partner could access the information shared by

other partners. Supply viability is provided in a distributed way.

Then we propose a new solution to balance the concerns on security, visibility and efficiency. Our solution requires a trust authority to handle tag ownership transfers. Supply viability is achieved in a centralised way. In order to enhance the efficiency of an RFID-enabled supply chain system without sacrificing its security and privacy, we distinguish the environments into two security levels. In a relatively secure environment with no active attacks, our RFID system can be set to the weak security mode so as to provide a high processing speed. While in a relatively less secure environment that is exposed to active attacks, our RFID system can be switched to the strong security mode so as to maintain strong unlinkability.

1.1.2 Three Party Authentication

Third party logistics (3PL) is one of the most dominating kind of supply chains, it has been widely adopted by many companies. The companies outsources part or all of their supply chains to professional logistics service provider to get better management efficiency and at same time reduce the cost. The outsourcing introduces new security and privacy challenges, since the involving parties, including the sender, the receiver, and the third party, may not be all credible. For example, a third party may steal some goods, and claim that the sender did not send sufficient goods.

Dozens of cryptographic protocols have been proposed to provide secure and private identification and authentication of the tag (Sometimes the reader authentication is also required). There are also many works deal with the secure and private ownership transfer between two parties. Most of the solutions for authentication and ownership transfer are “symmetric secret”-based that an authorized reader shares a secret with each tag.

The “symmetric secret”-based solutions are designed to protect the system against external adversaries who do not know the secrets. They may work well in generic supply chains. However, in 3PL supply chains that three parties (the sender

of the goods, the receiver of the goods and the 3PL provider) are involved in the processing of the tags, internal adversaries should be considered. In 3PL supply chains, all of the three parties need to access the tags, hence all of them should have a copy of tag secrets when a "symmetric secret"-based solution is deployed. With a tag's secret, any party can fabricate the tag. Disputes on the goods' originality will be hard to solve as all the three parties have the ability to fabricate the tags.

Currently, there does not exist any solution that is suitable for 3PL supply chains considering the exitance of internal adversaries. It does not mean that putting effort on 3PL supply chains is not necessary. 3PL has large market size, a study [1] shows that in U.S. the 3PL market gross revenues already reached $107.1 billion in 2009. It is crucial to enhance the security and privacy level of RFID-enabled system for 3PL supply chains. We are the first ones to work on this new direction. Our contributions can be summarized as follows:

- We firstly formulate the security and privacy requirements of RFID system for 3PL supply chains with respect to both the internal and external adversaries.
- To execute the authentication of the tags in 3PL supply chains without revealing the secrets to the 3PL provider and the receiver of the goods, we provide two solutions that enable the tags' aggregate authentication on batch level. One solution is based on an aggregate Message Authentication Code(MAC), the other is based on an aggregate signature scheme.

Both the two solutions match the privacy and security requirements of 3PL supply chains. The comparisons on performance and usability between the two proposals show that the aggregate MAC-based solution is more applicable than the aggregate signature-based solution in 3PL supply chains.

[1] http://www.prnewswire.com/news-releases/us-and-global-third-party-logistics-market-analysis-is-released-94771894.html

1.1.3 Path Authentication

Various tag authentication schemes have been proposed to enable privacy-preserving identification of tags.[2] However, most of proposals require tags to have certain computational capability, which may incur unbearable cost in practice. Currently, most prevalent tags on the market are standard EPCglobal C1 G2 tags [1, 3], which only has several hundred bits storage and no computational ability.

Another common problem of deploying existing solutions in supply chain is that: to monitor a supply chain, the manager should have access to the databases of all the entities in the supply chain. This requires high-quality network connection and fine-grained access control, which may not be realistic in practice.

Recently, RFID-enabled path authentication was proposed by Blass, Elkhiyaoui and Molva [8, 7], and extended later to be more practical [64], to tackle the counterfeiting problem in supply chains. In the proposal, which is named as TRACKER, the manager of a supply chain verifies the genuineness of a tag by checking whether it has been processed by a series of reliable readers. Compared to the existing "symmetric secret"-based tag authentication schemes, in TRACKER, the verification of a tag's genuineness is merely based on the credentials stored on the tag which are generated by the readers that have processed the tag. TRACKER can be implemented with standard EPCglobal C1 G2 tags. It also does not require the entities in the supply chain to have any connection except in the initial stage.

Path authentication schemes like TRACKER are extremely practical. Following Blass, Elkhiyaoui and Molva, we continue this line of research. In this thesis, path authentication schemes specifically refer to the ones that satisfy the following criteria: 1) the verification of a tag's genuineness is merely based on the contents stored on the tag; 2) it does not require supply chain partners to have any connection (except in the initial stage, if needed).

Our contributions include:

[2] Most of the existing tag authentication schemes and their extensions are listed on http://www.avoine.net/rfid/.

- We analyze the existing security and privacy notions for path authentication in RFID-enabled supply chain, including tag unlinkability and step unlinkability [8, 7]. We show that these two notions can be further refined to be more concise and formal.

- We propose a combined privacy notion that considers both tag unlinkability and step unlinkability for RFID-enabled supply chains. We analyze the relations among our new privacy notion, the tag unlinkability notion and the step unlinkability notion. We prove that our privacy notion implies tag unlinkability and step unlinkability.

- We propose a new path authentication solution for closed supply chain using the standard EPC Class 1 Gen 2 tags without sharing the secret among supply chain parties. In closed supply chains, supply chain managers are assumed to know all the tags' paths before they enter the supply chain. Our solution does not require the partners in the supply chain to have any connection except in the initial stage. Compare to TRACKER, our solution is more efficient and requires less storage. We prove that our solution satisfies the refined security notion proposed in [8, 7] and our new privacy notion.

- We propose a distributed path authentication solution for dynamic RFID-enabled supply chains to address the counterfeiting problem. Compared to existing general anti-counterfeiting solutions, our solution requires non sharing of item-level RFID information among supply chain parties, thus eliminating the requirement on high network bandwidth and fine-grained access control. Our solution is secure, privacy-preserving, and practical. It leverages on the standard EPCglobal network to share information about paths and parties in path authentication. Our solution can be implemented on standard EPC class 1 generation 2 tags with only 720 bits storage and no computational capability.

1.2 Organization of the Thesis

In Chapter 2, we show some backgrounds. We introduce the security and privacy concerns in different supply chain settings. We also provide literature reviews.

In Chapter 3, we demonstrate a server impersonation attack, a tag impersonation attack, and a de-synchronization attack to two protocols proposed in [61, 60]. We also propose the revised protocols to eliminate the vulnerabilities without violating any other security properties.

In Chapter 4, we propose a dual-mode protocol to balance the concerns on security, privacy and efficiency.

In Chapter 5, we formulate the security and privacy requirements for 3PL supply chains and provide two solutions for authenticating a batch of tags without using tag secrets.

In Chapter 6, we study the path authentication problem in RFID-enabled supply chains. We propose a new privacy notion, and prove that it implies tag unlinkability and step unlinkability proposed in [7]. We also propose two new path authentication schemes, one for closed supply chains, the other for dynamic supply chains.

Finally, we conclude this dissertation in Chapter 7. We summarize the work we have accomplished, and point out possible directions for future research.

1.3 Publications

This thesis contains contents that have been published in [16, 15, 17, 19, 18, 14].

The contents of [16] form the basis for Chapter 3. The contents of [15, 17] form the basis for Chapter 4. The contents of [19] form the basis for Chapter 5. And the contents of [18, 14] form the basis for Chapter 6.

Chapter 2

Backgrounds

In this chapter, we first introduce four types of supply chains. Then we summarize the security and privacy concerns on three settings: mutual authentication, three party authentication, and path authentication. We provide a literature review on the researches in each setting respectively.

2.1 RFID-Enabled Supply Chain Management Systems

There are four types of supply chains: third-party logistics (3PL), vendor managed inventory (VMI), collaborative planning, forecasting, and replenishment (CPFR), and supply network (SN) [47]. 3PL enables companies to concentrate on their core competencies and outsource shipping to other parties. In VMI, the vendors take over the replenishment planning tasks for their trading partners. CPFR is a collaborative arrangement of multiple parties for real-time sharing of demand and supply data. SN is similar to CPFR, however, with more complex collaborative processes and more ad hoc information flows. These structures are defined based on the collaboration relationships among supply chain partners [43].

RFID-enabled supply chain management systems contain many components, e.g., RFID tag reading sub-system, ownership transfer sub-system, inventory man-

agement sub-system, order management sub-system, and product information sharing sub-system, etc. RFID tag reading sub-systems and product information sharing sub-systems generally are considered open to external entities. Any person in the working range of tags can probe the tags without the owner's concern. And product information sharing sub-systems provide portals to external parties for querying information. Hence, both of RFID tag reading sub-systems and product information sharing sub-systems are potentially vulnerable to adversaries. And improper ownership transfer sub-systems may violate supply chain partners' privacy. Currently, regarding security, researchers' concerns are concentrated on tag reading sub-systems, ownership transfer sub-systems, and product information sharing sub-systems. In this thesis, we focus on building secure tag reading sub-systems and ownership transfer sub-systems. Mutual authentication protocols, tag ownership transfer protocols, and path authentication protocols are components of tag reading sub-systems and ownership transfer sub-systems in supply chain management systems. The readers are referred to [42] for more information about researches on the security of product information sharing sub-system .

2.2 Security and Privacy Concerns in RFID-Enabled Supply Chains

2.2.1 Mutual Authentication

Mutual authentication protocols for generic supply chains are applicable to VMI, CPFR, and SN, when the supply chain partners are well connected and can share bulky information of the tagged products. The security requirements of RFID-tagged supply chains are summarized in [43].

We increase the requirements to include forward and backward secrecy and de-synchronization resilience. The list of security requirements are given below:

- Authoritative access: Only legitimate readers of an authorized party are al-

lowed to identify and update a tag.

- Authenticity of tags: In a supply chain link, only legitimate RFID tags delivered by previous party will be accepted by the next party.

- Unlinkability: Weak unlinkability and strong unlinkability can be used to describe the security level of anti-tracking. Weak unlinkability requires that an unauthorized reader cannot link the responses of a tag interrogated before and after it is processed by an authorized party. Strong unlinkability requires that an unauthorized reader cannot link any two replies to the same tag [43].

- Forward and backward secrecy: If the communication between a tag and a party is compromised, it will not affect the security of the communication between the tag and any other party in the supply chain. The forward security means that the compromise of the current communication will not affect the security of previous communications [63], while the backward security means that the compromise of the current communication will not affect the security of later communications [13].

- De-synchronization resilience: The de-synchronization attacks are launched to disrupt the reader-tag update process so that the contents in the tag do not match with any record in the back-end database; consequently, the reader cannot identify or authenticate the tag [41].

Secure ownership transfer is another concern in RFID-enabled supply chains. Below are the security requirements for tag ownership transfer identified in [25, 44].

- New Owner Privacy: Once a tag has been transferred to a new owner, only the new owner should be able to identify and control the tag. The previous owner of the tag should no longer be able to identify or trace the tag.

- Old Owner Privacy: Once a tag has been transferred to a new owner, the new owner of the tag should not be able to trace past interactions between the tag and its previous owner.

- Authorisation Recovery: In some special cases, the previous owner of a tag might need to temporarily recover the ability to interact with it. The current owner should be able to transfer the tag's ownership to its previous owner.

2.2.2 Three Party Authentication

We depict the model of 3PL supply chains in Figure 2.1. A 3PL supply chain contains three parties. We denote the sender (customer A) who entrusts the transportation of goods to a 3PL provider as Party A, the 3PL provider as Party C and the receiver (customer B) of the goods as Party B [1]. The procedures in 3PL supply chains contain three steps:

1. *Ownership transfers from Party A to Party C*: Party A transfers the goods to Party C after the three parties have reached an agreement of the transaction.

2. *Party C' transports the goods*: Party C takes over the goods, and guarantees the goods' security during the transportation.

3. *Ownership transfers from Party C to Party B*: Party B verifies the goods, accepts them if the goods are intact, or denies the goods if the goods are not satisfactory.

A successful transaction is finished after Party B accepts the goods. Traditionally, when a party transfers goods to another party, the originality and the quantity of the goods are checked manually. However, when RFID system is deployed to enhance the efficiency of the supply chain, automatic identification replaces the manually checking. In RFID-enabled supply chains, the existence of the tags indicates the existence of the original goods[2].

[1] Party A and Party B can be the same entity in some occasions, eg. a factory entrusts a 3PL provider to transmit a batch of goods to its branch plant.

[2] Suppose each tag is imbedded in or stick on one item and it is hard to separate the tag from the item.

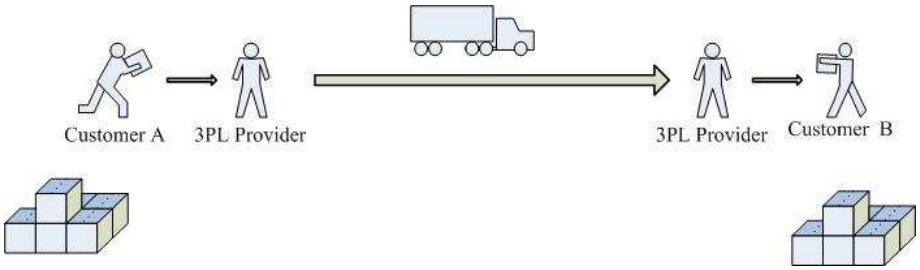

Figure 2.1: 3PL Supply Chain Model

Different with the general adversary model which only considers the external adversaries, our adversary model for 3PL supply chain also considers the internal adversaries as well as the external adversaries. We analyze the potential dishonest behaviors of the three parties and the disputes that may happen on the ownership transfer between Party C and Party B as below. Note that we suppose ownership transfer from Party A to Party C is free of disputes. The reason is that a transaction will not begin unless Party A and Party C get into an agreement.

- *In case Party A is dishonest:* Party A sends a batch of low quality goods which do not satisfy Party B's requirements. When Party B refuses to accept the goods, Party A may claim that the goods are not original ones that it delivered as they have been replaced by Party C.

- *In case Party C is dishonest:* In case Party C loses or damages some goods during the transportation, to escape the compensation [3], Party C then fabricates the tags of the lost goods, and attaches them on fake goods. When Party B detects the replacements, Party C may claim that the faked goods came from Party A.

- *In case Party B is dishonest:* The dishonest Party B may intentionally refuse to accept the goods by claiming that goods do not satisfy the requirements.

[3] Even worse, Party C replaces some goods and steals the original ones.

The major work for RFID system in 3PL supply chain is to facilitate Party C to transfer goods from Party A to Party B. The system should be able to detect Party C's malicious behavior. Inherently, we cannot prevent Party A (Party B) from cheating Party B (Party A), however, at least we should keep Party C away from Party A (Party B)'s malicious behaviors. The requirements of the RFID system for 3PL supply chains against internal adversaries are listed as below:

- *Restrain dishonest Party C:* Party C should not be able to replace any goods without being detected. About privacy, in 3PL supply chain, Party A and Party B may not want to leak the goods' information to Party C. While the tags will be under Party C's control, the system should protect the tags' information leakage against Party C.

- *Protecting honest Party C:* If Party C honestly and successfully transfers the goods to Party B, Party B should accept the goods unconditionally, even if the goods do not meet the requirements on product quality (Later, party B can negotiate with Party A without involving Party C).

We assume that external adversaries only conduct the attacks during the transportation of goods. [4] The privacy and security requirements against external adversaries are listed as below.

- *Tag information privacy:* It means that external adversaries cannot get the information of the tags.

- *Tag location privacy:* If the responses of a tag are linkable to each other or distinguishable from those of other tags, then the location of a tag could be tracked by multiple collaborating tag readers. Tag location privacy means no one except the legitimate party can trace the tags.

[4] The two ownership transfer happens in relative secure environments that under two parties' surveillance.

- *Resistance of tag impersonation attack:* It means that the attacker impersonates a target tag without knowing the tag internal secrets and pass the authentication of the reader.

- *Resistance of replay attack:* It means that the attacker reuses communications from previous sessions to perform a successful authentication between a tag and a server.

2.2.3 Path Authentication

As mentioned in Section 1.1.3, we refer path authentications schemes as the ones that satisfy the following criteria: 1) the verification of a tag's genuineness is merely based on the contents stored on the tag; 2) it does not require supply chain partners to have any connection (except in the initial stage, if needed).

A supply chain consists of multiple entities. We model a dynamic supply chain according to three properties: the affiliation of each entity, the membership management of the supply chain and the logistics flow of the supply chain. In a closed supply chain, there may exist dominant entities, for example, some entities serve as supply chain managers; the entities in supply chain are fixed; the logistics flow is also fixed. While in a dynamic supply chain, each entity is independent of each other; any entity can freely join or leave the supply chain; the logistics flow is not fixed.

The security goal of our system is to prevent an adversary from inserting counterfeited goods to the supply chain. The manager checks the authenticity of a tag merely based on the state stored on a tag. The system should prevent an adversary from forging a tag's internal state as it has gone through a valid path that actually has not been taken. Since standard EPC C1 G2 tags have no computation capability, no reader authentication is performed. If a tag's state has been changed by an adversary, even if it has gone through a valid path, it is not considered as a valid tag by a manager. The privacy goal is no adversary can infer any information about a

tag from the tag state. Formal definitions of the security and privacy requirements can be found in Section 6.1.

2.3 Related Work

2.3.1 Mutual Authentication

We first consider the tag-reader mutual authentication problem. The wireless communications between RFID readers and tags may subject to a variety of attacks, e.g., replay attack, DoS attack, tag impersonation attack, and server impersonation attack. As listed in [61], a secure mutual authentication protocol should resist all the above attacks, in addition, the protocol should guarantee the tag information privacy, tag location privacy, backward tranceability and forward traceability. Though many solutions have been proposed to protect RFID communications, few of them address all of the above security and privacy concerns.

The hash lock solutions proposed by Weis et al. [65] use a one-way hash function to lock a tag. Without providing appropriate key, a tag responds to a query with a fixed meta-ID in Hash-based Access Control (HAC) or a hash function of the tag ID and a random number in Randomized Access Control (RAC). HAC is subject to location tracking attack since the meta-ID is fixed at any time when a tag is queried. RAC prevents this tracking attack; however, it is vulnerable to tag impersonation attack since an intercepted tag's response can be replayed. Since the tag ID is fixed, these solutions do not provide any forward or backward security once a tag is compromised.

Many other solutions also suffer from tag tracking attack, tag impersonation attack (or replay attack), and/or lack of forward or backward security. In [21], Dimitriou proposed an RFID protocol that uses a challenge-response approach. Since the tag identifier remains the same between valid sessions, this solution is subject to tag tracking and tag impersonation attacks. In [38], Ohkubo, Suzki, and Kinoshita

proposed to use a low-cost hash chain mechanism to update tag secret information and provide forward security. Although this mechanism realizes the identification of the tags while ensuring privacy, it allows an adversary to impersonate tags without knowing the tag secrets. In [43], Li and Ding proposed a de-centralized solution for secure RFID communications in supply chains. In their solution, an access key is shared between each tag and each supply chain party. The access key of a tag can be updated by the current supply chain party before the tag is handed over to the next supply chain party. Since the updated access key is shared between the current supply chain party and the next supply chain party, their solution is vulnerable to insider attacks without backward or forward secrecy.

In [31], Henrici and Müller rely one-way hash function to thwart tag tracking attacks. In this solution, a tag responses a reader's query with two hash values and updates its stored values after a successful authentication. This solution does not provide full-degree of anti-tracking since a tag always replies with the same responses before it is successfully authenticated. In addition, it does not provide forward security as a strong adversary could derive tag identifiers in previous sessions from the tag's current identifier and the server's random number.

In [50], Molnar and Wagner proposed to use a tree structure to organize the tag secret keys for efficient authentication of RFID tags. In this proposal, the tags share some common secrets with other tags. Therefore, if one tag is tampered, other tags will be influenced. In [23], Duc et al. suggested a synchronization based protocol against tag tracking based on simple cryptographic primitives like pseudo-random number generator and cyclic redundancy code. This solution cannot prevent replay attacks between successful authentications. It does not provide forward security either if the fixed EPC code and access key PIN are compromised. In addition, it is vulnerable to denial of service attack since an adversary could permanently de-synchronize the interactions between a server and a tag.

In [20], Chien and Chen used a challenge-response protocol to prevent replay attacks. To prevent denial of service attacks, both new key and old key for authen-

ticating a tag are stored in back-end database. However, a strong adversary can still identify a tag's fixed EPC code, thus identify the tag's past and future interactions after compromising a tag.

Among existing RFID mutual authentication protocols, the protocol proposed by Song and Mitchell (SM) [60] is asserted to have the most security properties. In SM protocol, a tag's response to a query is generated from two fresh random numbers, one from the querying reader, one from the tag itself, as well as the tag secret. Without knowing the tag secret, it is computationally infeasible to identify the tag or track its location. This challenge response solution also prevents the replay attacks or tag impersonation attacks. To provide forward and backward security, the tag secret is updated by a legitimate reader pseudo-randomly during each successful authentication. To prevent denial of service attacks through de-synchronization of secret update, the back-end server stores both the updated tag secrets as well as the previous ones for each tag for future authentications.

While most research has been focused on mutual authentication between RFID tags and one owner, recent study has turned to transferring ownership from one owner to another in a secure manner. Song has extended SM protocol for ownership transfer and compared it to some other RFID ownership transfer protocols [60]. It is asserted that the Song's protocol not only inherits all the security properties of SM protocol, but also has three new properties including old owner privacy, new owner privacy and authorization recovery. Song's protocol uses a two-party model, which is different from the three-party model [49, 56] that involves a trusted third party to coordinate the ownership transfer.

Among the two-party ownership transfer protocols, the SIS-2 scheme [56] stores a fixed tag identifier in each tag; thus, a tag's past communications can be traced if the tag is compromised. In SIS-2 scheme, the new owner of a tag encrypts both the old key (obtained from the previous owner) and a new key using a nonce obtained from the backward channel (i.e., communication channel from tag to reader), which is assumed to be more secure against eavesdropping than the forward channel

(from reader to tag). The ownership transfer is at risk if a strong adversary is able to eavesdrop the backward channel [56]. Other two-party ownership transfer protocols either provide no old owner privacy [44] or are vulnerable to tag tracking attacks, denial of service, and/or replay attacks [53, 25, 26]. A comparison study performed in [60] shows that only the Song's ownership transfer protocol provides all the necessary security properties and yet is efficient in terms of storage and communication requirements.

2.3.2 Three Party Authentication

Dozens of cryptographic protocols have been proposed to provide secure and private identification and authentication of the tag (Sometimes the reader authentication is also required), such as the "hash-lock" protocol of Weis et al. [65], OSK protocol [52] of Ohkubo, Suzuki and Kinoshita, and the tree-based protocol of Molnar et al. [50]. Also, there are protocols [49, 60] which deal with the secure and private ownership transfer between two parties. Most of these solutions for authentication and ownership transfer are "symmetric secret"-based that an authorized reader shares a secret with each tag.

The "symmetric secret"-based solutions are designed to protect the system against external adversaries who do not know the secrets. However, in 3PL supply chains that three parties (the sender of the goods, the receiver of the goods and the 3PL provider) are involved in the processing of the tags, internal adversaries should be considered. In 3PL supply chains, all of the three parties need to access the tags, hence all of them should have a copy of the secret when a "symmetric secret"-based solution is deployed. With a tag's secret, any party can fabricate the tag. In case inside adversaries exist, disputes on the goods' originality will be hard to solve since all the three parties have the ability to fabricate the tags.

Currently, there does not exist any solution that is suitable for 3PL supply chains considering the exitance of internal adversaries. There is a concept called "group-

ing proof" proposed by Juels [34] which is similar with our "group checking". The pharmaceutical distribution example is used to illustrate how grouping proof protocols work. Yoking-proof would provide an evidence that each container of the medication was dispensed with a leaflet in case that a tag is embedded in the container and another tag is embedded in an accompanying leaflet. Yoking proof only enables two tags to prove their co-existence and is vulnerable to replay attack. Later works on grouping proof [57, 54, 12, 45] support multiple tags and putting their efforts on improving the security and efficiency.

In both "grouping proof" scenario and "group checking" scenario, RFID readers are not trusted; therefore, they do not hold tag secrets . In "grouping proof" scenario, the reader aims to prove to a verifier that the tags are processed together, in case the reader has the secrets, he can forge a proof using the keys. In "group checking" scenario, the reader should prove the tags' originality to another party, in case he has the secrets, he can forge the tags. Besides reading the tags in batch level without knowing the secrets, the two kinds of schemes work differently. In "group proof" scenario, the reader does not need to authenticate the tags. The reader only acts as a transfer stop in the grouping proof protocols in transmitting the messages among the tags. The whole tags generate a proof. While in "group checking" even without getting the secrets of the tags, the reader should have the ability to check the integrity and originality of a batch of tags. The reader interacts with the tags and verifies the tags' originality.

2.3.3 Path Authentication

In path authentication protocols, one can verify the genuineness of a tag by checking whether it has been processed by a series of reliable readers merely based on the information stored on the tag. Thus, path authentication could be deployed to resist counterfeiting in supply chains. Path authentication protocols rely on standard EPC C1 G2 tags.

Recently, Blass, Elkhiyaoui and Molva proposed TRACKER [8] and its extension [7]. In the TRACKER system, each tag stores encrypted verifiable ID and path information. The manager decrypts the information and verifies whether the tag has gone through a valid path. TRACKER dose not need the entities in a supply chain to have any connection except the initialization phase. However, it requires the manager to possess the secret keys of all involving entities, the manager can only verify the tags from a set of pre-fixed paths. Therefore, it is difficult to implement TRACKER in dynamic supply chains.

There is another work which addresses the counterfeiting problem in RFID-enabled supply chains based on standard EPC Class 1 Gen 2 tags. In Zanetti, Fellmann and Capkun's scheme [68], the entities cooperate together to verify the tags' genuineness without revealing information to each other. This scheme cannot resist any tracking attack since any reader can read out each tag's ID.

2.4 Summary

In this chapter, we provided an overall background of RFID-enabled supply chains. We summarized the security and privacy concerns of three kinds of tag-reader authentication protocols: mutual authentication, three party authentication, and path authentication. We showed how the three kinds of protocols can fit into various supply chains.

Chapter 3

Attacks and Improvements to Mutual Authentication Solutions in RFID-Enabled Supply Chains

While RFID technology has greatly facilitated visible supply chain management, it is challenging to design a secure and privacy-preserving RFID-tagged supply chain system. Many tag-reader mutual authentication protocols have been proposed; however, a significant portion of them are broken. In this chapter, we demonstrate specific attacks to two existing protocols, and provide our improvements to them.

In WiSec'08, Song and Mitchell proposed an RFID mutual authentication protocol(SM mutual authentication protocol) [61]. Song also extended this protocol for RFID tag ownership transfer. These two protocols are designed to have the most security properties in the literature. We examine both SM mutual authentication protocol and Song's ownership transfer protocol. We discover that SM mutual authentication protocol is vulnerable to both tag impersonation attack and reader impersonation attack. These attacks enable an adversary to impersonate any legitimate reader or tag; thus, the mutual authentication between readers and tags is broken. We also discover that the ownership transfer protocol is vulnerable to a denial of service attack, in which an adversary can update a tag's secret to a random

value. Consequently, this attack prevents a legitimate reader from authenticating a legitimate tag, and vice versa; it thus breaks the mutual authentication as well. After analyzing the weaknesses of these protocols, we propose revised protocols to prevent these attacks with comparable storage and computational requirements.

3.1 Protocols

We use the same notation as in [61, 60]. The notations are given in table 3.1 for ease of reference.

l	The bit-length of a tag identifier	ϵ	Error message
N	The number of tags	$\oplus$	XOR operator
f_k	A keyed hash function, $f_k : \{0,1\}^\star \times$	$\|\|$	Concatenation operator
	$\{0,1\}^l \rightarrow \{0,1\}^l$(a MAC algorithm)	$\leftarrow$	Substitution operator
h	A hash function, $h : \{0,1\}^\star \rightarrow \{0,1\}^l$	$x \gg k$	Right circular shift operator, which rotates all
T_i	The i-th tag $(1 \leq i \leq N)$		bits of x to the right by k bits, as if the left
D_i	The detailed information of tag T_i		and right ends of x were joined.
s_i	A string of l bits assigned to T_i	$x \ll k$	Left circular shift operator, which rotates all
t_i	T_i's identifier of l bits, which equals $h(s_i)$		bits of x to the left by k bits, as if the left and
x_{new}	The new (refreshed) value of x		right ends of x were joined.
x_{old}	The previous value of x	$\in_R$	The random choice operator, which randomly
r	A random string of l bits		selects an element from a finite set using
$[x]_{L/R}$	The left/right half part of the string x		a uniform probability distribution

Table 3.1: Notation

In initiation, an initiator (e.g. the tag manufacturer) assigns a unique string s_i of l bits to each tag T_i, computes $t_i = h(s_i)$, and stores t_i in the tag, where l should be large enough so that an exhaustive search to find the l-bit values t_i and s_i is computationally infeasible. The server stores a tuple of $[(s_i, t_i)_{new}, (s_i, t_i)_{old}, D_i]$ in its database for each tag, where $(s_i, t_i)_{new}$ is the newly assigned secrets, $(s_i, t_i)_{old}$ is the old secrets, and D_i is the tag's associate information. At the initial stage,

$(s_i, t_i)_{new} = (s_i, t_i)_{old} = (s_i, t_i)$. Only $t_{i(new)}$ is stored on the tag. Note that we use $(s_i, t_i)_{new/old}$ and $(s_{i(new/old)}, t_{i(new/old)})$ inter-changeably.

3.1.1 Mutual Authentication Protocol

The SM mutual authentication protocol [61] is shown in Figure 3.1. In this protocol, a reader first generates a random bit-string $r_1 \in_R \{0,1\}^l$ and sends it to a tag T_i. Upon receiving r_1, the tag T_i generates a random bit-string $r_2 \in_R \{0,1\}^l$, computes two messages $M_1 = t_i \oplus r_2$ and $M_2 = f_{t_i}(r_1 \oplus r_2)$, and sends (M_1, M_2) back to the reader. Note that both M_1 and M_2 are pseudo-random to any adversary who has no access to the tag secret t_i. Upon receiving the tag response, the reader sends (r_1, M_1, M_2) to the back-end server for tag authentication. The server will search in its database for a record $(s,t) = (s_i, t_i)_{new}$ or $(s,t) = (s_i, t_i)_{old}$ such that $M_2 = f_t(r_1 \oplus M_1 \oplus t)$. If no match is found, the server sends ϵ to the reader and terminates the session. If a unique record is found, the server computes $r_2 = M_1 \oplus t$, and then computes $M_3 = s \oplus (r_2 \gg l/2)$ and sends it with D_i to the reader. After this, the server updates $(s_i, t_i)_{old}$ with $(s_i, t_i)_{new}$ and updates $(s_i, t_i)_{new}$ to be $s_{i(new)} = (s \ll l/4) \oplus (t \gg l/4) \oplus r_1 \oplus r_2$ and $t_{i(new)} = h(s_{i(new)})$. Finally, upon receiving D_i and M_3, the reader forwards M_3 to T_i. Tag T_i computes $s_i = M_3 \oplus (r_2 \gg l/2)$ and checks if $h(s_i) = t_i$ for reader authentication. If $h(s_i) = t_i$ holds, the tag updates its secret t_i to be $t_i = h((s_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2)$; otherwise, the tag keeps the current value of t_i unchanged.

Extending SM protocol, Song proposed an ownership transfer protocol [60]. The ownership transfer protocol consists of three sub-protocols, ownership sharing protocol (which is also called ownership transfer protocol in [60]), secret update protocol, and authorization recovery protocol. Since the authorization recovery protocol is essentially the same as the secret update protocol (except that a tag's secret is updated to its old value), we focus on the first two sub-protocols.

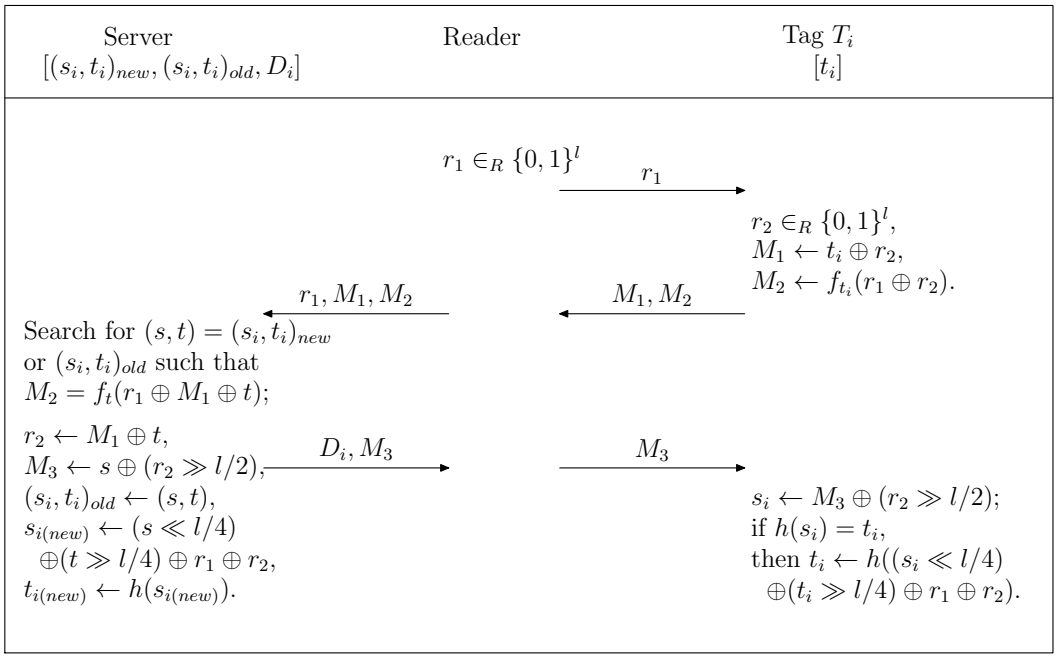

Figure 3.1: SM Mutual Authentication Protocol

3.1.2 Ownership Sharing Protocol

The ownership sharing protocol is a straightforward extension to SM mutual authentication protocol by replacing "server" with S_j and replacing "reader" with S_{j+1}, where S_j denotes the current owner of tag T_i, and S_{j+1} denotes the new owner to which the ownership of tag T_i is to be shared. At the beginning of the ownership sharing, S_{j+1} cannot identify nor authenticate a tag T_i; therefore, it communicates with S_j through a secure communication channel for help. The current owner S_j will search for its database, locate the tag's secrets, and update them exactly as what the server does in SM protocol. The only difference is that, besides transmitting the tag data D_i and an updating message M_3, the current owner S_j also transmits the updated tag secrets $(s_i,t_i)_{new}$ to the new owner S_{j+1}. Then, the new owner S_{j+1} inserts a new record $\langle D_i,(s_i,t_i)_{new},(s_i,t_i)_{new}\rangle$ into its database and updates T_j by sending M_3 to the tag[1]. At this point, both S_j and S_{j+1} have access to $(s_i,t_i)_{new}$; therefore, the ownership is shared between them. Since S_j has updated the tag se-

[1]A de-synchronization attack to the communication between the new owner and the tag can be easily detected in the secret update protocol. It can be corrected by repeating the last step in the ownership sharing protocol.

crets before it sends them to S_{j+1}, the privacy of the old owner is established. The privacy of new owner is realized through the secret update protocol that is given below.

3.1.3 Secret Update Protocol

To protect its own privacy, the new owner S_{j+1} needs to update the tag one more time with fresh new tag secrets after performing the ownership sharing protocol. The detail of this process, which is called secret update protocol, is shown in Figure 3.2.

For each tag T_i, the new owner S_{j+1} finds its secrets $(s_i, t_i)_{new}$ in its database. The owner then generates two random strings r_1 and s_i' of l bits, computes $t_i' = h(s_i')$, $M_1 = f_{t_{i(new)}}(r_1) \oplus t_i'$, and $M_2 = s_{i(new)} \oplus (t_i' \gg l/2)$, and sends ($r_1$, M_1, M_2) to T_i.

Upon receiving (r_1, M_1, M_2), the tag recovers $t_i' = M_1 \oplus f_{t_i}(r_1)$ and $s_i = M_2 \oplus (t_i' \gg l/2)$. The tag authenticates the new owner S_{j+1} by checking whether $h(s_i) = t_i$ holds; if no, the session stops; otherwise, the tag updates its secret as $t_i \leftarrow t_i'$, generates a random string r_2 of l bits, and computes $M_3 = f_{t_i}(r_1 \oplus r_2)$. Finally, it sends ($r_2$, M_3) back to S_{j+1} as confirmation. The server S_{j+1} checks whether M_3 is equal to $f_{t_i'}(r_1 \oplus r_2)$. If the validation succeeds, S_{j+1} is confirmed that T_i has successfully updated to the new secret t_i'. Then, the server updates the tag secrets $(s_i, t_i)_{old}$ to $(s_i, t_i)_{new}$, and $(s_i, t_i)_{new}$ to (s_i', t_i'), respectively.

We should emphasize that at this point of time, the privacy of new owner is still not fully established. The reason is that the previous owner S_j, with access to $(s_i, t_i)_{new}$ at the beginning of this secret update protocol, can easily derive the new tag secret t_i' by eavesdropping r_1 and M_1. Therefore, after the secret update protocol, S_j can still identify/authenticate the tag based on t_i'. However, S_j has no knowledge about s_i', which is known to the new owner S_{j+1} only. To ensure the privacy of new owner, the new owner needs to perform the SM protocol successfully for at least one time, which further updates the tag secret t_i' based on s_i'.

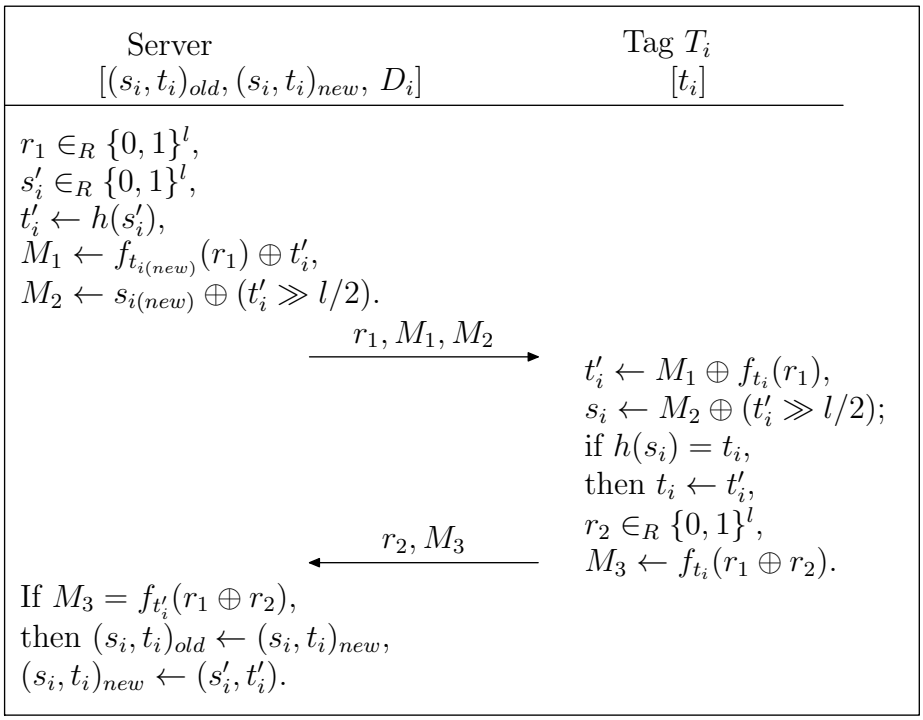

Figure 3.2: Song's Secret Update Protocol

3.2 Attacks

SM mutual authentication protocol is asserted to have a list of security properties [61]. In this section, we present concrete attacks to both SM protocol and Song's extensions. We show that SM mutual authentication protocol is vulnerable to a server impersonation attack an a tag impersonation attack, and Song's secret update protocol is vulnerable to a de-synchronization attack.

We use the standard Dolev-Yao intruder model [22], in which the adversary controls the "network." In this model, the adversary can eavesdrop, block, modify, and inject messages in any communication between a reader and a tag. However, we assume that the connections among the servers and the readers are secure, which can be guaranteed by using full-fledged cryptographic technologies.

3.2.1 Server Impersonation Attack

We first show that an active adversary can impersonate a legitimate server to a tag in the SM mutual authentication protocol even when the adversary has no access to any of the tag's secrets. This attack can result in de-synchronized updating of the secret information between tag and server. The detail of this attack is given below.

1. The adversary eavesdrops a valid session, denoted as $session_1$. In $session_1$, a legitimate reader generates a random string $r_1 \in \{0,1\}^l$ and sends r_1 to the tag. After receiving r_1, the tag chooses a random string $r_2 \in \{0,1\}^l$, computes $M_1 = t_i \oplus r_2$ and $M_2 = f_{t_i}(r_1 \oplus r_2)$. The tag sends the reply (M_1, M_2) to the reader. The reader then queries the back-end server, gets the updating message M_3, and sends it to the tag for updating of the tag's secret. The adversary records the values of r_1, M_1, M_2, M_3 and blocks the sending of M_3 from reaching the tag.

2. The adversary initiates another session, $session_2$, as an impersonated reader. It chooses a random string $r_1' \in \{0,1\}^l$ and sends it to the tag. The tag replies with (M_1', M_2'). Then, the adversary calculates a new value, M_3', using the following equations

$$[M_3']_L = [M_1]_R \oplus [M_3]_L \oplus [M_1']_R \tag{3.1}$$

$$[M_3']_R = [M_1]_L \oplus [M_3]_R \oplus [M_1']_L \tag{3.2}$$

3. The adversary sends M_3' to the tag and the tag will accept M_3' and update its secret t_i to a new value

$$h((s_i' \ll l/4) \oplus (t_i \gg l/4) \oplus r_1' \oplus r_2')$$

where $r_2' = M_1' \oplus t_i$ and $s_i' = M_3 \oplus (r_2' \gg l/2)$.

We now prove that the tag will accept M_3' and update its secret t_i accordingly. In $session_1$, M_3 is accepted by T_i; therefore, we have $h(s_i) = t_i$, where $s_i = M_3 \oplus (r_2 \gg l/2)$. To prove that M_3' is accepted by the tag, we need to prove that in $session_2$, $h(s_i') = t_i$ holds.

Since $r_2 = M_1 \oplus t_i$, we have

$$\begin{aligned} s_i &= M_3 \oplus (r_2 \gg l/2) \\ &= [M_3]_L \oplus [r_2]_R \parallel [M_3]_R \oplus [r_2]_L \\ &= [M_3]_L \oplus [M_1]_R \oplus [t_i]_R \parallel [M_3]_R \oplus [M_1]_L \oplus [t_i]_L \end{aligned} \tag{3.3}$$

On the other hand, we have

$$\begin{aligned} s_i' &= M_3' \oplus (r_2' \gg l/2) \\ &= [M_3']_L \oplus [r_2']_R \parallel [M_3']_R \oplus [r_2']_L \end{aligned}$$

Since $r_2' = M_1' \oplus t_i$, we have

$$s_i' = [M_3']_L \oplus [M_1']_R \oplus [t_i]_R \parallel [M_3']_R \oplus [M_1']_L \oplus [t_i]_L$$

Incorporating equations (3.1) and (3.2) in the above equation, we have

$$\begin{aligned} s_i' &= [M_1]_R \oplus [M_3]_L \oplus [M_1']_R \oplus [M_1']_R \oplus [t_i]_R \parallel \\ &\quad [M_1]_L \oplus [M_3]_R \oplus [M_1']_L \oplus [M_1']_L \oplus [t_i]_L \\ &= [M_3]_L \oplus [M_1]_R \oplus [t_i]_R \parallel [M_3]_R \oplus [M_1]_L \oplus [t_i]_L \end{aligned} \tag{3.4}$$

Comparing equations (3.3) and (3.4), we have $s_i = s_i'$. Therefore, we have $h(s_i') = h(s_i) = t_i$ and the tag accepts M_3'. Since the adversary can impersonate as a server, he or she can update the tag secret t_i to a new value. As a result, the tag can not be identified or authenticated by any valid server in the future sessions.

3.2.2 Tag Impersonation Attack

We next show that an active adversary can impersonate a tag to a legitimate server in the SM mutual authentication protocol even when the adversary has no access to any of the tag's secrets. This attack can be used to fool a reader that a tag is present while it is not. In this attack, an adversary first queries a valid tag with a random string $r_1' \in \{0,1\}^l$ and records the tag's reply (M_1', M_2'). For any query r_1 issued by a valid reader, the adversary can forge a valid tag reply (M_1, M_2) from recorded r_1', M_1', M_2' [62].

Phase 1: The adversary queries a tag T_i as a reader.

1. $Adversary \longrightarrow T_i$: The adversary generates a random string $r_1' \in_R \{0,1\}^l$ and sends r_1' to T_i.

2. $T_i \longrightarrow Adversary$: The tag T_i generates a random string $r_2 \in_R \{0,1\}^l$, computes $M_1' = t_i \oplus r_2$ and $M_2' = f_{t_i}(r_1' \oplus r_2)$, and sends (M_1', M_2') to the adversary.

Phase 2: The adversary impersonates the tag T_i to a valid reader.

1. $Reader \longrightarrow Adversary$: The valid reader queries the tags with a random string $r_1 \in_R \{0,1\}^l$.

2. $Adversary \longrightarrow Reader$: The adversary receives r_1, sends the reply (M_1, M_2) to the reader, where $M_1 = M_1' \oplus r_1 \oplus r_1'$, $M_2 = M_2'$.

3. $Reader \longrightarrow Server$: The reader sends (r_1, M_1, M_2) to the server. The server searches its database for t_i such that $M_2 = f_t(M_1 \oplus t_i \oplus r_1)$ holds.

Now, we prove that $M_2 = f_t(M_1 \oplus t_i \oplus r_1)$ indeed holds for tag T_i. Since (M_1', M_2') is a valid reply to the query r_1', we have $M_2' = f_{t_i}(M_1' \oplus t_i \oplus r_1')$. Since $M_1 = M_1' \oplus r_1 \oplus r_1'$ (or equivalently $M_1' = M_1 \oplus r_1 \oplus r_1'$), we have $M_2' = f_{t_i}(M_1 \oplus r_1 \oplus r_1' \oplus t_i \oplus r_1') = f_{t_i}(M_1 \oplus t_i \oplus r_1)$. Since $M_2' = M_2$, we have $M_2 = f_t(M_1 \oplus t_i \oplus r_1)$.

3.2.3 De-Synchronization Attack

Our third attack is launched to Song's secret update protocol, in which the new owner/server updates a tag with message (r_1, M_1, M_2) and gets confirmation (r_2, M_3) from the tag if the update is successful. The server keeps both the updated tag secrets and the old secrets for the purpose of preventing de-synchronization attacks. We will show that, however, the de-synchronization attack is still possible as an active adversary can block the first message (r_1, M_1, M_2) from reaching the tag, and then forge a second message (r_1, M_1', M_2') that can be accepted by the tag. Our attack will result in the tag's secret t_i to be updated to a value that is unknown to the legitimate server. As a result, the tag cannot be identified or authenticated by any legitimate reader in the future sessions. The steps of this de-synchronization attack are given below.

1. The legitimate server starts the secret update protocol to a tag by sending message (r_1, M_1, M_2) to the tag, where $r_1 \in_R \{0,1\}^l$, $M_1 = f_{t_i}(r_1) \oplus t_i'$, and $M_2 = s_i \oplus (t_i' \gg l/2)$. Recall that s_i and $t_i = h(s_i)$ are the current tag secrets and that $s_i' \in_R \{0,1\}^l$ and $t_i' = h(s_i')$ are the new secrets to be updated.

2. The adversary blocks the message (r_1, M_1, M_2) from reaching the tag. Then, the adversary forges another message (r_1, M_1', M_2') (without access to any tag secrets) and sends it to the tag, where $M_1' \in_R \{0,1\}^l$, and M_2' is forged from M_1, M_2 and M_1' as follows

$$[M_2']_L = [M_1]_R \oplus [M_2]_L \oplus [M_1']_R \tag{3.5}$$

$$[M_2']_R = [M_1]_L \oplus [M_2]_R \oplus [M_1']_L \tag{3.6}$$

3. The tag will accept message (r_1, M_1', M_2') and update the tag secret t_i to a new value

$$t_i'' = M_1' \oplus f_{t_i}(r_1) \tag{3.7}$$

We now prove that the tag will indeed accept (r_1, M_1', M_2') and update its tag secret to t_i''. According to the protocol, we need to prove that $h(M_2' \oplus (t_i'' \gg l/2)) = t_i$ holds on the tag side. Since (r_1, M_1, M_2) is generated by a valid server, we have $h(M_2 \oplus (t_i' \gg l/2)) = t_i$. Therefore, we only need to prove $M_2' \oplus (t_i'' \gg l/2) = M_2 \oplus (t_i' \gg l/2)$. Starting from the left-hand side, we have

$$M_2' \oplus (t_i'' \gg l/2) = [M_2']_L \oplus [t_i'']_R \parallel [M_2']_R \oplus [t_i'']_L \tag{3.8}$$

By incorporating equation (3.7), we have

$$\begin{aligned} M_2' \oplus (t_i'' \gg l/2) &= [M_2']_L \oplus [M_1']_R \oplus [f_{t_i}(r_1)]_R \parallel \\ &\quad [M_2']_R \oplus [M_1']_L \oplus [f_{t_i}(r_1)]_L \end{aligned}$$

By incorporating equations (3.5) and (3.6), we have

$$\begin{aligned} M_2' \oplus (t_i'' \gg l/2) &= [M_2]_L \oplus [M_1]_R \oplus [f_{t_i}(r_1)]_R \parallel \\ &\quad [M_2]_R \oplus [M_1]_L \oplus [f_{t_i}(r_1)]_L \end{aligned}$$

Since $M_1 = f_{t_i}(r_1) \oplus t_i'$ (or equivalently $t_i' = M_1 \oplus f_{t_i}(r_1)$), we have

$$\begin{aligned} M_2' \oplus (t_i'' \gg l/2) &= [M_2]_L \oplus [t_i']_R \parallel [M_2]_R \oplus [t_i']_L \\ &= M_2 \oplus (t_i' \gg l/2) \end{aligned}$$

After checking (r_1, M_1', M_2'), the tag updates its secret to t_i'', which is different from the current secret t_i. It is also different from t_i', to which t_i should be updated if the valid message (r_1, M_1, M_2) has not been blocked. Therefore, the server has no knowledge on the updated tag secret after this de-synchronization attack and thus the server will not able to identify or authenticate the tag in the future sessions.

3.3 Improvements

Inexpensive security operations such as $\oplus$ and $\gg$ are extensively used in the design of SM protocol and its extensions. Although these operations can help reduce the cost of RFID tags, they lead to various security vulnerabilities if used inappropriately as shown in the previous section. In this section, we propose to revise these protocols so as to eliminate the existing security vulnerabilities.

3.3.1 Revised Mutual Authentication Protocol

Our revised mutual authentication protocol is the same as the SM protocol except for the approach the messages M_2 and M_3 are generated. In SM protocol, M_2 is set to be $f_{t_i}(r_1 \oplus r_2)$ on the tag side, and $f_t(r_1 \oplus M_1 \oplus t)$ on the server side, where $t = t_i$ and $M_1 \oplus t = r_2$. We revise it to be $M_2 \leftarrow f_{t_i}(r_1||r_2)$ on the tag side and $M_2 = f_t(r_1||M_1 \oplus t)$ on the server side. In addition, we revise $M_3 \leftarrow (r_2 \gg l/2) \oplus s$ to $M_3 \leftarrow h(r_2) \oplus s$ on the server side, and revise $s_i \leftarrow M_3 \oplus (r_2 \gg l/2)$ to $s_i \leftarrow M_3 \oplus h(r_2)$ on the tag side.

3.3.2 Revised Secret Update Protocol

Our revised secret update protocol is the same as Song's secret update protocol except for the approach the message M_2 is generated. In Song's secret update protocol, M_2 is set to be $s_{i(new)} \oplus (t_i' \gg l/2)$ on the server side. We revise it to be $s_{i(new)} \oplus h(t_i')$. Consequently, on the tag side, we revise $s_i \leftarrow M_2 \oplus (t_i' \gg l/2)$ to $s_i \leftarrow M_2 \oplus h(t_i')$.

3.3.3 Reasoning

The SM protocol is vulnerable to the tag impersonation attack and the reader impersonation attack and Song's secret update protocol is vulnerable to the desynchronization attack. All these attacks are due to the use of inexpensive security

operations $\oplus$ and $\gg$ inappropriately. Our revised protocols eliminate the vulnerabilities to these attacks without affecting any other security properties.

Resistance to server impersonation attack In the design of the SM protocol, the updating message M_3 is used to authenticate a server. The validity of M_3 must rely on the possession of (s_i, t_i). Only with the knowledge of t_i, the server can get the tag's nonce r_2, and only with the knowledge of s_i and r_2, the server can form a valid M_3. The weakness of the SM protocol is that by examining the relationship between $M_1 = t_i \oplus r_2$ and $M_3 = s_i \oplus (r_2 \gg l/2)$, an adversary can forge a valid M_3 without knowing the value of r_2, nor the value of (s_i, t_i):

$$\begin{aligned} M_3 &= s_i \oplus (r_2 \gg l/2) \\ &= ([s_i]_L \oplus [r_2]_R) \| ([s_i]_R \oplus [r_2]_L) \\ &= ([s_i]_L \oplus [M_1]_R \oplus [t_i]_R) \| ([s_i]_R \oplus [M_1]_L \oplus [t_i]_L) \end{aligned}$$

Therefore, if the adversary has access to the values of $[s_i]_L \oplus [t_i]_R$ and $[s_i]_R \oplus [t_i]_L$, he or she can forge a valid M_3 according to M_1 sent by the tag. In the server impersonation attack, the adversary eavesdrops a valid session and computes $[s_i]_L \oplus [t_i]_R$ and $[s_i]_R \oplus [t_i]_L$ from M_1 and M_3, where $[s_i]_L \oplus [t_i]_R = [M_3]_L \oplus [M_1]_R$ and $[s_i]_R \oplus [t_i]_L = [M_3]_R \oplus [M_1]_L$. The adversary blocks M_3 so that the secret on the tag is not updated. In another session, the adversary can forge a valid M_3' from the tag's reply (r_1', M_1', M_2') using the following equations

$$[M_3']_L \oplus [M_1']_R = [s_i]_L \oplus [t_i]_R = [M_3]_L \oplus [M_1]_R$$

$$[M_3']_R \oplus [M_1']_L = [s_i]_R \oplus [t_i]_L = [M_1]_L \oplus [M_3]_R$$

The vulnerability is due to the use of operation $\oplus$ in computing M_3. Our revision replaces r_2 with $h(r_2)$ in the computing of M_3. Hence, the adversary cannot skip r_2, which is freshly chosen by the tag in each session, nor s_i in computing a valid

M_3. Our revised mutual authentication protocol can thus prevent the server impersonation attack.

Resistance to tag impersonation attack In the SM protocol, the tag T_i replies to a reader's query r_1 with $M_1 = t_i \oplus r_2$ and $M_2 = f_{t_i}(r_1 \oplus r_2)$, where r_2 is a random string chosen by T_i. For any $r_1' \in \{0,1\}^l$ and $M_1' = M_1 \oplus r_1' \oplus r_1$, the following holds on the tag side:

$$\begin{aligned} M_2 &= f_{t_i}(r_1 \oplus M_1 \oplus t_i) \\ &= f_{t_i}(r_1' \oplus M_1 \oplus r_1 \oplus r_1' \oplus t_i) \\ &= f_{t_i}(r_1' \oplus (M_1 \oplus r_1' \oplus r_1) \oplus t_i) \\ &= f_{t_i}(r_1' \oplus M_1' \oplus t_i) \end{aligned}$$

This means that (M_1', M_2) is also a valid reply. An adversary can forge this reply from (M_1, M_2) for any r_1'. This vulnerability is due to the inappropriate use of $\oplus$ in computing M_2. In our revised mutual authentication protocol, we replace this $\oplus$ with $||$. Consequently, the adversary can not impersonate the tag unless the server uses the same challenge r_1 repetitively.

Resistance to de-synchronization attack In Song's secret update protocol, the server sends (r_1, M_1, M_2) to update the tag's secret t_i to t_i', where $M_1 = f_{t_i}(r_1) \oplus t_i'$, $M_2 = s_i \oplus (t_i' \gg l/2)$, and $h(s_i) = t_i$. Upon receiving this message, the tag retrieves $t_i' = M_1 \oplus f_{t_i}(r_1)$ and $s_i = M_2 \oplus (t_i' \gg l/2)$, and updates the tag's secret if $h(s_i) = t_i$. The deficiency of this protocol is similar to that of the SM protocol w.r.t. server authentication. Since M_1 and M_2 both contain the value of t_i' (note that $t_i' \gg l/2$ is essentially the same as t_i'), an adversary can exploit the relationship between the values of M_1 and M_2 to construct a valid message (r_1, M_1', M_2') without access to the tag's secrets, where $M_1' = f_{t_i}(r_1) \oplus t_i''$ and $M_2' = s_i \oplus (t_i'' \gg l/2)$. This process is illustrated below.

From $M_1 = f_{t_i}(r_1) \oplus t'_i$ and $M_2 = s_i \oplus (t'_i \gg l/2)$, one can derive

$$M_1 = [f_{t_i}(r_1)]_L \oplus [t'_i]_L \ \| \ [f_{t_i}(r_1)]_R \oplus [t'_i]_R$$

$$M_2 = [s_i]_L \oplus [t'_i]_R \ \| \ [s_i]_R \oplus [t'_i]_L$$

Therefore, the value of t'_i can be crossed out

$$[M_1]_L \oplus [M_2]_R = [f_{t_i}(r_1)]_L \oplus [s_i]_R \tag{3.9}$$

$$[M_1]_R \oplus [M_2]_L = [f_{t_i}(r_1)]_R \oplus [s_i]_L \tag{3.10}$$

From $M'_1 = f_{t_i}(r_1) \oplus t''_i$ and $M'_2 = s_i \oplus (t''_i \gg l/2)$, one can derive

$$M'_1 = [f_{t_i}(r_1)]_L \oplus [t''_i]_L \ \| \ [f_{t_i}(r_1)]_R \oplus [t''_i]_R$$

$$M'_2 = [s_i]_L \oplus [t''_i]_R \ \| \ [s_i]_R \oplus [t''_i]_L$$

Therefore, the value of t''_i can be crossed out

$$[M'_1]_L \oplus [M'_2]_R = [f_{t_i}(r_1)]_L \oplus [s_i]_R \tag{3.11}$$

$$[M'_1]_R \oplus [M'_2]_L = [f_{t_i}(r_1)]_R \oplus [s_i]_L \tag{3.12}$$

Combining equations (3.9,3.10,3.11,3.12), one can derive

$$[M_1]_L \oplus [M_2]_R = [M'_1]_L \oplus [M'_2]_R$$

$$[M_1]_R \oplus [M_2]_L = [M'_1]_R \oplus [M'_2]_L$$

An adversary can derive M_1' and M_2' from M_1 and M_2 using the above equations. This vulnerability can be eliminated by replacing $(t_i' \gg l/2)$ with $h(t_i')$ in computing of M_2. This is because t_i' and t_i'' cannot be crossed out by an adversary for deriving M_1' and M_2' from M_1 and M_2; this is also because (M_1, M_2) and (M_1', M_2') are functions of t_i' and t_i'', which are independently chosen by the server.

3.3.4 Performance

Our revision on the original protocols does not change their storage requirements, which are comparable to other protocols in the literature. The reader is referred to [61, 60] for more details regarding the storage requirements.

Regarding the computation load, as our revised mutual authentication protocol uses $M_3 = h(r_2) \oplus s$ instead of $M_3 = s \oplus (r_2 \gg l/2)$, both the server and the tag need to compute one more hash operation than the SM protocol.

In our revised secret update protocol, we use $h(t_i')$ instead of $t_i' \gg l/2$. Thus, both the server and the tag require one more hash operation than the Song's secret update protocol.

	SM	Revised SM	**S**ecret **U**pdate	Revised **SU**
Server	$(k+1)F$	$(k+2)F$	3F	4F
tag	3F	4F	3F	4F

Table 3.2: Computational Requirements

Table 3.2 compares the computational requirements of our revised protocols with the original protocols. F denotes a computationally complex function such as hash and keyed hash, and k is an integer between 1 and $2N$. It is stated in [61] that the SM scheme requires one less hash operation on the tag than the other solutions in the literature for efficiency reasons. As we have shown in this paper, however, such reduction introduces new security vulnerabilities and the vulnerabilities can be eliminated by using one more hash operation. It remains interesting to investigate what the lower bound is for the computational requirements of secure RFID protocols.

Since SM mutual authentication protocol and Song's ownership transfer protocol are asserted to provide the most security properties, it is meaningful to examine these protocols in detail.

3.4 Summary

In this chapter, we analyzed two typical protocols that are asserted to have the most desired security properties for RFID communications. We discovered that these protocols are vulnerable to a series of active attacks including server impersonation, tag impersonation, and de-synchronization. We proposed revised protocols to eliminate the vulnerabilities without violation of any other security properties. The storage and computational requirements are comparable to the existing solutions.

Besides security and privacy, efficiency is another important concern in RFID-enabled supply chains. In the next chapter, we design RFID protocols with better efficiency without compromising security and privacy properties.

Chapter 4

Efficient Mutual Authentication in RFID-Enabled Supply Chains

To achieve high security and high efficiency at the same time, we categorize RFID-tagged supply chain environments into two security levels and design an RFID-tagged supply chain system accordingly. In the relatively secure environment, our system is set to the weak security mode, and the tagged products can be processed in a highly efficient manner. While in the less secure environment, our system is tuned into the strong security mode so as to maintain a high level of security with its efficiency lower than that in the weak security mode. A set of RFID protocols are designed to enable the dual security modes.

4.1 Preliminaries

Assumptions We focus on the attacks conducted on the wireless communications between RFID readers and tags. The adversaries can be either supply chain outsiders or insiders (i.e., dishonest supply chain parties). The adversaries are assumed to have the power to listen in communication channels, counterfeit as a valid supply chain party, to initiate, delete, modify, or transfer messages between RFID tags and readers. We do not consider the physical attacks, denial of service attacks, and

side-channel attacks. We further assume that the communications between a supply chain party and its RFID readers, and the communications between supply chain parties are secure, which can be protected by standard security techniques without limitation of resources.

Architecture Our solution involves four types of entities as shown in Figure 4.1: (i) a supply chain manager as a trusted authority, denoted by *TA*; (ii) independent supply chain parties denoted by P_i; (iii) RFID readers collectively denoted by R_i and a back-end database denoted by D_i inside supply chain party P_i; and (iv) RFID tags denoted by T_j.

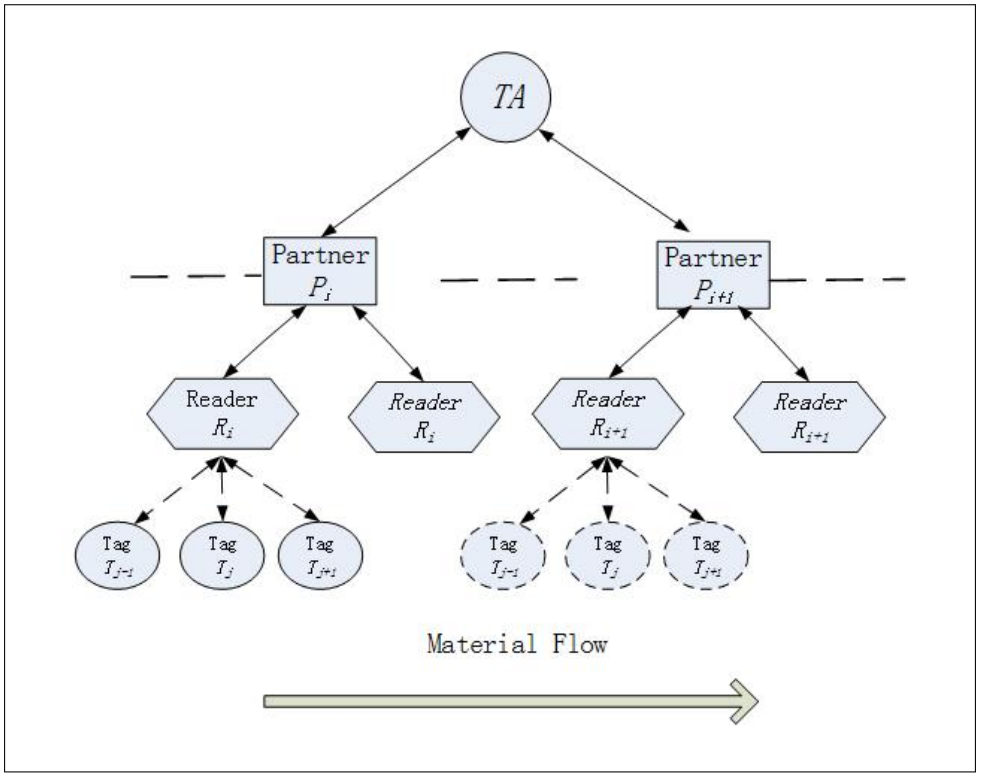

Figure 4.1: A Simplified RFID System Architecture

The architecture we proposed is suitable for various types of supply chain structures [47] and compatible with contemporary EPCglobal network architecture [5]. In particular, in a third-party logistics (3PL) supply chain, TA can be the shipping company, which is specialized in handling the shipping issues in the supply chain. In a vendor managed inventory (VMI) supply chain, the vendor manages all the

delivering of products; thus, it is straightforward for the vendor to take the role of TA. For a collaborative planning, forecasting, and replenishment (CPFR) supply chain, the supply chain hub is TA, which coordinates real-time sharing of supply chain information among supply chain parties. Finally, in a supply network (SN), the situation is similar but more complex than CPFR supply chain, where TA can be an existing supply chain hub or a dominant supply chain party.

4.2 Protocols

Since RFID tags will be used in vast numbers in supply chains, it is desired that the security design of the tags should be as cheap as possible. To achieve this goal, our protocols are designed to use passive tags that are equipped with pseudo-random number generators, XOR $\oplus$ and hash $H(\cdot)$ calculations.

A database is initialized by TA and sent to each supply chain party before the party can identify a batch of tags. With the help of the database, a supply chain party can switch security modes of RFID tags multiple times. Before a supply chain party sends a batch of tags to the next party in ownership handover, the current party needs to update the secret information in each tag. After ownership handover, the party can no longer identify the tags or track their movement.

4.2.1 Initialization

Tag initialization Before the first supply chain party P_1 starts processing tagged products, TA will initiate three values (α_j, $\beta_{1 \leftrightarrow j}$, $switch$) and embed them in each tag T_j in a secure manner.

- α_j is the tag root secret of length ℓ, which is fixed and shared between TA and the tag. The root secret is used to identify the tag uniquely.

- $\beta_{i \leftrightarrow j}$ is a temporary secret of length ℓ, which is shared between supply chain party P_i and tag T_j. The two parameters i and j of $\beta_{i \leftrightarrow j}$ denote the identity

number of the supply chain party and the tag separately. This secret is initiated by TA to be $\beta_{1\leftrightarrow j}$. The temporal secret $\beta_{i\leftrightarrow j}$ will be updated by P_i to $\beta_{i+1\leftrightarrow j}$ before ownership handover to P_{i+1}.

- $switch$ is a binary value used to indicate the security mode of a tag. This value is initiated to be 'on' for a strong security mode and it can be subsequently switched to 'off' for a weak security mode.

Database Initialization Each party P_i maintains a database D_i in its local storage where each tuple in the database corresponds to a tag. D_i contains all RFID information with respect to a batch of tags except for tag root secrets α. D_i consists of five attributes $(\beta, x, p, s, switch)$, where $(\beta, switch)$ are defined the same as in tag initialization, (x, p, s) are defined below.

- Tag response x: Tag response to be received from the corresponding RFID tag (in weak security mode). When the tag is on strong security mode, the value of this attribute is set to NULL.

- Pointer p: An octet string containing an address where the business information relevant to the tag is stored. An alternative approach is to store information in this field directly. Obviously, it trades the storage cost for communication efficiency.

- Status s: Binary bit; $s = 1$ means that the corresponding RFID tag has been processed; otherwise not. A database entry is denoted as "unmarked" if its status value is zero.

Reader initialization When the P_i is to handover a batch of tagged products to P_{i+1}, it updates the tag temporal secrets and informs TA that P_{i+1} is the next party. Then TA will distribute the database D_{i+1} to P_{i+1} through a secure channel (e.g., SSL).

4.2.2 Tag Reading

Upon receiving D_i from TA and the tagged products from P_{i-1}, supply chain party P_i can read any tag T_j in either the *strong security mode* if $switch$ is on or in the *weak security mode* if $switch$ is off.

1. $R_i \rightarrow T_j$: r_1, where r_1 is a random number of length ℓ generated by the reader.

2. $T_j \rightarrow R_i$: (r_2, x), where $x = H(r_1||r_2||\beta_{i\leftrightarrow j})$ and r_2 is a number of length ℓ. If the tag is in strong security mode, r_2 is a fresh random number generated by the tag; otherwise, $r_2 = 0$.

The tag reading protocol is illustrated in Figure 4.2.2. In the weak security mode, R_i can pre-compute the response x_j of each tag and store them in D_i. On receiving a response x, R_i identifies the tag if it can find a record $d_j = \langle \beta_{i\leftrightarrow j}, x_j, s_j, switch_j \rangle$ in D_i such that $x_j = x$ and $s_j = 0$. In the strong security mode, however, the response of each tag cannot be pre-computed due to the use of fresh random number in tag; the value of x is set to NULL in each tuple of D_i in this case. Given a response (r_2, x), the reader identifies a tag by searching all of the unmarked tuples in D_i until it finds a tuple d_j satisfying $x = H(r_1||r_2||\beta_{i\leftrightarrow j})$. For each identified tag T_j, the reader sets its status $s_j = 1$. If the pointer p_j is not empty, the reader can obtain relevant product information following the pointer.

The above reading process can be performed multiple times by supply chain party P_i if necessary.

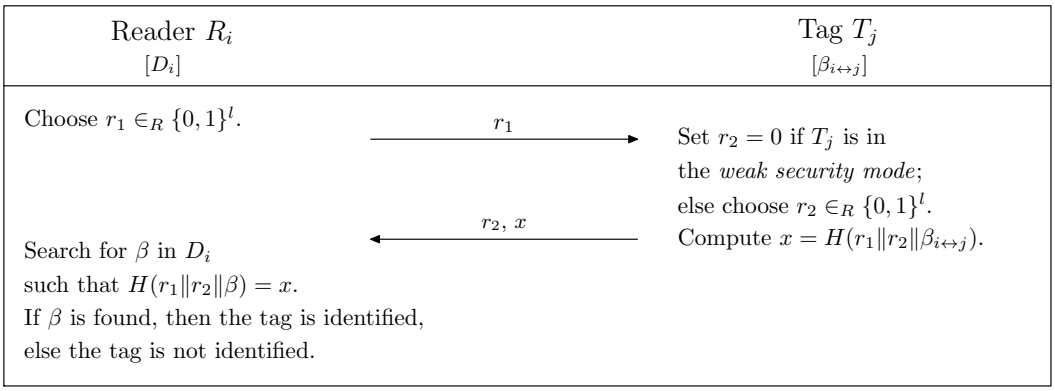

Figure 4.2: Tag Reading Protocol

4.2.3 Security Mode Switching

Once the party P_i receives D_i from TA, it can change the security mode of its tags in different environments. Although the strong security mode is secure against both active attacks and passive attacks, the RFID tags can be processed more efficiently in the weak security mode in an environment where the active attacks are impossible. We design a security mode switching protocol below.

1. $R_i \rightarrow T_j$: To update the security mode of tag T_j, reader R_i chooses a fresh random number r_3 of length ℓ and computes $a = \beta_{i\leftrightarrow j} \oplus r_3$, and $b = H(switch_0||a||r_3)$, where $switch_0$ is the new value of $switch$. The reader then sends the triple $(switch_0, a, b)$ to tag T_j.

2. $T_j \rightarrow R_i$: When tag T_j receives $(switch_0, a, b)$, it computes $r_3 = \beta_{i\leftrightarrow j} \oplus a$, and checks whether $b = H(switch_0||a||r_3)$ holds; if so, it updates $switch = switch_0$. After update of switch value, the tag will send a confirmation (r_2, x) back to the reader, where r_2 is generated based on the switch value and $x = H(r_2||r_3||\beta_{i\leftrightarrow j})$.

3. R_i: Upon receiving (r_2, x), reader R_i confirms the update of $switch$ by checking whether $x = H(r_2||r_3||\beta_{i\leftrightarrow j})$.

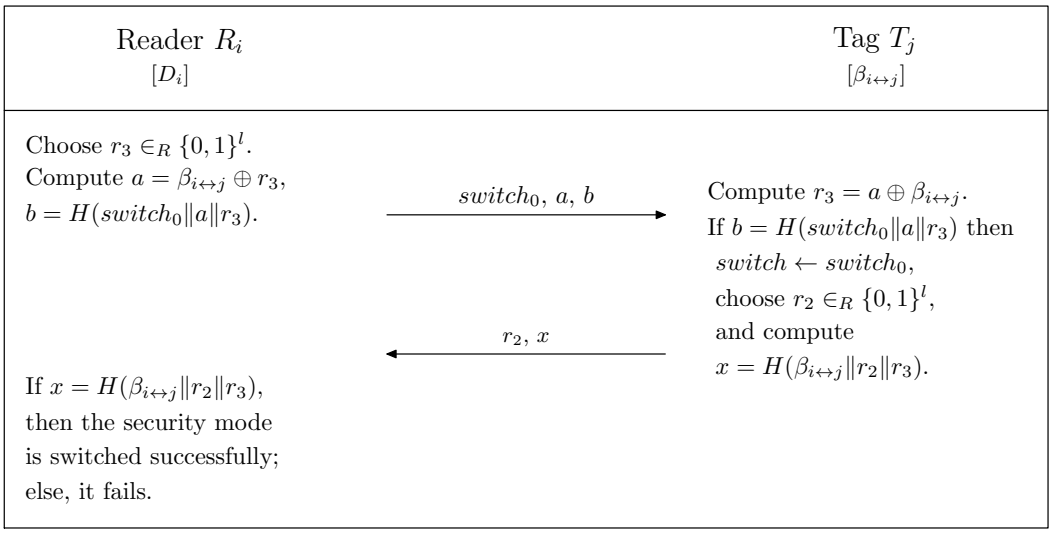

Figure 4.3: Security Mode Switching Protocol

The protocol is illustrated in Figure 4.2.3. Since a tag will send a confirmation to the reader after the update of its security mode, any failure can be detected by the reader.

4.2.4 Ownership Handover

Ownership handover is performed between two supply chain parties P_i and P_{i+1} with RFID tags in weak security mode without TA's active involvement. Before the handover, P_i will update the temporal secrets of its tags and informs TA, who will send D_{i+1} to P_{i+1} in a secure manner.

In order to prevent the tagged products from being tracked by party P_i after ownership handover, the tag's temporary secret must be updated from $\beta_{i\leftrightarrow j}$ to $\beta_{i+1\leftrightarrow j}$. This updating process is performed by P_i before handover. Without being appropriately updated, a tag will not be accepted by P_{i+1} in the handover process (see below). The update of tag temporal secrets guarantees that only P_{i+1} can access the tags although the update is conducted by P_i. This is under the assumption that P_i cannot get access to tag root secrets nor the new database D_{i+1}. After the update,

P_{i+1} cannot track the tags' previous sessions as it does not have D_i. The temporary secret updating protocol is shown in Figure 4.4.

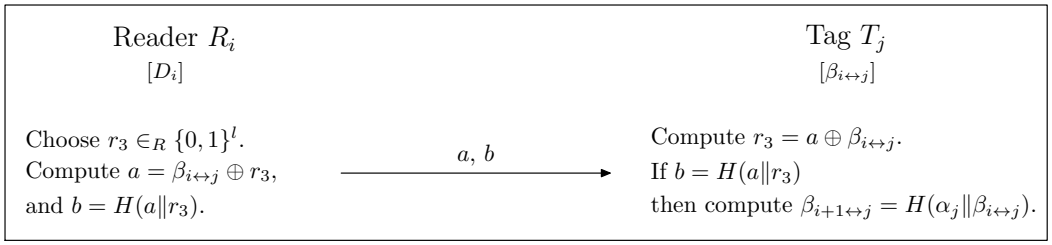

Figure 4.4: Temporary Secret Updating Protocol

During ownership handover, both parties need to agree upon a list of the tagged products and report the agreed list to TA for supply chain visibility.

1. For a batch of tagged products to be handed over to P_{i+1}, P_i performs tag reading protocol with the same random number r_1 of length ℓ and records a list L of responses x_j from all tags in the batch, where $x_j = H(r_1||r_2||\beta_{i+1\leftrightarrow j})$, $r_2 = 0$. Then, P_i sends r_1 to P_{i+1}.

2. Upon receiving r_1, P_{i+1} performs the reading protocol with the same random number r_1 and records a list L' of all responses x'_j from all tags in the batch, where $x'_j = H(r_1||r_2||\beta_{i+1\leftrightarrow j})$.

3. P_i and P_{i+1} compares the two list L and L'. If the lists match, then both sign on the matched list with the current time-stamp and keep a copy of the signed list. Party P_i sends the signed list to the TA for supply chain visibility. If the two lists do not match, the two parties will settle the disagreement till they reach an agreement. After the handover process, P_{i+1} should switch the tags into the strong security mode if P_i is still around.

The ownership handover process is illustrated in Figure 4.5. Note that P_i is responsible to report to TA since it is for P_i's interest to finalize the handover process as early as possible. TA is responsible to coordinate the handover process and

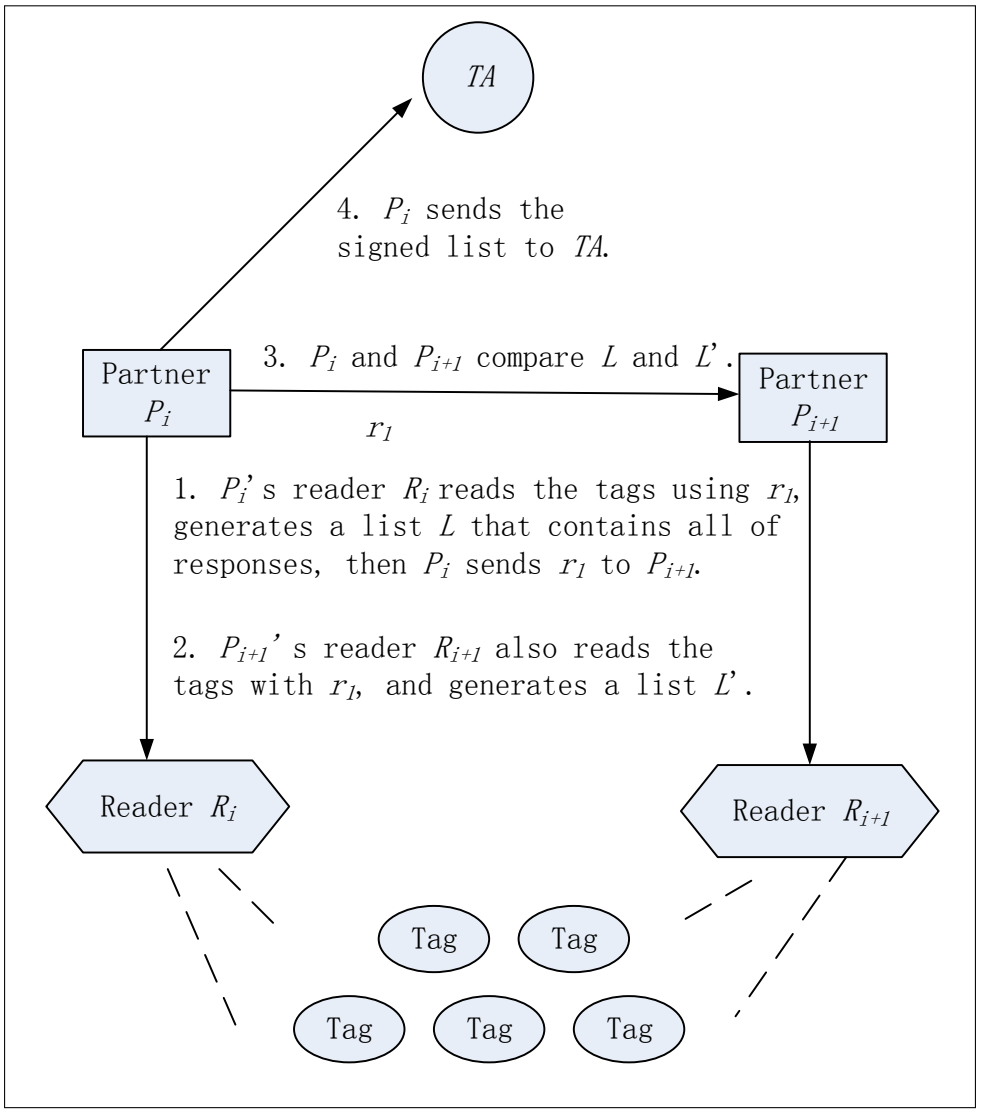

Figure 4.5: Ownership Handover Process between P_i and P_{i+1}

manage the supply chain visibility accordingly. Our system remains secure even the tags are set to the weak security mode in the ownership handover process. The reason is that the tagged products remain static in this process; there is no point to track tags while they are not moving. After ownership handover, the tags are in P_{i+1}'s control, who will keep the tags secure by switching to appropriate security modes. If P_i has not updated some tags appropriately before ownership handover due to de-synchronization attacks or communication errors, both parties will detect the mismatch; P_i can re-update the tags to facilitate the handover. Therefore, our solution has the de-synchronization resilience property.

4.3 Analysis

Supply chain visibility motivates the adoption of RFID technology in supply chain management, as it enables real-time track and trace of tagged products. Our protocols realize the supply chain visibility by introducing a trusted authority (TA), which can be seamlessly incorporated in various supply chain architectures as discussed in Chapter 2. In our solution, TA is responsible to generate and distribute tag temporal secrets to legitimate supply chain parties so as to enable tag reading and updating. TA is also reported with each batch of tags right after the tagged products are successful handed over from one supply chain party to another. Therefore, TA is able to maintain supply chain visibility and address any dispute among supply chain parties. It is straightforward for TA to make the supply chain visibility service available to any authorized entities across the Internet via EPCglobal Network mechanisms [5]. In particular, TA plays a role similar to EPC discovery service (D-S), which can answer any authorized query (after authentication and access control) about which supply chain party takes hold of a tag at a particular time. It may return the address of EPC information service (IS), which is a database server maintained by the corresponding supply chain party, to the querier. If the querier wants to know more detailed information about the tag such as the events happening within the corresponding supply chain party, it can further query the EPC information service following the address returned by TA.

While the requirement on supply chain visibility is satisfied, it is important to achieve high security and efficiency in protocol design. In the following, we analyze our protocols with respect to the requirements of security and efficiency.

4.3.1 Security

We state that our protocols satisfy the security requirements for RFID-tagged supply chains as enumerated in Section 2.2.

STATEMENT 4.1 (Authoritative access to RFID tags) *Only a valid read-*

er with a tag's temporary secret authorized by TA is able to conduct reading and updating on the tag successfully.

First, consider the tag reading protocol for a tag with temporal secret $\beta_{i\leftrightarrow j}$. Upon receiving any challenge r_1 from a reader, the tag responses with (r_2, x), where $x = H(r_1||r_2||\beta_{i\leftrightarrow j})$. It is computationally difficult for a polynomial-time bounded adversary to calculate the temporal secret $\beta_{i\leftrightarrow j}$ given x, r_1 and r_2 due to the one-way property of hash function. Since $\beta_{i\leftrightarrow j}$ is the only information used to identify the tag in D_i, no polynomial-time bounded adversary can conduct reading on the tag successfully.

In temporary secret updating protocol, message (a, b) is sent from the reader to the tag, where b is essentially an authenticator of a based on a shared key $\beta_{i\leftrightarrow j}$ between an authorized reader and the tag. Due to the weak collision resistance of hash function, only with the knowledge of r_3, can the reader compute $b = H(a||r_3)$. It implies that the originator of (a, b) has the possession of the matching key $\beta_{i\leftrightarrow j}$, which is used to generate r_3 from a. Upon verifying b from a based on $\beta_{i\leftrightarrow j}$ on the tag side, only the commands from a authorized reader will be accepted by the tag. It is straightforward to extend this proof to the security mode switching protocol.

Next, we consider the authenticity of a tag, which is a desired property for the tag reading protocol.

STATEMENT 4.2 (Authenticity of tags) *Given two numbers r_1 and r_2, the probability for an adversary without the knowledge of $\beta_{i\leftrightarrow j}$ to find a valid response x such that $x = H(r_1||r_2||\beta_{i\leftrightarrow j})$ holds is $\frac{1}{2^\ell}$, where ℓ is the length of $\beta_{i\leftrightarrow j}$.*

The number of all the possible values of $\beta_{i\leftrightarrow j}$ is 2^ℓ. Obviously, without the knowledge of the tag temporal secret $\beta_{i\leftrightarrow j}$, the probability for an adversary to choose the right value of $\beta_{i\leftrightarrow j}$ that yields a valid response $x = H(r_1||r_2||\beta_{i\leftrightarrow j})$ given r_1 and r_2 is $\frac{1}{2^\ell}$ in random tries, which are clearly impractical if ℓ is reasonably large (e.g., 100 bits). On the other hand, it is computationally infeasible for the adversary to derive the tag temporal secret $\beta_{i\leftrightarrow j}$ from an intercepted value x due to

the one-way property of hash function.

In fact, the protocols are built by referring to typical challenge-response authentication protocol, which defends them against both passive attacks like eavesdropping and replaying protocol messages, and active attacks like Man-in-the-middle attacks. Thus, for statements 4.1 and 4.2, we claim that both properties, authoritative access and authenticity, hold under the security assumptions we drawn in Section 2.2 and Section 4.1.

Next, we consider the unlinkability property of our proposed solution. Unlinkability is a critical security requirement in supply chains due to the existence of insider attacks (i.e., attacks launched by upstream or downstream supply chain parties). The adversaries of who may violate the unlinkability requirement include both supply chain otusiders and insiders; in particular, the adversaries of the tags being processed by supply chain party P_i can be anyone except for P_i.

STATEMENT 4.3 (Weak unlinkability) *Given a response x_1 from a tag prior to being processed by party P_i and a response x_2 from a tag after being processed by P_i, it is computationally infeasible for an adversary to determine whether x_1 and x_2 is from the same tag.*

Without loss of generality, let $x_1 = H(r_1||r_2||\beta_{i\leftrightarrow j})$ and $x_2 = H(r_1'||r_2'||\beta_{i+1\leftrightarrow j})$ be two responses from a tag being queried before and after it is processed by party P_i, respectively, where $\beta_{i\leftrightarrow j}$ is the tag's temporary secret before it is processed by P_i, and $\beta_{i+1\leftrightarrow j}$ is the updated temporary secret after it is processed by P_i. Since the current tag temporary secret $\beta_{i\leftrightarrow j}$ is updated by P_i before it is handed over to P_{i+1}, we have $\beta_{i\leftrightarrow j} \neq \beta_{i+1\leftrightarrow j}$; because of this, neither P_i (who knows $\beta_{i\leftrightarrow j}$ only) nor P_{i+1} (who knows $\beta_{i+1\leftrightarrow j}$ only) can make a successful link between x_1 and x_2.

STATEMENT 4.4 (Strong unlinkability) *A tag is strong unlinkable in the strong security mode. It is also strong unlinkable in the weak security mode in an environment of no active attacks.*

Without loss of generality, let $x_1 = H(r_1||r_2||\beta)$ and $x_2 = H(r_1'||r_2'||\beta')$ be two

responses from a tag at any different times. In the strong security mode, the tag generates a new random number each time it is read. Even if an adversary queries the same tag before its update and with the same reader nonce r_1 (i.e., $\beta = \beta', r_1 = r_1'$), he or she still cannot link x_1 and x_2 due to the use of different tag random numbers. Therefore, the tags are strong unlinkable in the strong security mode.

In the weak security mode, we have $r_2 = r_2'$. In an environment of no active attacks, an adversary can only listen to the RFID communications that are initiated by a valid reader, in which case $r_1 \neq r_1'$. Consequently, the adversary still cannot link x_1 and x_2 due to the use of different reader random numbers.

Combining statements 4.3 and 4.4, we can conclude that the tags are strong unlinkable in our solution if we switch the security modes appropriately.

STATEMENT 4.5 (Forward and backward secrecy) *If the protocol communication between a tag and a reader is compromised in certain party, it will not affect the security of the protocol communication between the tag and the reader in any other parties.*

Since a tag's temporal secret is updated right before each ownership handover, any supply chain party, without given this new temporal secret by TA, cannot identify or track the updated tag in RFID communications. This is because the new temporal secret is generated by hashing the current temporal secret in concatenation with the tag's root secret, which is shared between the tag and TA only. Due to the one-way property of hash function, it is computationally infeasible for an adversary to derive the root secret from a temporal secret by compromising the communication between the tag and the current party. Therefore, the adversary still cannot derive the updated temporal secret given the current temporal secret.

STATEMENT 4.6 (De-synchronization resilience) *The temporary secret updating protocol and the security mode switching protocol in our security solution are resilient to de-synchronization attacks.*

In our security solution, two protocols, the temporary secret updating proto-

col and the security mode switching protocol, are used to update the content of RFID tags and back-end databases. This gives an adversary the opportunity to de-synchronize the update processes such that the content in the tag and the content in the back-end database do not match. In the temporary secret updating protocol which is performed right before ownership handover, if some tag is not updated appropriately, a mismatch will be detected during handover process when P_i and P_{i+1} compare their lists. To address the problem, P_i can repeat its temporary secret updating protocol for the detected tag until a match is achieved. In the security mode switching protocol, a tag must send a confirmation message after its value of $switch$ is successfully updated by a reader; any failure due to de-synchronization attacks can be detected by the reader if it cannot receive the correct confirmation.

4.3.2 Efficiency

In supply chain management, the tagged products are most likely to be processed by batch instead of by item. Our analysis on the efficiency focuses on batch processing. It is more practical and efficient for an authorized reader to use the same random number in the interactions with a batch of tags in one instance of processing (including read, update, and ownership handover). This is because the tags in the same batch are in the same waiting state when queried. The reader can simply send its random number to all tags in the batch (instead of sending a different random number to each single tag) and wait for their responses. Note that this will not downgrade the security properties because the tags in the same batch are usually in the same waiting state, putting closely to the authorized reader; thus, it is meaningless for an adversary to eavesdrop the repetitively used random number, query a not-yet-processed tag in the batch, and impersonate it. It is important though, that the same random number should not be used by other authorized reader to process the same batch of tags before ownership handover.

For convenience of analysis, we assume that the reader will divide a batch of

n tags into m sessions (though our efficiency results are irrelevant of m). If the reader is able to process all of the products in one pass, the value of m would be one. For simplicity, assume that the number of the tags in each session is n/m. We assume that each database D_i is used for a single batch of tags and that different databases are used for different batches. The same random number r_1 will be used by an authorized reader in interaction with the tags in different sessions of the same batch.

STATEMENT 4.7 *In the weak security mode, the time complexity for an authorized reader to identify a batch of n tags is $O(n \log n)$. While in the strong security mode, the time complexity is $O(n^2)$.*

First, consider the weak security mode. With the database D_i, an authorized reader can pre-compute the response x of each tag in the batch given the same random number r_1, where $x = H(r_1||r_2||\beta_{i \leftrightarrow j})$ and $r_2 = 0$. The time complexity of this computation is $O(n)$. The reader then sorts the database records according to x before interacting with the tags. The time complexity for sorting the database is $O(n \log n)$ and the space complexity is $O(\log n)$. With the sorted database, the reader can identify a tag comparing the tag's reply with the unmarked database records in a sorted order. The reader requires $\log n + \log(n-1) + \ldots + \log(n - n/m + 1)$ comparisons in binary search for the first session, and requires $\log(n - n/m) + \log(n - n/m - 1) + \ldots + \log(n - 2n/m + 1)$ comparisons for the second session (if any), and so on. Overall, the reader needs to conduct $\log n + \log(n-1) + \ldots + \log 1 = \log n!$ comparisons to identify the whole batch. The time complexity for identifying a batch of n tags is $O(n \log n)$. The space complexity for sorting is $O(\log n)$. Next, consider the strong security mode, in which the tag will generate a fresh random number r_2 in each interaction with a reader. The reader cannot pre-compute tag responses due to the use of fresh r_2. To identify a tag, the reader needs to search all the records in D_i one by one until the right one is discovered. In the average case, the time complexity of identifying the first tag in a batch of n tags is

$1/n * (1 + 2 + \cdots + n) = \frac{1}{n} * \frac{n(n+1)}{2} = \frac{n+1}{2}$. The expected time to identify the jth tag in a batch is $\frac{n+1-j}{2}$. Therefore, the time complexity of processing the whole batch is $O(n^2)$. No extra space consumption is required in this process.

The tag temporary secret updating protocol and the security mode switching protocol should be performed for each session of tags right after the tags are identified in the tag reading protocol. The updating of the tags and the verification of tag confirmations can be performed one tag by one tag; thus, the time complexity is $O(n)$ for processing the whole batch.

In ownership handover, each of the two involving parties needs to query a batch of tags using the same random number and obtain a list of tag responses. There is no need to identify these tags. The two involving parties then sort their lists and compare them before they sign an agreed list and send it to TA. The time complexity is $O(n \log n)$ for each party to sort a list, and $O(n)$ for comparing the lists. What the TA needs to do is to verify the signature of the signed list. Our ownership handover protocol is efficient in a sense that TA is not directly involved in the handover process.

The bottleneck of most RFID-tagged supply chain systems including ours is the process of identifying a large number of tags by each reader. According to [48, 66], we roughly estimate that it takes about 210 CPU cycles of a Pentium CPU to perform a hash function (e.g., SHA-1) for digesting a 128-bit message and about 40 CPU cycles per sort operation for merge-sort or quick-sort algorithms. We assume that there are 2^{20} tags in each batch, and a 1-GHz Pentium machine is used in each reader's servant computer. In the weak security mode, it requires about 800ns in database search for identifying each tag. In the strong security mode, however, the batch size is better below 10^4 so that a reader can identify about 500 tags per second. Since an RFID reader usually can perform about 100 times of reading, the speed of searching tags in database is sufficiently fast enough as it is higher than 100 tags per second. Since the tags are usually attached to pallets, cases, or containers

in supply chains, 10^4 can be a rational upper bound for the batch size in practice.

4.3.3 Comparisons

We compare our solution with the most related works [43, 35, 60] on RFID authentication protocols in supply chain environments. We realize that although these works might have been attacked in some way [62], their original ideas are still meritable and worth being reviewed. A summary of our comparison is given in Table 4.1.

Table 4.1: Comparisons with Existing Solutions

Solutions	Unlinkability (anti-tracking)	Visibility (handover)	Efficiency (tag search)	Cost (tag)
[43]	Weak	Distributed	Batch process	Moderate
[35]	Null	Distributed	Decryption	Low
[60]	Strong	Distributed	Tag by tag	Moderate
Our solution	Strong	Centralized	Switch	Moderate

In [43], Li and Ding proposed a de-centralized solution for secure RFID communications in supply chains. Their solution is similar to our solution in the weak security mode in a sense that only weak unlinkability is provided. The time complexity of their solution is also similar to our solution in the weak security mode, which is $O(n \log n)$ for processing each batch of tags. Since there is no trusted authority involved in their solution, the supply chain visibility should be maintained by each party's database in a distributed manner.

Juels, Pappu, and Parno proposed an interesting solution for secure RFID-tagged supply chains [35]. In their work, a secret sharing method is used to break a secret key to multiple shares, with each share stored in a single tag along with the cipher of the tag id, which is encrypted with the secret key. By getting access to a large number of tags, an authorized supply chain party can collect enough shares to recover the secret key, and thus decrypt the tags' IDs. An adversary is assumed to have limited access to the tags; thus, he or she cannot recover the secret key nor decrypt any tags' IDs. It is clear that this solution does not have any unlinkability

feature. Anyone can track the movement of a tag even if it is encrypted. The advantage of this solution is that it can be directly used with the current EPC Gen2 tags [32] without any cryptographic extensions; therefore, the cost of tags is apparently lower than other solutions which have to incorporate hash computation and random number generation in tags.

Song proposed an RFID ownership transfer protocol recently [60]. The weakness of this solution is its low efficiency, especially in the handover process which takes $O(n^2)$ time for processing n tags one by one.

Comparing to the above works, our proposal is the only solution that involves a trusted authority. Therefore the supply chain visibility can be easily maintained in a centralized manner, as it is required in EPCglobal Network through discovery service. On the one hand, our solution provides strong unlinkability in both strong security mode and weak security mode, under the assumption that the weak security mode is used in a relative secure environment of no active attacks, in which case a higher efficiency in tag reading can be achieved. It thus provides higher efficiency in certain environment without downgrading the security features. In terms of tag cost, our solution is similar to [43, 60] as it involves hash computation and random number generation in tags. Note that our solution is suitable for the RFID tags of cost around US$0.5 and RFID reader of cost around US$1000. Such RFID readers and tags are currently available in the market and their costs are affordable in supply chain management at container, pallet, or case level (probably not at the item level).

4.4 Summary

In this chapter, we investigated how to achieve high security and high efficiency while maintaining visibility for RFID-tagged supply chains. Supply chain security (which protects tracking) and visibility (which enables tracking) seem to conflict to each other. The key to solve the apparent dilemma is to distinguish two types of entities: unauthorized entities who are prevented from tracking the movement of

material flow, and authorized entities who are provided with supply chain visibility. In our design, when a tag is processed by a supply chain party, neither other supply chain parties nor supply chain outsiders can link any two communications between the tag and the current supply chain party, while a trusted authority facilitates the handover of tags between supply chain parties and provides visibility services. On the other hand, high efficiency is particularly desirable in RFID-tagged supply chains since a large quantity of tagged products are routinely processed and exchanged among multiple supply chain parties such as suppliers, manufacturers, distributors, and retailers. In order to enhance the efficiency of an RFID-tagged supply chain systems, we distinguish the environments into two secure levels. In a relatively secure environment with no active attacks, our RFID system can be set to the weak security mode so as to provide high processing speed without sacrificing its security. While in a relatively less secure environment that is exposed to active attacks, our RFID system can be switched to the strong security mode so as to maintain its high security with lower processing speed.

Till now, we have introduced our work in RFID-enabled supply chains where all the readers are trusted to know tag secrets. The protocols proposed in Chapter 3 and Chapter 4 are not suitable for third-party logistics, where three parties are involved to process tagged products. In the next chapter, we propose secure and privacy-preserving tag-authentication protocols for third-party logistics under the assumption that not all supply chain parties are fully trusted.

Chapter 5

Three Party Authentication in RFID-Enabled Supply Chains

"Symmetric secret"-based RFID systems are widely adopted in supply chains. In such RFID systems, a reader's ability to identify a RFID tag relies on the possession of the tag's secret which is usually only known by its owner. If a "symmetric secret"-based RFID system is deployed in third party logistics (3PL) supply chains, all the three parties (the sender of the goods, the receiver of the goods and the 3PL provider) should have a copy of those tags' secrets to access the tags. In case the three parties in 3PL supply chain are not fully trusted, sharing the secrets among the three parties may cause security and privacy problems. To solve these problems, we firstly formalize the security and privacy requirements for 3PL supply chains in the presence of internal adversaries as well as external adversaries. Then we propose two different protocols which satisfy the requirements: one is based on aggregate message authentication codes, and the other is based on aggregate signature scheme. After comparing the two protocols in terms of performance and usability, we conclude that the aggregate MAC-based solution is more applicable in 3PL supply chains.

5.1 Designing Principles

Adopting "symmetric secret"-based RFID systems in 3PL supply chains requires the three parties to share tag secrets. Considering the existence of internal adversaries, sharing the secrets among three parties is problematic since having a tag's secret means having the ability to fabricate it. If all the three parties have the ability to fabricate the tags, disputes on the originality of the tags are difficult to solve.

Our method is to authorize each valid party with a credential instead of the secrets. The credential can be used to check the status of the goods. At the same time the credential should not reveal any information about the tags. In a 3PL supply chain, Party A is the tags' owner. Only Party A possesses the tags' secrets. Party A grants a credential $credential_C$ to Party C so that Party C can check the tags' existence during the transportation. Party A grants a credential $credential_B$ to Party B, so that Party B can use it to verify the goods. The credential construction and using is a subtle work.

Recall that there are two requirements required against internal adversaries, namely, restraining Party C and protecting Party C. Restricting Party C requires that with $credential_C$, Party C cannot get any information of the tags. Protecting Party C requires that Party C should be able to confirm that the tags will pass the verification according to $credential_B$ before taking over the goods from Party A.

Hence a systematic scheme should be designed for the three parties to make an agreement on the credentials. And considering outside adversaries, the system should make sure that external adversaries cannot forge the tags that pass the verification according to the credentials.

Our work contains two parts: 1) designing a protocol that enables an authorized party to verify the tags with a credential; 2) designing a scheme that enables the three parties to make an agreement on the credentials. We observe that in 3PL supply chains, normally, the goods are checked on batch level. In the following, we provide two group checking protocols to enable an authorized party to verify

the tags according to a credential on batch level based on two different credential designing schemes. With any one of our group checking protocols, any change to the tags is detectable. If the tags are not changed, their originality cannot be denied.

5.2 Solution Based on Aggregate MAC Scheme

Our first solution is based on an aggregate MAC scheme proposed in [37]. The intuition of this proposal is: each tag T_i (with k_i as its individual secret) is deployed with a MAC function. The authorized reader is granted with a credential that contains several couples of m_j and the aggregate MAC value $Agg(m_j)$ on m_j under each tag's key, where $1 \leq j \leq d$, d is the number of the pairs. The reader chooses an unused pair $(m, Agg(m))$, and uses m to query all the tags, each tag replies with the MAC value on m under its key. Upon receiving all tags' replies, the reader aggregates them and compares the aggregated value with $Agg(m)$, if they are the same, then the tags are intact, else, there are not all original ones.

5.2.1 Building Blocks of Our MAC-based Solution

MAC: In cryptography, a message authentication code (MAC) is a short piece of information used to authenticate a message. A MAC algorithm, sometimes called a keyed (cryptographic) hash function $h_k(\cdot)$, it accepts the inputs as a secret key k and an arbitrary-length message m to be authenticated, and outputs a MAC tag $t = h_k(m)$ (sometimes known as a tag). The MAC value protects both the message's data integrity and its authenticity by allowing verifiers (who also possess the secret key) to detect any changes to the message content.

Aggregate MAC: In [37], Katz and Lindell proposed and investigated the notion of aggregate message authentication codes (MACs) which has the property that multiple MAC tags, computed by (possibly) different senders on multiple (possibly different) messages, can be aggregated into a shorter tag that can still be verified by

a recipient who shares a distinct key with each sender. The aggregation is done by computing XOR of all the individual MAC tags. They proved that if the underlying MAC scheme is existentially unforgeable under an adaptive chosen-message attack and is deterministic, then the aggregate message authentication code generated by computing the XOR of every individual MAC values is secure.

Commitment Scheme: A commitment scheme allows one to commit to a chosen value (or chosen statement) while keeping it hidden to others, with the ability to reveal the committed value later [28]. A commitment protocol for value v can proceed as follows: $c = H(v, r)$, where c is the commitment value, H is a cryptographic hash function that is one-way, collision-free, and has pseudo-random output if r is a random unpredictable nonce. To open the commitment, v and r are disclosed. The collision resistance property of H ensures that it is computationally infeasible to find another v or r that will result in the same commitment c. Note that if v is unpredictable (e.g., a freshly generated nonce), the additional nonce value r is not needed and we simply have $c = H(v)$ [24].

5.2.2 Aggregate MAC-based Solution

Requirements of the tags: The tags should be able to perform a MAC function $h_k(\cdot)$ under a key k stored in the tag. The tag should contain a random string generater.

Initialization: Party A initializes the tags. Suppose there are n tags in the system. Each tag T_i stores two secrets (b, k_i), $1 \leq i \leq n$, b is a common group secret that is shared by all the tags, and k_i is the tag T_i's individual secret.

Authorization to a valid party: Party A keeps k_i secret, and grants to a valid party the group secret b and a credential. d denotes the estimated upper bound of the number of times that the party will check the goods. The credential contains d pairs of $(m_j, Agg(m_j))$, $Agg(m_j) = \bigoplus_{i=1}^{n} h_{k_i}(m_j)$, for $1 \leq j \leq d$ and $1 \leq i \leq n$.

Group checking protocol for an authorized party: For each checking, the autho-

rized party chooses an unused pair $(m_j, Agg(m_j))$ from the credential, then uses m_j to query all the tags. The details are depicted as below. Figure 5.1 illustrates the protocol.

1. *Reader* $\rightarrow$ *Tag* T_i*:* The reader sends m_j to the tag T_i.

2. *Tag* T_i $\rightarrow$ *Reader:* On receiving m_j, T_i chooses $r_2 \in_R \{0,1\}^l$, l is the system parameter. T_i computes $M_1 = h_{k_i}(m_j) \oplus h(r_2)$, $M_2 = b \oplus r_2$, then sends (M_1, M_2) to the reader.

3. *Reader:* On receiving (M_1, M_2), the reader computes $r_2 = M_2 \oplus b$, and computes $t_i(m_j) = M_1 \oplus h(r_2)$. The reader stores the value of $t_i(m_j)$.

- After getting all the values $t_i(m_j)$, the reader checks whether $\bigoplus_{i=1}^{n} t_i(m_j) = Agg(m_j)$. If $\bigoplus_{i=1}^{n} t_i(m_j) = Agg(m_j)$, then the reader confirms the existing of the tags. If $\bigoplus_{i=1}^{n} t_i(m_j) \neq Agg(m_j)$, then the tags are not all the original ones.

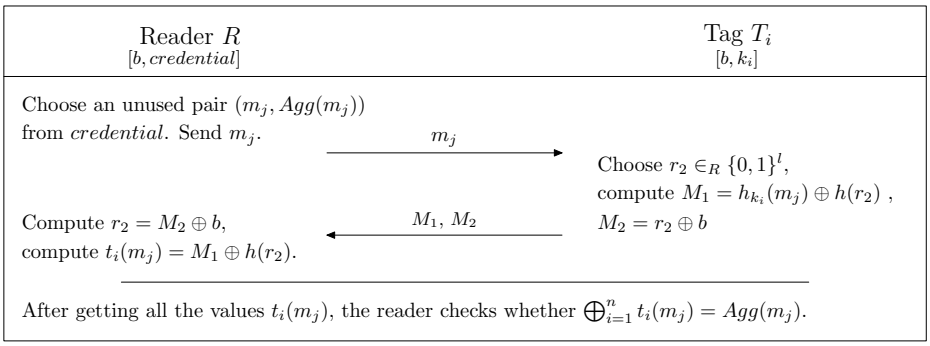

Figure 5.1: Aggregate MAC-Based Solution

Party A gives credentials $credential_B$ and $credential_C$ respectively to Party B and Party C. Party B uses $credential_B$ to verify the goods when taking over them from Party C. Party C uses $credential_C$ to verify the tags during the transportation. In aggregate MAC-based scheme, Party B and Party C should keep its own credential in secret to each other, and $credential_B$ and $credential_C$ should not contain

same $(m, Agg(m))$. Party C should make sure that the original tags will pass the verification using $credential_B$.

5.2.3 Analysis of the MAC-based Solution

We first analyze the security and privacy properties of the MAC-based protocol against external adversaries.

- *Tag Information Privacy:* Without knowing b, the adversary cannot calculate the value of r_2. Without r_2, the adversary cannot get the value of $h_{k_i}(m_j)$. Then without $h_{k_i}(m_j)$, the adversary cannot get any information about k_i.

- *Tag location Privacy:* Without common secret b and individual secret of each tag, due to the cryptographic property of the MAC function $h_k(\cdot)$, the adversary cannot get any information of the tags through (M_1, M_2).

- *Resistance of tag impersonation attack:* Given (M_1, M_2), the adversary cannot retrieve any information of b and k_i. Without the knowledge of b and k_i, the adversary cannot retrieve any information about the tag. The probability that the adversary successfully impersonate a tag is equal to the probability that the adversary randomly chooses (M_1', M_2') and then (M_1', M_2') together with other valid tags' replies pass the verification.

- *Resistance replay attack:* The authenticated reader uses fresh pair of $(m_j, Agg(m_j))$ to verify the tags in each checking, so that the attacker cannot reuse the tags' replies from previous sessions.

Then we analyze the security and privacy properties of the MAC-based protocol against internal adversaries.

- *Protect Party C:* For Party C, given a tag T_i, with the common secret b, it can challenge the tag with arbitrary message m and get the MAC tag $t_i(m) = h^{k_i}(m)$ on m with T_i's secret k_i, however it cannot compute the value of k_i

if the underlying MAC scheme is secure. While without the knowledge of k_i, based on the security of the aggregate MAC scheme, it is computational impossible to forge the tags so that given another message m', the aggregation of the tags' replies $\bigoplus_{i=1}^{n} t_i(m')$ equals $Agg(m') = \bigoplus_{i=1}^{n} h_{k_i}(m')$. Hence, if the tags pass the verification using Party B's credential, then Party B cannot deny that the tags are original ones.

- *Restrain Party C:* Providing some pairs of $(m, Agg(m))$, together with group secret s, Party C can verify the tags on batch level. However, given a tag T_i, although Party C can get the value $h_{k_i}(m)$ on any message m, it cannot compute the value k_i. Hence Party C cannot obtain any extra information of the tag.

5.2.4 Discussions

Here we provide a solution sketch for the three parties to make an agreement on the credentials. Upon reaching an agreement, all the three parties should sign on the agreement.

When a transaction starts, Party A gives credentials $credential_C$ and $credential_B$ to Party C and Party B respectively. Suppose there are n tags in the system. Each tag T_i stores two secrets (b, k_i), where b is a common group secret shared by all the tags and k_i is T_i's individual secrets. Suppose a party will check the tags without exceeding d times, a credential for the party should contain the group secret b and d pairs of $(m_j, Agg(m_j))$ where m_j is a fresh nonce and $Agg(m_j) = \bigoplus_{i=1}^{n} h_{k_i}(m_j)$, for $1 \leq j \leq d$.

Party A, Party B and Party C should reach an agreement on $credential_B$ and $credential_C$ before Party C takes over the products from Party A. Party C can verify the validity of $credential_C$ by randomly choosing some $(m_j, Agg(m_j))$ pairs to query the tags. While Party B cannot check the validity of $credential_B$ at that time

as it cannot access the goods/tags. Note that when Party C hands over the products to Party B, Party B will check the products using $credential_B$. If $credential_B$ is not a valid credential, the handover will fail. Party C has an incentive to check the validity of $credential_B$ as well. However, if Party C knows the content of $credentail_B$, it could forged tags/products to fool Party B. We should find a way to enable Party C to check the validity of $credential_B$ without revealing $credential_B$'s content.

We can solve the problem using commitment scheme. We call a pair of $(m_j, Agg(m_j))$ as a credential item. Party A generates $2d$ fresh credential items. Then it sends the commitment of each item to Party C. Party C randomly selects d commitments among them. Party A de-commits the selected d commitments and passes the de-committed credential items $items_1$ to Party C. The other d credential items are denoted as $items_2$ as well. Party C then verifies all credential items in $items_1$ by querying the tags. If all the items are valid, Party C should acknowledge the validity of the committed $items_2$. $credential_B$ contains credential items $items_2$, and Party C keeps the commitments of the credential items in $items_2$. When Party C hands over the goods to Party B, Party B can use any credential item in $credential_B$ to check the goods. In case Party B is dishonest and claims that the products received from Party C are not intact, Party C could verify the validity of the credential item that is used by Party B as it has commitment values of all the credential items in $credential_B$. Note that, we check the validity of a credential by sampling some of its credential items. Party A still can inject some invalid credential items without being detected. To address this issue, Party A should put the commitment of the tag secrets into each credential as well. Thus if Party B and Party C use credential items generated by Party A to query the tags, but have conflicted results, a trusted authority may involve in and ask Party A to de-commit the tag secret commitment. Upon reaching an agreement, all the three parties should sign on the agreement.

5.3 Solution Based on Aggregate Signature Scheme

Another solution is based on aggregate signature scheme proposed in [10]. Party A authorizes the valid party with a credential that contains a value V. Each tag T_i is considered as a signature function with key k_i. Upon receiving a query m, tag T_i replies with the signature $\sigma_i(m)$ on m under its key. Then the reader aggregates the individual signatures $\sigma_i(m)$ for $1 \leq i \leq n$, verifies the aggregate signature using the value V.

5.3.1 Building Blocks of Our Aggregated Signature-based Scheme

Bilinear Map: A bilinear map is a map $e : G_1 \times G_2 \rightarrow G_T$, where: (a) G_1 and G_2 are two (multiplicative) cyclic groups of prime order q; (b) $|G_1| = |G_2| = |G_T|$[1]; (c) g_1 is a generator of G_1 and g_2 is a generator of G_2. The bilinear map $e : G_1 \times G_2 \rightarrow G_T$ satisfies the following properties: (a) Bilinear: for all $x \in G_1, y \in G_2$ and $a, b \in \mathbb{Z}_q$, $e(x^a, y^b) = e(x, y)^{ab}$; (b) Non-degenerate: $e(g_1, g_2) \neq 1$.

Short Signature Scheme: Boneh, Lynh, and Shacham proposed the short signature scheme in [11] using the bilinear map. The system contains two groups G_1 and G_2 with prime order q, a full-domain hash function $H(\cdot) : \{0, 1\}^* \rightarrow G_1$, and a bilinear map $e : G_1 \times G_1 \rightarrow G_2$. g is a generator of G. Each signer has public key $X = g^x$, where $x \in \mathbb{Z}_q$ is the corresponding private key. Signing a message M involves computing the message hash $h = H(M)$ and then the signature $\sigma = h^x$. To verify a signature one computes $h = H(M)$ and checks whether $e(\sigma, g) = e(h, X)$.

Aggregate Signature Scheme: Aggregate signature scheme aggregates n signatures on n distinct messages from n distinct users to one signature. Any one should be able to do the aggregation without knowing the users' keys. Boneh and Gentry proposed a scheme [10] to aggregate BLS signatures. Given n individual signatures,

[1] G_1 and G_2 can be the same group.

one computes the aggregate signature as follows: $\sigma_{1,n} = \prod_{i=1}^{n} \sigma_i$, for $1 \leq i \leq n$, where σ_i corresponds to the user $user_i$'s signature on message M_i. Verification of an aggregate BLS signature $\sigma_{1,n}$ includes computing the product of all message hashes and verifying the following match: $e(\sigma_{1,n}, g) \stackrel{?}{=} \Pi_{i=1}^{n} e(h_i, X_i)$ where X_i is the public key of the signer who generates σ_i on message M_i.

ElGamal encryption scheme: ElGamal encryption system [27] is an public key encryption scheme based on the Diffie-Hellman problem. The scheme firstly chooses a multiplicative cyclic group G of order q with generator g. Each user $user_i$ chooses $x_i \in_R \mathbb{Z}_q$, sets x_i as the private key, then computes his public key $X_i = g^{x_i}$. To encrypt a message m to $user_i$, the sender converts his secret message m into an element m' of G, then chooses $r \in_R \mathbb{Z}_q$, computes $c_1 = g^r$ and $c_2 = m' \cdot X_i^r$, and then sends (c_1, c_2) to $user_i$. To decrypt the ciphertext (c_1, c_2), $user_i$ calculates the value of $c_2 \cdot s^{-1}$ from which she can get the plaintext message.

5.3.2 Basic Aggregate Signature-based Solution

Requirements of the tags: The tags should be able to perform multiplication and addition on a multiplicative cyclic groups G_1 of prime order q. Each tag stores a secret.

Initialization: Suppose there are n tags in a batch, each tag is denoted as T_i, where $1 \leq i \leq n$. Let G_1, G_2 be cyclic groups of the order q. Then Party A chooses a bilinear map: $e : G_1 \times G_1 \rightarrow G_2$. For each tag T_i , Party A chooses a value $k_i \in_R \mathbb{Z}_q$ as the tag's individual secret. k_i is T_i 's individual secret. Then Party A generates the credentials. A credential contains a value V. V is computed to satisfy the following equation:

$$V = g^{\Sigma_{i=1}^{n} k_i} \tag{5.1}$$

where g is a generator of G_1.

Authorization to valid party: Party A keeps the tags' secrets and grants a credential to a valid party as well as the system parameters.

Group checking protocol for authorized party: The details of the protocol are shown below. Figure 5.2 depicts this solution.

1. *Reader* $\rightarrow$ *Tag* T_i: The reader firstly chooses a random number $r_i \in_R \mathbb{Z}_q$, computes $m_i = g^{r_i}$, sends m_i to the tag.

2. *Tag* $T_i \rightarrow$ *Reader*: After receiving m_i, T_i computes $\sigma_i = m_i^{k_i}$, then sends σ_i to the reader. For each tag T_i, the reader records the reply σ_i.

- After getting all the tags's replies σ_i, the reader checks whether $e(\prod_{i=1}^{n} \sigma_i, g) = e(g^{\sum_{i=1}^{n} r_i}, V)$. If the equation holds, the reader confirms that the tags are the original ones; else, the tags are not all original.

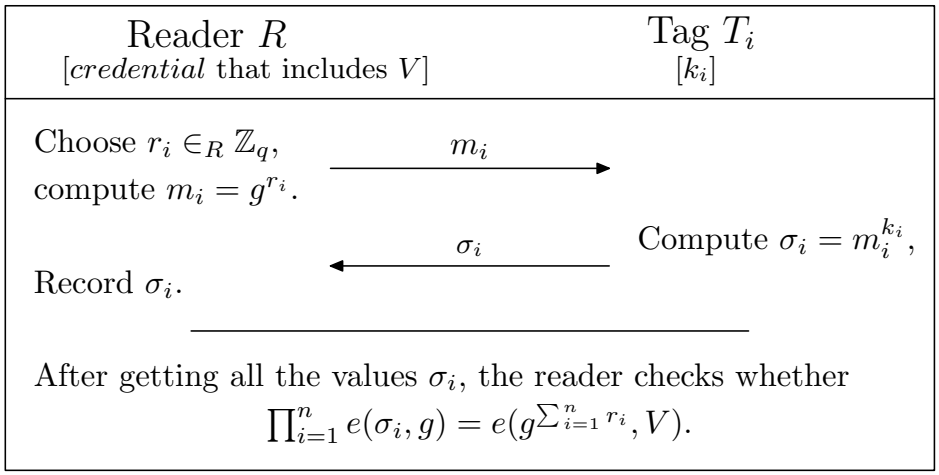

Figure 5.2: Basic Aggregate Signature-Based Solution

5.3.3 Advanced Aggregate Signature-based Solution

The basic aggregate signature-based scheme guarantees that the authorized party can check the tags without the secrets in batch level. However, it does not provide location privacy since a tag sends the same reply to the same challenge in different

sessions. To get the anti-tracing property, we randomize the tag's reply by using the ElGamal encryption scheme.

Requirements of the tags: Additional to the requirements in basic scheme, each tag T_i stores a copy of a public key S, $S = g^s$, where s is the corresponding private key in advance.

Initialization: The same as the basic scheme except that the credential contains another value s.

Authorization to valid party: The same as the basic scheme.

Group checking protocol for authorized party: The details of the advanced aggregate signature based scheme are shown below. Figure 5.3 illustrates the proposal.

1. *Reader* $\rightarrow$ *Tag* T_i: The reader firstly chooses a random number $r_i \in_R \mathbb{Z}_q$, computes $m_i = g^{r_i}$, sends m_i to query T_i.

2. *Tag* T_i $\rightarrow$ *Reader*: After receiving m_i, T_i computes $\sigma_i = m_i^{k_i}$. Then T_i generates a random number $r_2 \in_R \mathbb{Z}_q$, computes $M_1 = \sigma_i \cdot S^{r_2}$, $M_2 = g^{r_2}$, namely T_i encrypts σ using the ElGamal encryption scheme. T_i sends (M_1, M_2) to the reader finally.

3. *Reader*: Receiving (M_1, M_2), the reader decrypts M_1, M_2, gets $v_i = M_1/M_2^s$. The reader records the value of σ_i.

- After getting all the tags's replies σ_i, the reader checks whether $e(\prod_{i=1}^{n} \sigma_i, g) = e(g^{\sum_{i=1}^{n} r_i}, V)$. If the equation holds, the reader confirms that the tags are the original ones, else the tags have been replaced.

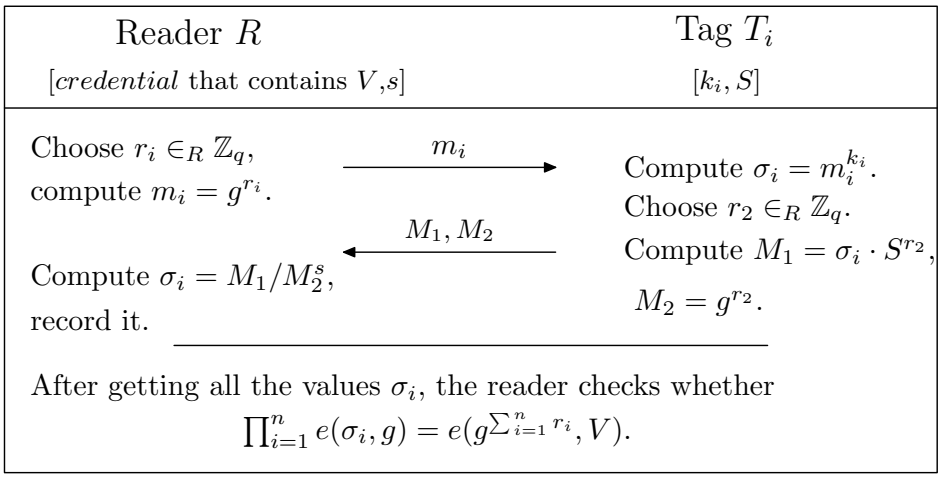

Figure 5.3: Advanced Aggregate Signature-Based Solution

Note that different with the aggregate MAC-Based scheme that each checking consumes a pair of $(m, Agg(m))$, in aggregate signature-based scheme, the value V is reusable. Given V, one cannot forge the tags. Hence, credentials for different parties include the same value V.

5.3.4 Analysis of the Aggregated Signature-based Solution

We first analyze the security and privacy properties of the advanced aggregate signature-based protocol against external adversaries.

- *Tag Information Privacy:* The security of ElGamal encryption scheme guarantees that only the authorized reader with s can decrypt the message (M_1, M_2), then get σ_i. For the authorized reader, with m_i and $\sigma_i(m_i)$, it is computational impossible for one to compute the value of k_i based on the hardness of the discrete logarithm problem. Hence our system guarantees tag information privacy.

- *Tag location Privacy:* Our system provides location privacy to external adversaries. Without knowing the secret s, the external adversaries cannot distinguish the two tags because ElGamal encryption introduces randomization.

- *Resistance of tag impersonation attack:* Based on the secure of the aggregate signature scheme, without the knowledge of the tags' keys, it is computational impossible for an adversary to forge the tags that pass the verification using the credential, even with the authorized parties' public key S.

- *Resistance of replay attack:* The authenticated reader uses fresh message m_i to query each tag. Hence, one cannot reuse the reply (M_1, M_2) that contains m_i's information as the response to another query m_i'.

Then we analyze the security and privacy properties of the aggregate signature-based protocol against internal adversaries.

- *Protect Party C:* The security of the aggregate signature scheme guarantees that without the knowledge of tags' secrets, Party C cannot forge the tags T_i' for $1 \leq i \leq n$ that satisfies $e(\prod_{i'=1}^{n} \sigma_i', g) = e(g^{\sum_{i=1}^{n} r_i}, V)$, where σ_i' corresponds to tag T_i''s signature on message m_i'. Hence if the tags pass the verification using V, no one can claim that Party C has replaced the tags.

- *Restrain Party C:* Since our system achieves the tag information private property and resist tag impersonation attack against the adversaries that do not know the tags' secret, Party C cannot gather any extra information of the tags and replace any of the tags.

5.4 Comparisons

We provide two solutions to implement the group checking for the 3PL supply chains. One is aggregate MAC (AMAC)-based, the other is aggregate signature(AS)-based. As analyzed in Section 5.2 and Section 5.3, both the two proposals meet the requirements listed in Section 2, they achieve the same security and privacy level. In this section, we compare the two schemes' performances in Table 1 and their usability in Table 2. We could use the aggregate signature-based

Table 5.1: Comparisons of the AMAC-Based Scheme and the AS-Based Scheme on Computation Performance

	AMAC-based solution	AS-based solution
Generation of a credential	$n \cdot d$ hash operations	1 point multiplication
Computations required on tag (running the protocol)	2 hash operations	3 point multiplications 1 point addition
Computations required on reader (running the protocol)	1 hash operation	2 point multiplications 1 point subtraction
Computations required on reader (Aggregation and verification)	none	2 paring operations 1 point multiplication

Table 5.2: Comparisons of the AMAC-Based Scheme and the AS-Based Scheme on Usability

	AMAC-based solution	AS-based solution
Computation capability (tag)	hash function, random number generator	operations on elliptic curve, random number generator
Storage requirements (tag)	2 values	2 values
Length of the credential	$O(d)$	$O(1)$
Restrictions on query	only allow to use a same pre-fixed value to query a batch of tags	allow to use arbitrary value to query each tag
Systematical support required	a scheme enables Party C to verify the validity of $credential_B$ without knowing the contents of $credential_B$	none

scheme over Elliptic Curves, more details on implementing Elliptic Curve Cryptography(ECC) on RFID chips can be found in [30]. Hence in aggregate signature-based solutions, operation $\cdot$ denotes point addition , operation $/$ denotes point subtraction, exponential operations denotes point multiplication.

In the following tables, n denotes the number of tags in a batch. d denotes the number of $(m, Agg(m))$ pairs in a credential. We ignore the cheap operations $\oplus$, addition, subtraction and comparison of two values on calculating the computation consumptions on the reader side.

From above comparisons, we can find that the aggregate signature-based scheme is better compared to the aggregate MAC-based scheme. The reader can use arbitrary challenges to query the tags, while in aggregate MAC-based scheme, the reader should use a same pre-fixed value to query the whole batch of tags. The

length of a credential is constant in the aggregate signature-based scheme while in aggregated MAC-based scheme, the length of the credential relates to the number of checking granted to a party. In aggregate signature-based scheme, $credential_B$ and $credential_C$ share the same value V, while in aggregate MAC-based scheme, $credential_B$ and $credential_C$ should not contain same $(m, Agg(m))$ pairs and additional scheme is required to convince Party C the validity of $credential_B$ without knowing the contents of $credential_B$.

Although aggregate signature-based scheme is more elegant, the aggregate MAC-based scheme overall takes more advantage since it requires much cheaper tag and performs more efficiently in running the protocol. Although the additional required systematical support will be counted on the reader side, since the efficiency bottleneck of the system is on the tag side, the aggregate MAC-based solution is more suitable for supply chains application.

5.5 Summary

In this chapter, we analyzed the security and privacy requirements of RFID system for 3PL supply chains; we designed two "group checking" protocols for a third party C to check the existences and originality of tags in batch level without knowing any tag secrets. One protocol is designed based on aggregate MAC and the other is based on aggregate signature. Both protocols achieve the goals of protecting Party C and restraining Party C, as required in 3PL supply chains. Comparing the usability and performance of the two schemes, we conclude that the aggregate MAC-based protocol outperforms the aggregate signature-based protocol.

Chapter 6

Path Authentication in RFID-Enabled Supply Chains

Recently, RFID-enabled path authentication was proposed by Blass, Elkhiyaoui and Molva [7, 8], and extended later to be more practical [64], to tackle the counterfeiting problem in supply chains. RFID-based path authentication enables supply chain managers to verify the exact path that a tag has taken. In this chapter, we refine the existing security and privacy notions for RFID-based path authentication proposed in [7, 8]. We propose a new privacy notion, called path privacy. Path privacy captures the privacy of both tag identity and path information in a single game. Compared to existing two-game based privacy notions, it is more rigorous, powerful, and concise. We also construct two new path authentication schemes. One is for closed supply chains, and another is for dynamic supply chains. Both of the two schemes can be deployed with standard EPC Class 1 Gen 2 tags in the market.

6.1 Formal Framework

In this section, firstly, we provide an RFID-enabled supply chain management system model and an adversary model; then we refine the existing security and privacy notions for RFID-enabled path authentication in supply chains; finally, we show our

new privacy notion, path privacy.

6.1.1 RFID-Enabled Supply Chain Management System

Supply chain is a network of multiple parties, which can be represented by a digraph $G = (V, E)$, where V is a set of vertices, E is a set of edges. Each vertex $v \in V$ represents one *step* in the supply chain. Note that each supply chain party may conduct several steps to process an item. Each directed edge $e \in E$, $e = \overrightarrow{v_i v_j}$, denotes that v_j is a possible next step to step v_i in the supply chain. A *path* is a finite sequence of steps $P = (v_0, \cdots, v_l)$, where $(v_i, v_{i+1}) \in E$, for $i \in \{0, l-1\}$. Every path shares the same source v_0. The last step v_l of a valid path $P_{valid_i} = (v_0, \cdots, v_l)$ represents a *check point*. Every item enters the supply chain from v_0, and goes through a path according to its own procedure. When it arrives at the check point, the manager will verify the item. Note that if a path consists of an empty set of steps (except v_0), we call it empty path, and denote it as "$-$".

An RFID-enabled supply chain system consists of an issuer I, a set of managers $\mathcal{M}$ and a set of normal readers $\mathcal{R}$. The issuer I is located at the source v_0 of the supply chain; a manager from $\mathcal{M}$ is placed at the end of each valid path and normal readers from $\mathcal{R}$ are placed at other places of a supply chain. The issuer I initializes a tag by storing certain information on the tag. While a tag goes through the supply chain, each reader in its path updates the content of the tag. Eventually, the tag arrives at a manager, the manager reads out the content of the tag and checks the validity of the tag. Formally, the system has the following functions:

- Initialize(κ): Given the security parameter κ, the system prepares a supply chain G, an issuer I and a set of l managers $\mathcal{M}$, a set of m readers $\mathcal{R}$ and a set of n tags $\mathcal{T}$, and a set of ν valid path $\mathcal{P}_{valid}$. We denote the content stored on any tag T_i as state S_{T_i}.

- Read(T_i): a function that returns back the current internal state S_{T_i} of T_i.

- Write(T_i): a function that writes a new state S'_{T_i} to T_i. Here we assume that the readers in each step are honest, that is, they update a tag only if the tag is authenticated.

- PathCheck($S^j_{T_i}$): a function that verifies whether tag T_i has gone through a valid path P_{valid}. If it is the case, it returns the valid path P_{valid}, else it returns $\emptyset$.

6.1.2 Adversary Model

We use the following notations. If $A(\cdot, \cdot, \cdots)$ is a randomized algorithm, then $y \leftarrow A(x_1, x_2, \cdots; \rho)$ means that y is assigned with the unique output of algorithm A on inputs x_1, x_2, $\cdots$ and coins ρ, while $y \leftarrow A(x_1, x_2, \cdots)$ is a shorthand for first picking ρ at random and then setting $y \leftarrow A(x_1, x_2, \cdots)$. $y \leftarrow A^{O_1,\cdots,O_n}(x_1, x_2, ...)$ denotes that y is assigned with the output of algorithm A which takes $x_1, x_2, ...$ as inputs and has oracle accesses to $O_1, ..., O_n$. If S is a set, then $s \in_R S$ indicates that s is chosen uniformly at random from set S. Let Pr[E] denote the probability that an event E occurs. Let $\mathcal{N}$ denote the set of all integers. Let $\mathcal{R}$ denote the set of all real numbers. A function $f : \mathcal{N} \rightarrow \mathcal{R}$ is said to be *negligible* if for every $c > 0$ there exits a number $n_0 \in N$ such that $f(n) < \frac{1}{n^c}$ holds for all $n > n_0$.

An adversary $\mathcal{A}$, against RFID path authentication, is given accesses to four oracles $\mathcal{O} = \{O_1, O_2, O_3, O_4\}$. O_1, O_2, O_3 denote Read, Write, PathCheck functions, respectively. O_4 denotes a function Move($T_i, k, \mathcal{K}, b$), where $k \in \mathcal{N}$, $\mathcal{K} \in \{P, G\}$, $b \in \{0, 1\}$. Move ($T_i, k, \mathcal{K}, b$) is defined as follows:

- If $\mathcal{K} = G$, no matter whether $b = 0$ or $b = 1$, starting from the current step of T_i with internal state $S^j_{T_i}$, move the tag T_i forward $k \geq 1$ steps arbitrarily in the supply chain system G.

- If $\mathcal{K} = P$, works as follows: If $b = 1$, from the current step of T_i with internal state $S^j_{T_i}$, move the tag T_i forward $k \geq 1$ steps through the designated path $\mathcal{P}$

(the length of P is at least k steps). If $b = 0$, move tag T_i forward $k \geq 1$ steps according to any path that does not have a common step with P. The reader in each step updates the tag's state. Finally, $\mathsf{Move}(T_i, k, P, b)$ returns back the state transcript $\{S_{T_i}^{j+1}, \cdots, S_{T_i}^{j+k}\}$ of T_i from step $j+1$ to $j+k$.

It is critical to precisely model various kinds of tag movement. In [8, 7], the concept of path is not explicitly defined, and the operations on tag movement are specified through step-level oracles; thus, it is difficult to describe the tag movement at path level. We firstly propose an oracle, namely O_4, to model the tag movement at path level. Using O_4, any tag movement can be precisely represented by adjusting the parameters of $\mathbf{Move}$ function. The introducing of O_4 facilitates defining clear security and privacy notions.

The four oracles capture the adversary's ability to read from a tag, write into a tag, check the validity of a tag, and follow a tag through a designated path P (for the case of $\mathcal{K} = P$) or simply update the state of the tag by forwarding it arbitrarily in the system G (for the case of $\mathcal{K} = G$). We denote by $\mathcal{A}^{\mathcal{O}}(para)$ a probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$ that, on input of some system public parameters $para$, runs a supply chain system via the four oracles in $\mathcal{O}$. An adversary is a (t, n_1, n_2, n_3, n_4)-adversary if it works in time t and makes oracle queries to O_μ without exceeding n_μ times, where $1 \leq \mu \leq 4$.

6.1.3 Existing Security and Privacy Notions

Security notion The security goal of our system is to prevent an adversary from inserting counterfeited goods to the supply chain. As the manager checks the authenticity of a tag merely based on the state stored on a tag, the system should prevent an adversary from forging a tag's internal state with a valid path that has not been actually taken by the tag in the supply chain. Since standard EPC C1 G2 tags have no computation capability, no reader authentication is performed. If a tag's state has been changed by an adversary, even if it has gone through a valid path, it

Experiment $\mathbf{Exp}_{\mathcal{A}}^{Security}[\kappa]$

1. run $\mathsf{Setup}(\kappa)$ to setup $I, \mathcal{R}, \mathcal{T}, \mathcal{M}$.
2. $\{st\} \leftarrow \mathcal{A}_1^{\mathcal{O}}(para)$. // the learning phase
3. $T \leftarrow \mathcal{A}_2(st)$. //the challenge phase
4. $s_T^j \leftarrow \mathsf{Read}(T)$.
5. output 1, if $\mathcal{P}_{valid} \longleftarrow \mathsf{PathCheck}(S_T^j)$,
   and there is a step $v_z \in P_{valid}$ which T has not gone through in its z-th step;
   output 0, otherwise.

Figure 6.1: Security Experiment

is not considered as a valid tag by a manager.

An RFID path authentication scheme is considered secure if it is infeasible for any probabilistic polynomial-time adversary $\mathcal{A}$ to create a state $S_{T_i}^l$ for a tag T_i such that given $S_{T_i}^l$, a manager M outputs a valid path $P_{valid} = \{v_0, \cdots, v_l\}$ which T_i has not gone through. It is formalized by an experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathbf{Security}}[\kappa]$ shown in Figure 1. The adversary $\mathcal{A}$ consists of two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ which run in two phases, learning phase and challenge phase. Firstly, given parameter κ, the experiment setups the system through $\mathsf{Setup}(\kappa)$, and passes the public system parameters $para$ to $\mathcal{A}_1$. In the learning phase, $\mathcal{A}_1$ is allowed to collect information by querying the four oracles without exceeding n_1, n_2, n_3, n_4 times, respectively. Then it generates a transcript st which contains the information about the system it gathered during the learning phase. In the challenge phase, $\mathcal{A}_2$ creates a tag T with state s_T^j using st. The tag T may have a new ID or an existing ID in the system. Then the game checks the validity of T_i through $\mathsf{Check}(s_T^j)$. $\mathbf{Exp}_{\mathcal{A}}^{\mathbf{Security}}[\kappa]$ outputs 1 if both of the following two conditions hold: $\mathsf{Check}(s_T^j)$ returns a valid path P_{valid}; and there exists $z \in \{1, \ldots, l\}$ such that the tag has not passed v_z in its z-th step, where l denotes the length of the path and v_z denotes the z-th step in P_{valid}. $\mathbf{Exp}_{\mathcal{A}}^{\mathbf{Security}}[\kappa]$ outputs 0, otherwise.

Definition 6.1.1. *The advantage of $\mathcal{A}$, denoted $Adv_{\mathcal{A}}^{Security}(\kappa)$, in the security ex-*

periment is

$$\left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathbf{Security}}[\kappa] = 1]\right|$$

Definition 6.1.2. *We say an RFID path authentication scheme is* $(t, n_1, n_2, n_3, n_4, \epsilon)$*-secure, if for any* t*-time adversary* $\mathcal{A}$ *who makes at most* n_1, n_2, n_3, n_4 *queries to* O_1, O_2, O_3, O_4 *respectively,* $Adv_{\mathcal{A}}^{Security}(k) < \epsilon$ *holds. The probability is taken over coins of* $\mathcal{A}$ *and the oracles.*

Privacy notions For an RFID-enabled supply chain system, Blass, Elkhiyaoui and Molva [7] considered two privacy notions: tag unlinkability and step unlinkability. Tag unlinkability corresponds to the privacy of a tag's identity. Step unlinkability corresponds to the privacy of a tag's path. Note that in the older version of TRACKER [8], there is another path privacy notion, namely path unlinkability, which is proven to be weaker than step unlinkability in [7].

Tag Unlinkability Briefly, tag unlinkability requires that no efficient adversary can link the state information stored in a tag to the tag's identity. In [8], the tag unlinkability is defined through a formal experiment. The experiment contains two phases: the learning phase and the challenge phase. An adversary $\mathcal{A}$ is provided with two tags T_0 and T_1. In the learning phase, the adversary can access the system and gather information without exceeding the constraints set by the game. In the challenge phase, the game updates the tags by moving them one more step further in the supply chain. The experiment then flips a coin $\delta \in_R \{0, 1\}$, and provides the updated state of T_δ to the adversary. The adversary guesses the value of δ. The adversary wins the game if it can successfully guess δ with probability $1/2$ plus a non-negligible quantity.

We slightly modify the experiment to $Exp_{\mathcal{A}}^{Tag-Unlinkability}[\kappa]$. In the learning phase, the adversary is allowed to access the oracles O_1, O_2, O_3, O_4 without exceeding n_1, n_2, n_3, n_4 times, respectively. Then, the adversary outputs two tags T_0 and T_1 together with a transcript st, where st is the information it has gathered. In the challenge phase, the experiment tosses a coin $\delta \in_R \{0, 1\}$. The experimen-

t moves the tag T_δ one step forward arbitrarily in the system G, and provides the updated state S_δ of tag T_δ to the adversary. With S_δ and the transcript st, the adversary guesses the value of δ, then outputs the guessed value δ'. If $\delta = \delta'$, the experiment outputs 1; else, the experiment outputs 0. The adversary wins the game if the experiment outputs 1 with probability $1/2$ plus a non-negligible quantity.

A key difference between the original tag unlinkability notion [7] and our refined one is that, in the original notion, the challenge tags T_0 and T_1 are selected by the experiment, while in our notion, the challenge tags T_0 and T_1 are selected by the adversary; therefore, the adversary in our notion is stronger than the adversary in [7]. We depict $Exp_{\mathcal{A}}^{Tag-Unlinkability}[\kappa]$ in Figure 2.

Experiment $\mathbf{Exp}_{\mathcal{A}}^{Tag-Unlinkability}[\kappa]$

1. run Setup(κ) to setup $I, \mathcal{R}, \mathcal{T}, \mathcal{M}$.

   Denote by $para$ the public system parameter.

2. $\{T_0, T_1, st\} \leftarrow \mathcal{A}_1^{\mathcal{O}}(para)$.

3. $\delta \longleftarrow_R \{0, 1\}$.

4. $S_\delta \leftarrow$Move$(T_\delta, 1, G, 1)$, i.e., move T_δ one step arbitrarily forward in the system G.

   Denote by $\mathcal{S}_\delta$ the updated state of T_δ.

5. $\delta' \leftarrow \mathcal{A}_2^{\mathcal{O}}(S_\delta, st)$.

6. output 1 if $\delta' = \delta$, 0 otherwise.

Figure 6.2: Tag Unlinkability Experiment

Definition 6.1.3. *The advantage of $\mathcal{A}$, denoted $Adv_{\mathcal{A}}^{Tag-Unlinkability}(\kappa)$, in the tag unlinkability experiment is* $\left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathbf{Tag-Unlinkability}}[\kappa] = 1] - \frac{1}{2}\right|$

Definition 6.1.4. *An RFID path authentication scheme is $(t, n_1, n_2, n_3, n_4, \epsilon)$-tag-unlinkable, if for any t-time adversary $\mathcal{A}$ who makes at most n_1, n_2, n_3, n_4 queries to O_1, O_2, O_3, O_4, respectively, we have $Adv_{\mathcal{A}}^{Tag-Unlinkability}(\kappa) < \epsilon$. The probability is taken over the choice of δ, coins of $\mathcal{A}$ and the oracles.*

Step Unlinkability Step unlinkability requires that no efficient adversary is feasible

to tell whether the two paths of any two different tags share a common step or not. In [7], the step unlinkability game is defined as follows. Firstly, the experiment randomly chooses a tag T for the adversary. In the learning phase, the adversary arbitrarily queries the oracle without exceeding the constraints. The adversary may gather information from the system. It may follow T, so that it knows the path of the targeted tag. In the challenge phase, the experiment provides the adversary with another tag T_c, the adversary lets T_c move forward along its path for several steps and then reads the state of T_c. Finally, the adversary is asked to guess whether T and T_c have a step in common besides v_0. The adversary breaks the path privacy if the probability of correct guessing is non-negligibly more than $\frac{1}{2}$.

The above experiment defined in [8] is based on the assumption that every tag passes through every step with the same probability. However, given a tag, in case that the probabilities of the tag to pass by different steps are not even, then, an adversary can trivially win the game. We give an example to illustrate the situation. Suppose there are four paths in the system, P_a, P_b, P_c and P_d and every tag will go through the fours paths with equal probability. P_a, P_b, P_c shares a common step v besides v_0, while P_d has no common step with the other three paths besides v_0. In case that the adversary learns that tag T has gone through path P_a, for any T_c the probability that it has a common step v with T is 75%. Thus the adversary will win the game with non-trivial advantage.

We modify the step unlinkability experiment to make it more rigorous. The new step unlinkability experiment $Exp_{\mathcal{A}}^{Step-Unlinkability}[\kappa]$ is shown in Figure 3. The experiment starts by setting the system $I, \mathcal{R}, \mathcal{T}, \mathcal{M}$ through $\mathsf{Setup}(\kappa)$. An adversary $\mathcal{A}$ runs two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively in the two phases. In the learning phase, $\mathcal{A}_1$ queries the oracle set $\mathcal{O}$ and outputs a tag T and transcript st. In the challenge phase, the experiment creates a new tag T_c, and then tosses a coin $\delta \in_R \{0, 1\}$. The experiment sets a path P as follows: if $\delta = 0$, the path P does not have any common step with T's path; else the path P have certain common steps with T's path. After getting the path P, the experiment moves T_c along path P in k steps. $\mathcal{A}_2$

reads the state S_{T_c} of T_c, guesses the value of δ, and outputs the guessed value δ'. Note that S_{T_c} contains the states updated by the readers in path P. If the probability of $\delta' = \delta$ is non-negligibly more than $\frac{1}{2}$, the adversary wins the game.

Experiment $\mathbf{Exp}_{\mathcal{A}}^{Step-Privacy}[\kappa]$

1. run Setup(κ) to setup $T, \mathcal{R}, \mathcal{T}, \mathcal{M}$. Denote by $para$ the public system parameter.
2. $\{T, k, st\} \leftarrow \mathcal{A}_1^{\mathcal{O}}(para)$.
3. create a new tag T_c.
4. randomly selects a bit $\delta \in \{0, 1\}$.
5. if $\delta = 0$, selects a path P that dose not have any common step with T' path; else, select a path P that has one or more common steps with T's path. The length of the path is at least k.
6. $S_{T_c} \leftarrow$Move$(T_c, k, P, 1)$.
7. $\delta' \leftarrow \mathcal{A}_2(S_{T_c}, st)$.
8. output 1 if $\delta' = \delta$, 0 otherwise.

Figure 6.3: Step Unlinkability Experiment

Definition 6.1.5. *The advantage of $\mathcal{A}$, denoted $Adv_{\mathcal{A}}^{Step-Unlinkability}(k)$, in the step unlinkability experiment is* $\left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathbf{Step-Unlinkability}}[\kappa] = 1] - \frac{1}{2}\right|$

Definition 6.1.6. *An RFID path authentication scheme is $(t, n_1, n_2, n_3, n_4, \epsilon)$-step unlinkable, if for any t-time adversary $\mathcal{A}$ who makes at most n_1, n_2, n_3, n_4 queries to O_1, O_2, O_3, O_4, respectively, we have $Adv_{\mathcal{A}}^{Step-Unlinkability}(k) < \epsilon$. The probability is taken over the choice of δ, coins of $\mathcal{A}$ and oracles.*

6.1.4 A New RFID Privacy Notion for Path Authentication

In this section, we propose a new privacy notion, named path privacy, for path authentication. This notion captures the privacy of tag identity and path information in a single game. We show that path privacy implies tag unlinkability and step unlinkability.

Path privacy

In [7], two privacy notions, tag-unlinkability and step-unlinkability, should be used together to analyze the privacy of a path authentication scheme. These two notions are formulated separately (via four algorithms). We present a single game-based privacy notion, path-privacy, which implies tag unlinkability and step unlinkability.

The experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathbf{Path-Privacy}}[\kappa]$ of path privacy is shown in Figure 4 and formalized as follows. The experiment consists of two phases: the learning phase and the challenge phase. An adversary $\mathcal{A}$ runs two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively in the two phases. The experiment sets up the system $I, \mathcal{R}, \mathcal{T}, \mathcal{M}$ through $\mathsf{Setup}(\kappa)$. In the learning phase, $\mathcal{A}_1$ queries the four oracles without exceeding n_1, n_2, n_3, n_4 times, respectively. $\mathcal{A}_1$ outputs two tags T_0, T_1, a path P that has at least k steps left for both tags, and state information st. In the challenge phase, the experiment firstly flips a coin δ. If $\delta = 1$, the experiment moves T_1 k steps along the path P, and T_1 is updated by k readers in the path. Let the state of T_1 be denoted as S_1. If $\delta = 0$, the experiment moves T_0 k steps without going through the path P (T_0 is updated by k readers that are not in the path). Let the state of T_0 be denoted as S_0. The Move operations are performed by the game challenger, and the adversary has no access to the readers and the tag during the Move operations. In the challenge phase, the experiment provides $\mathcal{A}_2$ with S_δ and st. $\mathcal{A}_2$ guesses the value of δ as δ'. If $\delta' = \delta$, the experiment outputs 1; else the experiment outputs 0. If the experiment outputs 1 with probability non-negligibly more than $\frac{1}{2}$, the adversary wins the game.

Definition 6.1.7. *The advantage of $\mathcal{A}$, denoted $Adv_{\mathcal{A}}^{Path-Privacy}(k)$, in the path privacy experiment is* $\left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathbf{Path-Privacy}}[\kappa] = 1] - \frac{1}{2}\right|$

Definition 6.1.8. *A RFID path authentication scheme is $(t, n_1, n_2, n_3, n_4, \epsilon)$-private, if for any t-time adversary $\mathcal{A}$ who makes at most n_1, n_2, n_3, n_4 queries to O_1, O_2, O_3, O_4, respectively, we have $Adv_{\mathcal{A}}^{Privacy}(k) < \epsilon$. The probability is taken over the choice of δ, coins of $\mathcal{A}$ and oracles.*

Experiment $\mathbf{Exp}_{\mathcal{A}}^{Path-Privacy}[\kappa]$

1. run $\mathsf{Setup}(\kappa)$ to setup $I, \mathcal{R}, \mathcal{T}, \mathcal{M}$. Denote by $para$ the public system parameter.
2. $\{T_0, T_1, P, k, st\} \leftarrow \mathcal{A}_1^{\mathcal{O}}(para)$, where P is a path of length at least k, st is state information.
3. $\delta \leftarrow \{0, 1\}$.
4. $S_\delta \leftarrow$Move(T_δ, k, P, δ). Denote by $\mathcal{S}_\delta$ the state of $\mathcal{T}_\delta$.
5. $\delta' \leftarrow \mathcal{A}_2^{\mathcal{O}}(S_\delta, st)$.
6. output 1 if $\delta' = \delta$, 0 otherwise.

Figure 6.4: Path Privacy Experiment

Relations among Privacy Notions

Now, we show that path-privacy is stronger than tag unlinkability and step unlinkability.

Theorem 1. *Path-privacy implies tag unlinkability.*

Proof. Path privacy implies that S_0 and S_1 in the path privacy experiment are computationally indistinguishable, even if the adversary $\mathcal{A}$ has full control over the supply chain system via the four oracle access *except that the random bit δ is blinded to $\mathcal{A}$*. Intuitively, tag unlinkability is implied by path privacy, as the ability of linking tag's state to tag's identity can be *directly* used to break path privacy.

In more details, we show that it is possible to construct an adversary $\mathcal{B}$ that $(t, n_1, n_2, n_3, n_4, \epsilon)$-breaks path privacy using $\mathcal{A}$ as a subroutine, where $\mathcal{A}$ is an adversary which can $(t, n_1, n_2, n_3, n_4, \epsilon)$-break tag unlinkability. Adversary $\mathcal{B}$ plays the path privacy game using adversary $\mathcal{A}$ as a subroutine; it is $\mathcal{A}$ who conducts the attacks to the system, while A aims to win the tag-unlinkability game. Firstly, $\mathbf{Exp}_{\mathcal{B}}^{\mathbf{Path-Privacy}}[\kappa]$ sets up the system $I, \mathcal{R}, \mathcal{T}, \mathcal{M}$ and publishes the public system parameter $para$. Then $\mathcal{B}$ passes $para$ to $\mathcal{A}$. $\mathcal{A}$ plays the tag-unlinkability game. In the learning phase, when $\mathcal{A}_1$ queries the oracles $\mathcal{O}$, the queries are trans-

ferred to $\mathcal{B}_1$, and $\mathcal{B}_1$ queries the oracles $\mathcal{O}$ for $\mathcal{A}_1$ in the path-privacy experiment. Then $\mathcal{A}_1$ outputs $\{T_0, T_1, st\}$. Upon receiving $\mathcal{A}_1$'s output, $\mathcal{B}_1$ chooses a path P and submits $\{T_0, T_1, P, 1, st\}$ to the path-privacy experiment. The experiment $\mathbf{Exp}_{\mathcal{B}}^{\mathbf{Path-Privacy}}[\kappa]$ chooses $\delta \in_R \{0, 1\}$, and returns $S_\delta \leftarrow \text{Move}(\mathcal{T}_\delta, 1, P, 1)$ to $\mathcal{B}_2$. $\mathcal{B}_2$ transfers S_δ to $\mathcal{A}_2$. When $\mathcal{A}_2$ stops, $\mathcal{B}_2$ outputs whatever output by $\mathcal{A}_2$. It is clear that if $\mathcal{A}$ wins the tag unlinkability game, then $\mathcal{B}$ wins the path privacy game. We have:

$$\Pr[\mathbf{Exp}_{\mathcal{B}}^{\mathbf{Path-Privacy}}[\kappa] = \mathbf{1}] = \Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathbf{Tag-Unlinkability}}[\kappa] = \mathbf{1}] \tag{6.1}$$

If $\mathcal{A}$ $(t, n_1, n_2, n_3, n_4, \epsilon)$-breaks tag-unlinkability, then $\mathcal{B}$ also $(t, n_1, n_2, n_3, n_4, \epsilon)$-breaks path privacy. □

Theorem 2. *Path privacy implies step unlinkability.*

Proof. Assuming that a system is not step-unlinkable, there exists an adversary $\mathcal{A}$ which can $(t, n_1, n_2, n_3, n_4, \epsilon)$-break its step unlinkability. We can construct an adversary $\mathcal{B}$ that breaks the path privacy using $\mathcal{A}$ as a subroutine.

$\mathbf{Exp}_{\mathcal{B}}^{\mathbf{Path-Privacy}}[\kappa]$ sets up the system $I, \mathcal{R}, \mathcal{T}, \mathcal{M}$ and publishes the public system parameter $para$. $\mathcal{B}$ passes $para$ to $\mathcal{A}$. If $\mathcal{A}$ can break the step unlinkability in $\mathbf{Exp}_{\mathcal{A}}^{\mathbf{Step-Unlinkability}}[\kappa]$. Then $\mathcal{B}$ can use $\mathcal{A}$ as a subroutine to break path-privacy. In the learning phase, $\mathcal{A}_1$ gathers the information of the system. In this process, $\mathcal{A}_1$ cannot query the oracles directly; instead, it submits the queries to $\mathcal{B}_1$ and then $\mathcal{B}_1$ queries the oracles $\mathcal{O}$ for $\mathcal{A}_1$. Then $\mathcal{A}_1$ outputs $\{T, st\}$. As $\mathcal{A}_1$ fully controls the system during the learning phase, then $\mathcal{A}_1$ knows the path of T. We denote the path by P, which is contained in st. Then $\mathcal{A}_1$ passes $\{T, k, st\}$ to $\mathcal{B}_1$. $\mathcal{B}_1$ creates two new tags T_0 and T_1 and outputs $\{T_0, T_1, P, k, st\}$. $\mathbf{Exp}_{\mathcal{B}}^{\mathbf{Path-Privacy}}[\kappa]$ tosses a coin δ. If $\delta = 0$, then the experiment moves T_0 without going through path P in k step, and the state of T_0 is denoted as S_0; else, the experiment moves T_1 through path P

in k step, and the state of T_1 is denoted as S_1. The experiment returns S_δ to $\mathcal{B}_2$. $\mathcal{B}_2$ transfers S_δ to $\mathcal{A}_2$, and outputs whatever output by $\mathcal{A}_2$.

In the above path-privacy game, $\mathcal{B}_2$ is provided with the state S_δ. If $\delta = 0$, then the tag with state S_0 does not have any common step with T. If $\delta = 1$, then the tag with state S_1 has at least one common step with T. Given S_δ, $\mathcal{A}_2$ guesses whether the tag has a common step with T or not. $\mathcal{B}_2$ can directly use the result of $\mathcal{A}_2$. It is clear that:

$$\Pr[\mathbf{Exp}_{\mathcal{B}}^{\mathbf{Path-Privacy}}[\kappa] = \mathbf{1}] = \Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathbf{Step-Unlinkability}}[\kappa] = \mathbf{1}] \tag{6.2}$$

Hence, if $\mathcal{A}$ $(t, n_1, n_2, n_3, n_4, \epsilon)$-breaks the step-unlinkability, then $\mathcal{B}$ also $(t, n_1, n_2, n_3, n_4, \epsilon)$-breaks the path privacy. □

6.2 Path Authentication Protocol for Closed Supply Chain Systems

In this section, we propose a new RFID-based path authentication scheme. Our path authentication scheme is suitable for a supply chain that the paths of products are pre-determined. We use pseudorandom functions and elliptic curve ElGamal encryption scheme as building blocks.

6.2.1 Building Blocks

Pseudorandom function Given a security parameter κ, let $m(\cdot)$ and $l(\cdot)$ be two positive polynomials in κ. We say that

$$\{F_k : \{0,1\}^{m(\kappa)} \longrightarrow \{0,1\}^{l(\kappa)}\}_{k \in_R \{0,1\}^\kappa} \tag{6.3}$$

is a PRF ensemble if the following two conditions hold:

1. Efficient evaluation: There exists a polynomial-time algorithm that on input k and $x \in \{0,1\}^{m(\kappa)}$ returns $F_k(x)$.

2. Pseudorandomness: A PPT oracle machine $\mathcal{A}(t,\varepsilon)$-breaks the PRF ensemble, if

$$|Pr[\mathcal{A}^{F_\kappa}(\kappa)=1] - Pr[\mathcal{A}^{H_\kappa}(\kappa)=1]| \geq \varepsilon \tag{6.4}$$

where F_κ is a random variable uniformly distributed over the multi-set $F_k, k \in_R \{0,1\}^\kappa$, H_κ is uniformly distributed among all functions mapping $m(\kappa)$-bit-long strings to $l(\kappa)$-bit-long strings, and the running time of $\mathcal{A}$ is at most t (here each oracle query accounts for one unit operation).

The PRF ensemble is pseudorandom, if for all sufficiently large κ, there exists no algorithm A that can (t,ϵ)-break the PRF ensemble, for any t that is polynomial in κ and any ϵ that is nonnegligible in κ [67].

Elliptic Curve ElGamal Cryptosystem We have introcuded ElGamal cryptosystem over finite fields in Section 5.3. Elliptic curve ElGamal cryptosystem is over elliptic curves. ElGamal system supports re-encryption operation denoted as ReE. Given a ciphertext $c = (U,V)$ under a public key $pk = (P, Y = sk \cdot P)$, and the public key pk, ReE re-randomizes the ciphertext c to c', where $c' = (U',V') = (U + r \cdot P, V + r \cdot Y)$, for $r \in_R \mathbb{F}_q$. ElGamal system preserves the semantic security property under re-encryption [29]. Let $O_{re-encrypt}$ be an oracle that, provided with two ciphertexts c_0, c_1, randomly chooses $b \in \{0,1\}$, re-encrypts c_b using ElGamal and public key pk, and returns the resulting ciphertext c_b. The semantic security of ElGamal under re-encryption implies that guessing the value of b is as difficult for $\mathcal{A}$ as the decisional Diffie-Hellman (DDH) problem [29].

6.2.2 Protocol

Assume that an RFID-enabled supply chain path authentication system consists of a set of n tags, an issuer I, a set of l managers $\mathcal{M}$, and a set of m normal readers

$\mathcal{R}$. Our protocol has three steps: initialization, updating and verification. In the initialization step, the issuer and the managers setup the system together and initialize the tags. When the tags enter the supply chain, the corresponding reader updates the tags on each step. Finally, when a tag reaches a manager in $\mathcal{M}$, the manager reads out the content of the tag and checks the validity of the tag. Each tag stores an encrypted ID and an encrypted credential generated by the readers in its path.

Initialization: The managers $\mathcal{M}$ generate a couple of public key and secret key (pk, sk) for ElGamal encryption and send pk to the issuer and the readers. The underlying elliptic curve of the ElGamal system is denoted as $\mathcal{E}$. The issuer $\mathcal{I}$ selects a secret-key $k_0 \in \{0,1\}^\kappa$, where κ is the system parameter. $\mathcal{I}$ sets for each reader R_j a secret key k_j, where $1 \leq j \leq m$. $\mathcal{I}$ distributes k_j to R_j. The issuer selects a pseudorandom function ***PRF***, and sends ***PRF*** to all the normal readers.

Each tag T_i has an unique identity ID_i, where $ID_i \in \mathcal{E}$. For each T_i, the issuer $\mathcal{I}$ sets its initial state to be $\{c_i = E(ID_i), t_i = PRF_{k_0}(ID_i)\}$. We denote the path which T_i will go through as P_i. Suppose $P_i = (R_{i_0}, R_{i_1}, R_{i_2}, \cdots, R_{i_l})$, for any $0 \leq j \leq l$, where i_j denotes the reader ID in the position j of path P_i. Then for T_i, the issuer $\mathcal{I}$ computes $v_i = PRF_{k_{i_l}}(PRF_{k_{i_{l-1}}}(\cdots(PRF_{k_0}(ID_i))))$, and stores a copy of (ID_i, v_i) on the databases of the managers $\mathcal{M}$.

Interaction between reader and tag: When tag T_i reaches R_j, reader R_j reads out T_i's current state $S_{T_i} = \{c_i, t_i\}$. R_j computes the new state $\{c_i', t_i'\}$, where c_i' is re-randomization of c_i under the public key pk and $t_i' = PRF_{k_j}(t_i)$, and then writes $\{c_i', t_i'\}$ to the tag.

Check the validity of tag: Only the managers $\mathcal{M}$ can check the validity of tags. Upon the arrival of a tag at a check point, with state $\{c_i, t_i\}$, M decrypts c_i to get ID_i, and searches its database; if and only if it can find a tuple (ID_i, v_i) that satisfies $t_i = v_i$, then T_i is considered as a valid tag.

6.2.3 Security and Privacy Analysis

The security and privacy of the proposed protocol are based on the pseudorandomness of PRF and the semantic security of Elgamal Encryption scheme under re-encryption. In the following, we provide a formal security and privacy analysis.

Suppose PRF is a pseudorandom function that maps $m(\kappa)$-bit-long strings to $l(\kappa)$-bit-long strings. We call the function $CPRF(m) = PRF_{k_l}(PRF_{k_{l-1}}(\cdots(PRF_{k_0}(m)))$ as "cascaded" pseudorandom function, where $k_0, ..., k_l$ are randomly chosen keys for the pseudorandom function PRF. If for all k_i, $0 \leq i \leq l$, PRF_{k_i} is a pseudorandom function, $CPRF(m)$ is a pseudorandom function (formal proof please refer to [8]).

[Lemma] Producing a new valid pair $\{c_i, t_i\}$ contradicts with the pseudorandomness property of $CPRF$. Here a new pair of $\{c_i, t_i\}$ means that c_i is a ciphertext of a new ID_i under the public key of the system, or c_i is a ciphertext of an existing ID_i in the system while t_i is a new value that has not appeared in the system.

Proof (sketch). The security of our system is based on the pseudorandomness of $CPRF(m)$. Suppose there is an oracle $O_{CPRF}^{distinguish}$, given a message m, the oracle randomly returns the value of $CPRF(m)$ or $H(m)$, denoted as m', where $H()$ is an arbitrarily selected function among all functions mapping $m(\kappa)$-bit-long strings to $l(\kappa)$-bit-long strings. After getting m', the adversary outputs 1 if it guesses $m' = CPRF(m)$, else he outputs 0. $Pr[\mathcal{A}^{CPRF}(\kappa) = 1]$ denotes the probability that the adversary outputs 1 when the oracle $O_{CPRF}^{distinguish}$ returns value $CPRF(m)$. $Pr[\mathcal{A}^{H}(\kappa) = 1]$ denotes the probability that the adversary outputs 1 when the oracle $O_{CPRF}^{distinguish}$ returns value $H(m)$. Since $CPRF$ is a pseudorandom function, given $\mathcal{A}$ with limited access to the function $CPRF$, we have $|\Pr[\mathcal{A}^{CPRF}(\kappa) = 1] - \Pr[\mathcal{A}^{H}(\kappa) = 1]| \geq \varepsilon$, where ε is negligible. We will show that if an adversary $\mathcal{A}'$ can successfully forge a new pair $\{c_i, t_i\}$, then using $\mathcal{A}'$ as a subroutine, there exists an adversary $\mathcal{A}$ that breaks $CPRF(m)$'s pseudorandomness, namely the value of $|Pr[\mathcal{A}^{CPRF}(\kappa) = 1] - Pr[\mathcal{A}^{H}(\kappa) = 1]|$ will be non-negligible.

$\mathcal{A}$ sets up a supply chain system with public key pk, private key sk for Elgamal encryption system, and a valid path in which the readers have the keys $k_0, \cdots, k_l$, respectively, where l is the length of the path. $\mathcal{A}$ does not know the keys $k_0, \cdots, k_l$, while it is provided with $PRF_{k_0}, \cdots, PRF_{k_i}$ by the oracle $O_{CPRF}^{distinguish}$. $\mathcal{A}$ transfers the public system parameters to $\mathcal{A}'$ which runs two algorithms $\mathcal{A}'_1$ and $\mathcal{A}'_2$ in $\mathbf{Exp}_{\mathcal{A}'}^{\mathbf{Path-Privacy}}[\kappa]$, . In the learning phase, $\mathcal{A}'_1$ accesses the supply chain system without exceeding the constraints defined in $\mathbf{Exp}_{\mathcal{A}'}^{\mathbf{Path-Privacy}}[\kappa]$. In the challenge phase, $\mathcal{A}'_2$ outputs a new pair $\{c_i, t_i\}$. $\mathcal{A}$ decrypts c_i to get ID. Then $\mathcal{A}$ queries $O_{CPRF}^{distinguish}$ with ID. $O_{CPRF}^{distinguish}$ returns a message mes_{ID}. In case $\{c_i, t_i\}$ is valid, then by checking whether $mes_{ID} = t_i$, $\mathcal{A}$ knows whether $O_{CPRF}^{distinguish}$ has chosen the function $CPRF$ or a random function $H()$. As a result, if $\mathcal{A}'$ $(t, n_1, n_2, n_3, n_4, \epsilon)$-breaks the the security of path authentication, then $\mathcal{A}$ (t, ϵ)-breaks the pseudorandomness of function $CPRF$.

Theorem 3. *If PRF is pseudorandom, then our system has path privacy property under the semantic security of ElGamal re-encryption.*

Proof (sketch). Assume that our system is not path private, namely, there exists an adversary $\mathcal{A}$ that breaks the path privacy of our system. Then we can construct an adversary $\mathcal{B}$ to break the semantic security of ElGamal encryption system under re-encryption. $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine and maintains a list L to answer $\mathcal{A}$'s queries as follows.

Suppose the public key of an ElGamal encryption cryptosystem is pk, and its corresponding private key is sk. Adversary $\mathcal{B}$ can break the semantic security of the system under re-encryption. $\mathcal{B}$ firstly simulates a path authentication system; it initializes the system the same as Initialization step defined in Section 4.2, except that the public and private keys of the manager are set to pk and sk, respectively. Note that $\mathcal{B}$ knows all the secret keys of the readers, but it does not know the value of sk. Then an adversary $\mathcal{A}$ starts the path-privacy experiment. In the learning phase of $\mathcal{A}_1$, when $\mathcal{A}_1$ queries the oracles, $\mathcal{B}$ answers the queries. $\mathcal{B}$ can answer the

queries to O_1, O_2 and O_4 directly. However, $\mathcal{B}$ does not have the private key sk, hence in case $\mathcal{A}_1$ queries the O_3 with a state $\{c_i, t_i\}$, $\mathcal{B}$ cannot decrypt c_i to get ID_i and compare the value of t_i with v_i in the database. In order to answer the queries to $\mathcal{O}_3$, $\mathcal{B}$ maintains a list L that records the history of each oracle's operations. Firstly, $\mathcal{B}$ inserts the tuples (ID_i, c_i, t_i, v_i) for $i = 1$ to n into list L. Then, each time a tag's state is changed, $\mathcal{B}$ adds a link between the tag's new state and old state. With the list L, given a tag's state, even through $\mathcal{B}$ cannot decrypt the ciphertext, it can get the tag's ID through the records of the tag's state in list L. Thus $\mathcal{B}$ can answer the queries to $\mathcal{O}_3$ by searching the database and comparing t_i with v_i. At the end of the learning phase, $\mathcal{A}_1$ outputs two tags T_0 and T_1, a path P with no less than k steps, st. Suppose that the state of T_0 is $\{c_0, t_0\}$, the state of T_1 is $\{c_1, t_1\}$. $\mathcal{B}$ firstly submits the two messages $\{c_0, c_1\}$ to $\mathcal{O}_{re-encrypt}$. $\mathcal{O}_{re-encrypt}$ randomly chooses $b \in \{0, 1\}$, and re-encrypts c_b to c'_b under the public key pk. Then $\mathcal{B}$ sends $S = \{c'_b, r\}$ to $\mathcal{A}_2$, where r is a random string. Note that actually, $\mathcal{B}$ should provide $\{c'_b, t_b\}$ to $\mathcal{A}_2$, where t'_b is the new value of t_b after been processed by k readers in path P. We argue that $\{c'_b, r\}$ and $\{c'_b, t'_b\}$ contain same information that can be used by $\mathcal{A}_2$. $\mathcal{A}_2$ cannot get any information from t'_b since the function PRF is a pseudorandom function. $\mathcal{A}_2$ guesses the value of b by analyzing $\{c'_b, r\}$. $\mathcal{B}$ outputs whatever output by $\mathcal{A}$.

Assuming the pseudorandomness of PRF, the advantage of $\mathcal{B}$ to break the semantic security of ElGamal under re-encryption is the same as the advantage of $\mathcal{A}$ to break the path privacy of the system. Since the ElGamal encryption scheme under re-encryption is semantic secure, hence our system is path private. □

6.2.4 Performance Analysis

Computational requirement: Our scheme does not require the tags to perform any computation. All the computation will be performed at the reader side. To update a tag, each reader requires one re-encryption operation and one computation on PRF.

For a manager to verify a tag's validity, it requires one decrypting operation and one comparison.

Storage requirement: Each tag T_i's state S_i consists of $\{c_i, t_i\}$. c_i is ElGamal ciphertext on ID_i which requires $2 \cdot 160$ bits. t_i is the path mark, generated by the PRF, thus 160 bits are sufficient. Therefore 480 bits storage is required for each tag. The protocol can thus be implemented with the standard EPC Class 1 Gen 2 tag with an extensible EPC memory bank (scalable between 16-480 bits), a scalable user memory bank (64-512 bits), which are available on the market.

On the reader side, the issuer stores a copy of system parameters including pk and k_j, for $0 \leq j \leq m$, m is the number of normal readers. So the storage requirement for the issuer is $O(1)$. Each normal reader R_j at step v_j needs to store the public key pk of the system and its own key k_j, the storage requirement for each normal reader is $O(1)$. Each manager stores a copy of sk. It also maintains a database DB, for each tag T_i, DB stores the verification information (ID_i, v_i). The storage requirement for a manager is $O(n)$, n is the number of the tags. As a tag's record takes 480 bits, a manager with 1GB storage can stores more than 17 million tags' records.

Compare to TRACKER [7, 8], our system is more practical. Since the tags' paths are predetermined in the initial stage, there is no need to store the path information on tag. A manager can perform path verification by simple comparison. Consequently, the storage and computational requirements on updating tags are reduced. The comparisons of storage and computational requirements between our protocol and TRACKER are shown in Table 1. Note that in comparing the computational load, we omit the cheap operations such as hash operation, computing PRF, and point addition on elliptic curve etc. We only count the expensive operations such as point multiplication on elliptic curve.

	TRACKER [7]	Our protocol
storage requirement		
tag	960 bits	480 bits
issuer	$O(1)$	$O(1)$
normal reader	$O(1)$	$O(1)$
manager	$O(n + vp)$, vp is the number of valid paths n is the number of tags	$O(n)$, n is the number of tags
computational requirement of processing a tag (operation on elliptic curve)		
issuer	8 point multiplication	2 point multiplication
normal reader	10 point multiplication	2 point multiplication
manager	5 point multiplication	1 point multiplication

Table 6.1: Comparisons of TRACKER and Our Protocol

6.3 Path Authentication Protocol for Dynamic Supply Chain Systems

6.3.1 Dynamic RFID-Enabled Supply Chain

A supply chain consists of multiple entities. We model a dynamic supply chain according to three properties: the affiliation of each entity, the membership management of the supply chain and the logistics flow of the supply chain. In a dynamic supply chain, each entity is independent of each other; any entity can freely join or leave; the logistics flow is not fixed.

An RFID-enabled dynamic supply chain management system should meet the requirements below.

- Each reader is independent with other readers and has an unique ID. Note that some readers may belong to the same entity. Even that, we assume the readers are independent with each other and have no pre-existing connections, such as network connection among themselves to cater for the most flexible deployment condition.

- Each reader does not share its secret (eg. private key) with other readers.
- Each reader in the supply chain is isomorphic, with the same functionalities. Each reader should be able to initialize the tags, identify the tags, verify the tags and update the tags.

Figure 6.5 illustrates our adversary model. The entities in a supply chain provide relative secure environments within their teritories, where the tags are beyond the accessible distance of an adversary. During the transportation of tagged goods between entities, a "hit and run" adversary can only approach the goods for a short period of time from a not-so-close distance.

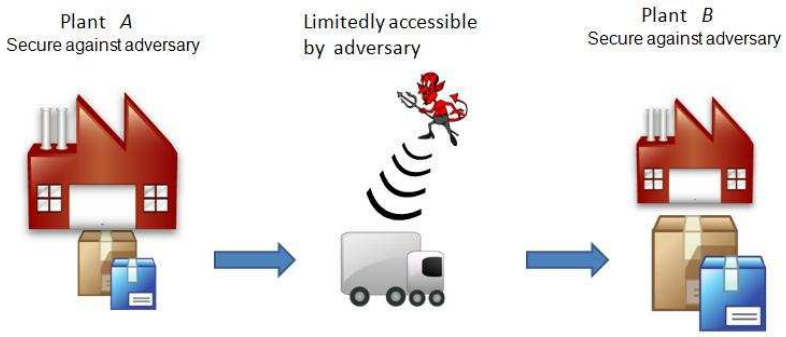

Figure 6.5: Adversary Model of Supply Chain

The "hit and run" adversary model was firstly proposed by Ari Juels [33]. Several works [40, 39, 35] have also adopted this model in designing secure RFID systems. The rationals to adopt this model in supply chains are: 1) the tagged goods usually rapidly change their physical locations and ownerships so that it is difficult for an adversary to keep in the working distance of the tags (normally no more than ten meters); 2) as both readers and tags work in short range, an adversary bringing a reader into a monitored environment like a shop or warehouse might face difficulties in attempting prolonged intelligence gathering [33].

In dynamic supply chains, we list the following practice requirements for designing RFID-enabled path authentication:

- Any valid reader can extract a tag's ID and the exact path that the tag has passed through in the supply chain. However, no readers needs to store the information of all possible paths in its own database in advance.

- An adversary cannot create new tag or modify existing one without being detected. A tag cannot pass the verification process for a path by which it has not passed.

- No efficient adversary can link the state information stored in a tag to the tag's identity. No efficient adversary can distinguish whether two tags have taken the same path or not.

- Path authentication can be performed on general EPC Class 1 Gen 2 tags, which have at most 1088 bits and no computational capability.

6.3.2 Challenges and Solution Sketch

The challenges of designing our scheme come in two aspects. First, the tag storage is restricted to no more than 1088 bits for the most popular low-cost standard C1 G2 tags. In order to prove that a tag has been processed by a series of entities, the tag should carry certain credentials generated by the entities. To avoid complicated key management at item level, a natural way is to use the entities' signatures on a tag's ID as the credentials. However, it is not practical to store all signatures and public keys on a tag as the tag's storage is limited, especially in the case that such information may continually increase as the tag goes through more entities. Our solution keeps the information stored on a tag in constant size, which is less than 1088 bits, satisfying the storage constraint for EPC C1 G2 standard tag. In our solution, an ordered multi-signature scheme [9] is adopted to generate a constant-size signature of the entities on the tag's ID. A path index is used to indicate the series of entities which have processed a tag. The index is stored on the tag instead of the entities' public keys, while the detailed information about the path is stored

on a trusted server such as EPCglobal Discovery Server.

The second challenge is to reduce the communication load between an entity and the trusted server when verifying a signature. To verify a signature, an entity queries the trusted server with the index of the path. The server sends the public keys of the entities in the path to the entity. The communication load increases as the path getting longer in a naive solution. We reduce the communication load to a constant level regardless of path length in our solution. Due to the specific constructions of the underlying ordered multi-signature scheme, in our scheme, the server only needs to send an aggregated "path public key" instead of the public keys of the entities. With the "path public key", one can verify whether a signature is generated orderly by the entities in the path or not. The "path public key" even has smaller size than a single public key. In the case where a batch of tags share the same path, a verifier only needs to query the trusted server once for the aggregated "path public key" in practice.

The security of our scheme relies on the unforgeability of the underlying ordered multi-signature scheme. Our scheme is secure in a sense that an adversary cannot forge any valid tag or path. To protect the privacy of each tag, we take advantage of supply chain's batch processing property. In a batch, each tag's information is encrypted with the same key. The key is divided to several shares with a secret sharing scheme. Each tag stores a share of the key together with its encrypted information. Only authorized readers can access the whole batch of tags, recover the key and then decrypt the contents stored on the tags. Our scheme is privacy preserving in a sense that an adversary cannot identify any valid tag, or distinguish whether two valid tags have taken the same path or not. Our scheme leverages on standard EPCglobal network and can be implemented on standard EPC class 1 generation 2 tags with only 720 bits storage and no computational capability.

6.3.3 Our Protocol

Our system contains three components: tags, readers and a trusted server. Each tag stores a tag ID, a path code, and a signature on the tag's ID generated by the readers in the path. Any valid reader has a pair of public key and private key and a random number used for generating the path code. A valid reader is able to extract each tag's ID and the path code; while to verify them, it needs to connect with the trusted server which stores the reader's public information and detailed path information. In designing a secure and privacy-preserving path authentication scheme, we use the following building tools.

Building tools

Bilinear map: See Section 5.3.

Path encoding method: Noubir et al. [51] proposes to encode a software's state machine using polynomials such that the exact sequence of states visited during run-time generates a unique "mark". We adopt this technique in generating the path code. Suppose there is a path $P_v = \{R_{v_1}, \cdots, R_{v_{l_v}}\}$, where l_v is the length of path P_v, v_i represents the reader's identity of the ith step in path P_v; we assign each reader R_j with a unique random number $a_j \in \mathbb{F}_q$, where q is a large prime. A path is represented with a polynomial on $\mathbb{F}q$. Then the polynomial corresponding to a path $P_v = \overrightarrow{R_{v_1} \cdots R_{v_l}}$ is defined below (all operations are in $\mathbb{F}q$.):

$$Q_{P(x)} := \sum_{i=1}^{l} a_{v_i} x^{l-i} \tag{6.5}$$

Given a generator x_0 of $\mathbb{F}_q$, we calculate the path code as $\phi(P_v) := Q_{P_v}(x_0)$ and identify a path P_v using its polynomial evaluation $\phi(P_v)$.

Secret Sharing Scheme A (τ, n)-secret sharing scheme is an algorithm that divides data D into n pieces in such a way that: 1) knowledge of any τ or more pieces makes D easily computable; 2) knowledge of any $\tau - 1$ or fewer pieces leaves D

completely undetermined (in the sense that all its possible values are equally likely) [59]. We adopt the Tiny Secret Sharing (TSS) [35] proposed by Juels et al. in our construction.

Ordered Multisignature Scheme Ordered multisignature scheme (OMS) allows signers to attest to a common message as well as the order in which they signed. The concept is raised by Boldyreva et al. in [9].

A construction of OMS is provided in [9], and we denote it as BGOY-OMS scheme from now on. We summarize BGOY-OMS system as follows. Each BGOY-OMS system has global information $I = (p, \mathbb{G}, \mathbb{G}_T, e, g, H)$, where $(p, \mathbb{G}, \mathbb{G}_T, e)$ is generated by a bilinear-group generation algorithm $\mathcal{G}$, g is a random generator of $\mathbb{G}$, and $H : \{0,1\}^* \to \mathbb{G}$ is cryptographic hash function.

- **Key Generation:** On input I, the algorithm chooses random $s, t, u \in \mathbb{Z}_p$ and returns $(S = g^s, T = g^t, U = g^u)$ as pk and (s, t, u) as sk.

- **Signing:** On inputs sk_i, m, σ, $L = (pk_1, \cdots, pk_{i-1})$, the algorithm first verifies whether Equation 2 defined below holds and if not, outputs $\perp$. (This step is skipped for the first signer, i.e. if $i = 1$, for whom σ is defined as $(1_G, 1_G)$.) Then it parses σ as (Q, W) and chooses random $w \in \mathbb{Z}_p$ and computes $W' = W \cdot g^w$, $X = (W')^{t_i + iu_i}$, $Y = (\prod_{j=1}^{i-1} T_j (U_j)^j)^w$ and $Q' = H(m)^{s_i} \cdot Q \cdot X \cdot Y$. Finally, it returns (Q', W').

- **Verification:** On inputs $\{(pk_1, \cdots, pk_n), m, \sigma\}$, the algorithm first checks that all of $pk_1, \cdots, pk_n$ are distinct and outputs 0 if not. Then it parses σ as (Q, W) and checks if

$$e(Q, g) \stackrel{?}{=} e(H(m), \prod_{i=1}^{n} S_i) \cdot e(\prod_{i=1}^{n} T_i (U_i)^i, W). \tag{6.6}$$

  If so, it outputs 1. If not, it outputs 0.

Protocol Details

The tag information, including tag ID, path code, and signature is encrypted. The tags in the same batch share the same encryption key. Then the encryption key is distributed using TSS secret sharing scheme and each tag stores a share of the key together with encrypted data. A valid reader can collect enough shares, recover the key, and decrypts the information of each tag. For each tag, after decryption, a valid reader obtains the tag ID, a path code, and a signature on the tag ID. Querying a trusted server with the path code, the reader gets the aggregated "path public key" of the path which is computed from the readers' public keys, and uses it to verify the signature. If the tag is valid, then the reader can update the path code and the signature and encrypt them with a new random key. Finally, the reader stores the new tag information on the tag. The tags are processed by batch in supply chain management. Polynomial signature based path encoding method [51], BGOY-OMS [9], and TSS [35] are incorporated in the design of our system.

System Setup: A BGOY-OMS system is set up by running a bilinear-group generation algorithm $\mathcal{G}$ for output $(p, \mathbb{G}, \mathbb{G}_T, e, g, H)$. Choosing a large prime q, and a random number $x_0 \in \mathbb{Z}_q$, we get $I = (p, q, \mathbb{G}, \mathbb{G}_T, e, g, H, x_0)$, which is the global information for the scheme. Assume that there are m readers in total. Each reader R_j is assigned with public key $pk_j = (S_j, T_j, U_j)$ and secret key $sk_j = (s_j, t_j, u_j)$, where s_j, t_j, u_j are randomly chosen from $\mathbb{Z}_p$, $1 \leq j \leq m$. Each reader R_j is also assigned with a random number $a_j \in \mathbb{Z}_q$, where a_j will be used in generating a path code.

A trusted server publishes the global information I, each reader's public key pk_j and random number a_j. The trusted server also stores each path P_v 's information, including "$pathcode_v$, l_v, $P_v = \{R_{v_1}, R_{v_2} \cdots, R_{v_{l_v}}\}$, $ppk_v = (ppk1_v, ppk2_v)$ $= (\prod_{j=1}^{l_v} S_{v_j}, \prod_{j=1}^{l_v} T_{v_j}(U_{v_j})^j)$", where l_v is the number of readers in path P_v, v_j denotes the identity of the jth reader in path P_v, $pathcode_v$ is the path code of P_v generated using Equation (1) in $\mathbb{F}_q$, and $ppk_v = (ppk1_v, ppk2_v) = (\prod_{j=1}^{l_v} S_{v_j},$

$\prod_{j=1}^{l_v} T_{v_j}(U_{v_j})^j)$ is each path's public key stored in a path record. With ppk_v, a valid reader can verify the signature on the tag without knowing any other reader's individual public key. This will reduce the communication load between a reader and the trust server. In case a reader needs to verify the signature on a tag based on all involving readers' public keys, it can also get the public keys from the trusted sever. Table 1 shows the contents stored on the trusted server.

Batch initialization of the tags: Suppose that a batch $\mathcal{T}$ of n tags enters in a supply chain, where each tag is denoted as T_i with unique ID id_i, for $i \in \{1, \cdots, n\}$. The tags can be initialized by a reader valid R_x, $1 \leq x \leq m$. In particular, R_x generates a key k and n shares of k using TSS-scheme, where each share is denoted as s_i, for $1 \leq i \leq n$. For each tag T_i, R_x generates a signature $\sigma_i = (Q_i, W_i)$ on the tag ID id_i under its private key using BGOY-MOS scheme. Then R_x sets $pathcode_i$ of each tag T_i to a_x. Finally, R_x encrypts $(id_i, \sigma_i, pathcode_i)$ with the key k, and stores $\{s_i, E_k(id_i, \sigma_i, pathcode_i)\}$ on T_i. After initializing the batch of tags, R_x queries the trust server to check whether the path $P = \{a_x, 1, \{R_x\}, ppk = (S_x, T_x U_x)\}$ already exists; if not, R_x inserts path P to the database on the trust server. Then R_x releases the batch of tags into the supply chain.

Interactions between reader and tag: When a batch $\mathcal{T}$ of tags in the supply chain arrives at reader R_y, where $1 \leq y \leq m$. Each tag T_i in the batch stores a state $st_i = \{s_i, E_k(id_i, \sigma_i = (Q_i, W_i), pathcode_i)\}$. Reader R_y firstly reads all the tags in $\mathcal{T}$ to get st_i for all $1 \leq i \leq n$. Using at least τ shares, R_y recovers a key k, and decrypts the information on each tag $E_k(id_i, \sigma_i, pathcode_i)$ and gets $\{id_i, \sigma_i = (Q_i, R_i), pathcode_i\}$. According to $pathcode_i$, R_y gets the path's information $P = \{pathcode_i, l, \{R_{P_0}, \cdots, R_{P_l}\}, ppk = (ppk_1, ppk_2)\}$ form the trusted server. If $e(Q_i, g) = e(H(id_i), ppk_1) \cdot e(ppk_2, W)$, then T_i passes the verification. To update the batch of tags, R_y generates a new key k' and n shares of k' in TSS, where each share is denoted as s'_i. For each tag T_i, R_y shall update both the signature σ_i and the path code $pathcode_i$. To do these, R_y chooses a random number w in $\mathbb{Z}_p$, computes $W'_i = W_i \cdot g^w$, $Q'_i = W_i'^{t_y + (l+1)u_y} \cdot ppk_1^w \cdot Q \cdot H(id_i)^{s_y}$ and

$pathcode'_i = pathcode_i \cdot x_0 + a_y$. R_y updates each tag T_i by writing $\{s'_i, E_{k'}(id_i, \sigma'_i = (Q'_i, W'_i), pathcode'_i)\}$ to the tag. Finally, the reader queries the trusted server: if the path $P_{new} = \{pathcode'_i, l+1, \{R_{P_0}, \cdots, R_{P_l}, R_y\}, ppk = (ppk_1 \cdot S_y, ppk_2 \cdot T_y U_y^{l+1})\}$ does not exist in the trusted server, then path P_{new} will be added for future queries.

Implementation details: TSS scheme can be implemented with Reed-Solomon code [55]. A Reed-Solomon code is specified as $RS(N, K)_S$. A codeword has S bits. A reader chooses a pre-key with $K \cdot S$ bits and encodes the pre-key to N shares with each share S bits. A hash value of the pre-key is used as the encryption key for a batch of N tags. Reed Solomon encoding and decoding have been implemented on an Intel Core 2 CPU 6320 1.86GHz. Choosing the parameters (N, K, S) as $(32768, 16384, 16)$, the fastest algorithm achieves 6.17 Mbytes throughput per second for encoding, and 2.73 Mbytes throughput per second for decoding. Both encoding procedure and decoding procedure consume less than one second. Suppose there are 20000 tags in a batch, one can firstly choose a pre-key with $16 * 16384$ bits, encode them to 32768 symbols. Each share contains one symbol. Then one can select 20000 shares, randomly store a share on each tag. Any reader that can successfully read more than 16384 tags is able to get the pre-key. Regarding the encryption algorithm, Blowfish [2] is an appropriate choice. Blowfish has good performance that achieves 64.386MB per second of encryption throughput on a Pentium 4 2.1 GHz processor under Windows XP SP 1.

6.3.4 Analysis

Both the security and privacy of our scheme rely on the security and privacy properties of the underlying secret sharing scheme, encryption scheme and OMS scheme.

Security

We assume that in supply chain management, an adversary's goal is to insert counterfeited goods into the supply chain. The security goal of our system is to prevent

an adversary from forging a tag's internal state so that the tag is considered from a reliable source and has gone through a valid path that has not actually been taken by the tag in the supply chain.

The security of our solution is based on the unforgeability of the BGOY-OMS scheme. Intuitively, the unforgeability of BGOY-OMS scheme can be described as follows: given an uncorrupted party with key pair (pk, sk), a forger cannot generate a valid BGOY-OMS signature of the uncorrupted party on any message m' if the forger does not know the party's signature on m'. Note that in the original BGOY-OMS unforgeability game, given a key pair (pk, sk), the adversary is able to get signature on any tag's ID under the key sk so as to learn useful information. In our system, the adversary cannot get a valid reader's signature on any ID it chooses since the valid reader will verify the genuineness of any tag before generating a signature on its ID. Hence, the adversary in our system is weaker than the adversary in the BGOY-OMS unforgeability game. BGOY-OMS scheme is secure against the forger; hence it is also secure against the adversary in our system. Unless an adversary has corrupted all the readers in a path with path code $pathcode$, it cannot forge a tuple $(ID, \sigma, pathcode)$ such that σ is a valid signature on ID generated by the readers in the path.

While an adversary cannot forge any valid new tuple $(ID, \sigma, pathcode)$ by itself, another way to counterfeit an tag is to use an existing valid tuple in the supply chain. From this aspect, the counterfeiting countermeasure of our system relies on the batch processing property. Only valid readers can read the whole batch of tags and decrypt the contents of the tags. A "hit and run" adversary is not able to read more than $\tau - 1$ tags in a batch , thus cannot recover the decryption key. In EPC Class 1 Gen 2 tags, the user memory bank can be protected by access pin. We use the encryption/decryption key as the access pin for the tags; therefore, an adversary cannot get the complete content stored on a tag and our system is secure against the counterfeiting attack.

Privacy

The privacy of RFID path authentication can be considered at two levels: tag unlinkability and path unlinkability. Tag unlinkability requires that no efficient adversary can link the state information stored in a tag to the tag's identity. Path unlinkability requires that no efficient adversary can distinguish whether two tags have taken the same path or not. In our scheme, each tag stores a copy of encrypted ID, pathcode and signature together with a single share of the encryption key. The privacy of our system relies on honest behaving of valid readers. Each valid reader uses a new random key to encrypt the updated state for each tag. An hit-and-run adversary who cannot collect enough shares for recovering the encryption keys cannot distinguish between any two ciphertexts of the same tag and any two ciphertexts of different tags. Thus our scheme preserves tag unlinkability. Similarly, our scheme preserves path unlinkability since an adversary cannot obtain any information about a tag's path from the content of the tag.

Performance

We analyze the performance of our solution in three aspects: computation requirements, communication requirements and storage requirements.

Computational requirements: Our scheme does not require tag to perform any computation. All the computation can be performed at RFID reader side. In computing the computational load of a reader, we omit the cheap operations such as Reed Solomon encoding, decoding, encryption and decryption using Blowfish, hash operation, and point addition on elliptic curve. We only count the relative expensive operations such as point multiplication on elliptic curve and paring. A reader needs to perform three paring operations to verify the signature and four point multiplication four updating the signature in each tag.

Due to batch processing in supply chain, we can reduce the computational cost if a batch of tags share the same path. Assuming that a batch of tagged goods is

transferred in a supply chain without being mixed with other goods, then a reader can sign the batch of tags with the same random number. That is, each tag shares the same randomization factor W in the signature. To update the signature, the reader firstly computes (W', X, Y), which requires three point multiplication operations, and uses (W', X, Y) to generate each tag's signature. With (W', X, Y), the reader's computational load is reduced to one point multiplication in generating each tag's signature. For signature verification, since each tag in the batch stores the same randomization factor W in the signature, a reader can pre-compute $e(ppk_2, W)$ and store the value. The computational requirements for each tag is reduced to two paring operations. Since all the computation can be performed on reader side, our solution is applicable on standard low-cost tags with no computation capability.

Communication requirement: The verification of each tag's ID and path code requires a valid reader to connect to the trusted server. The reader sends the path code to the server, then the server returns necessary path's information. In the case that a batch of tags passed the same path, the reader needs only one path information from the server. Since batch processing is commonly used in supply chain practice, the communication load required for processing a batch of tags should be almost constant. The communication load can be further reduced if a reader stores path information for frequently used paths in its own database.

Storage requirement: In each tag T_i, we need to store $\{s_i, E_k(id_i, \sigma_i = (Q_i, W_i), pathcode_i)\}$. s_i is a share of encryption key k. As we implement TSS scheme with Reed-Solomon code, s_i can be a symbol, which we use 16-bit string (a symbol's length depends on the parameters of Reed-Solomon code). Tag ID id_i is an EPC code, which has 96 bits. Tag signature σ_i generated by BGOY-OMS scheme consists of two elements on $\mathbb{G}$. For 80-bit security level with embedding degree $k = 2$, an element in $\mathbb{G}$ can be represented in 512 bits. For embedding degree $k = 6$, the length can be reduced to 237 bits [11]. Hence, the storage requirement for σ_i is at least 474 bits. We use 80 bits to represent a path ID, which supports at most 2^{80} different path codes. We adopt Blowfish block cipher [58] for encryption which has

a block size of 64-bits. Thus, we need 720 bits in total. Our system can thus be implemented with EPC Class 1 Gen 2 tags with an extensible EPC memory bank (scalable between 16-480 bits), a scalable user memory bank (64-512 bits), a TID bank (32 bits), and a reserved bank (64 bits), which are available on the market.

6.4 Summary

In this chapter, we analyzed the existing security and privacy notions for RFID path authentication in [7]. We proposed the first single-game-based privacy notion for path authentication and proved that it implies the existing notions. We also proposed two RFID path authentication schemes, one for closed supply chains, and the other for dynamic supply chains. Our path authentication schemes can be implemented on standard EPC Class 1 Generation 2 tags, and they outperform the existing path authentication solutions [7, 8] in RFID-enabled supply chains.

Chapter 7

Conclusion and Future Work

In this thesis, we have addressed the security and privacy issues in RFID-enabled supply chains. In brief, we have made the following contributions:

- We analyzed two typical protocols that were asserted to have the most desired security properties for RFID communications. We discovered that these protocols are vulnerable to a series of active attacks including server impersonation, tag impersonation, and de-synchronization. We proposed revised protocols to eliminate the vulnerabilities without violating of any security properties. The storage and computational requirements of our proposed solutions are comparable to the existing solutions.

- We investigated the security, visibility, and efficiency requirements in RFID-enabled supply chain systems. High efficiency is particularly desirable in RFID-enabled supply chains since a large quantity of tagged products are routinely processed and exchanged among multiple supply chain parties. In order to enhance the efficiency of an RFID-enabled supply chain system without sacrificing its security, we distinguish the working environments into two security levels. In a relatively secure environment with no active attacks, our RFID system can be set to the weak security mode so as to provide a high processing speed. While in a relatively less secure environment that is exposed to

active attacks, our RFID system can be switched to the strong security mode so as to maintain strong unlinkability.

- Considering both internal adversaries and external adversaries, we investigated the security and privacy requirements of RFID system in 3PL supply chains. We provided two "group checking" protocols for a 3rd party to check the existence and originality of tags in a batch level without knowing the tag secrets. One protocol is based on aggregate MAC and the other is based on aggregate signature. Both protocols achieve the goals of protecting and restraining the third party. Comparing the usability and performance of these two schemes, we conclude that the aggregate MAC-based protocol outperforms the aggregate signature-based protocol.

- We analyzed and refined the existing security and privacy notions for RFID path authentication. We proposed the first single-game-based privacy notion for path authentication and proved that it implies the existing notions. We also proposed a path authentication protocol that satisfies the privacy notion. Our protocol can be implemented on standard EPC Class 1 Generation 2 tags, and it outperforms the existing path authentication solutions [7, 8] in closed supply chains. We also proposed a distributed path authentication scheme for dynamic supply chains. Our proposed solution leverages on sharing path information on standard EPCglobal network, and can be implemented on standard low-cost RFID tags with no computation capability and limited memory.

In the future, we will continue our work on the path authentication problem in RFID-enabled supply chains. We will study the relationships between our new privacy notion and the existing privacy notions in RFID-enabled supply chains, and will provide a formal analysis of our path authentication scheme for dynamic supply chains.

A new problem which has not been explored rigorously before is the reader path authentication problem. The reader authentication is different from the tag authen-

tication. While the tag path authentication checks whether a tag has passed through a valid path or not, the reader path authentication aims to verify whether a reader has processed a series of tags or not. The reader path authentication is also different from the grouping proof. While in the grouping proof, the tags are processed in the same location and at the same time, in the reader path authentication, however, the tags are processed individually (possibly at different locations and at different times). We will formally model the reader authentication problem, and propose a solution.

In the future, we would also like to explore the fast group checking problem in RFID-enabled supply chains. Given a batch of tags, most of current solutions directly authenticate the tags in item-level or batch-level. Fast group checking aims to detect the existence of counterfeited tags in a batch of goods without authenticating the tags. Unless counterfeited items are detected in a batch, the goods can be processed in a high speed. Fast group checking is particularly suitable for large-scale supply chains.

Bibliography

[1] ALIEN tags. http://www.alientechnology.com/tags/. Accessed: 2014-4-11.

[2] The Blowfish encryption algorithm. https://www.schneier.com/blowfish.html. Accessed: 2014-2-7.

[3] CAENRFID RT0005ET tags. http://www.caenrfid.it/en/CaenProd.jsp?parent=65&idmod=806. Accessed: 2014-4-11.

[4] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM side-channel(s). In *CHES*, pages 29–45, 2002.

[5] F. Armenio, H. Barthel, L. Burstein, P. Dietrich, J. Duke, J. Garrett, B. Hogan, O. Ryaboy, S. Sarma, J. Schmidt, K. Suen, K. Traub, and J. Williams. The EPCglobal architecture framework. EPCglobal Standards, Sept. 2007.

[6] G. Avoine. *Cryptography in radio frequency identification and fair exchange protocols*. PhD thesis, Lausanne, 2005.

[7] E.-O. Blass, K. Elkhiyaoui, and R. Molva. Tracker: security and privacy for RFID-based supply chains. *IACR Cryptology ePrint Archive*, 2010:219, 2010.

[8] E.-O. Blass, K. Elkhiyaoui, and R. Molva. Tracker: security and privacy for RFID-based supply chains. In *NDSS*, 2011.

[9] A. Boldyreva, C. Gentry, A. O'Neill, and D. H. Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In *CCS*, pages 276–285, 2007.

[10] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Eurocrypt*, pages 416–432, 2003.

[11] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[12] M. Burmester, B. de Medeiros, and R. Motta. Provably secure grouping-proofs for RFID tags. In *CARDIS*, 2008.

[13] M. Burmester and J. Munilla. Flyweight RFID authentication with forward and backward security, 2009.

[14] S. Cai, R. H. Deng, Y. Li, and Y. Zhao. A new framework for privacy of RFID path authentication. In *ACNS*, pages 473–488, 2012.

[15] S. Cai, T. Li, Y. Li, and R. H. Deng. Ensuring dual security modes in RFID-enabled supply chain systems. In *ISPEC*, pages 372–383, 2009.

[16] S. Cai, Y. Li, T. Li, and R. H. Deng. Attacks and improvements to an RFID mutual authentication protocol and its extensions. In *WiSec*, pages 51–58, 2009.

[17] S. Cai, Y. Li, T. Li, R. H. Deng, and H. Yao. Achieving high security and efficiency in RFID-tagged supply chains. *IJACT*, 2(1):3–12, 2010.

[18] S. Cai, Y. Li, and Y. Zhao. Distributed path authentication for dynamic RFID-enabled supply chains. In *SEC*, pages 501–512, 2012.

[19] S. Cai, C. Su, Y. Li, R. H. Deng, and T. Li. Protecting and restraining the third party in RFID-enabled 3pl supply chains. In *ICISS*, pages 246–260, 2010.

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

[21] T. Dimitriou. A lightweight RFID protocol to protect against traceability and cloning attacks. In *SecureComm*, pages 59–66, 2005.

[22] D. Dolev and A. C.-C. Yao. On the security of public key protocols (extended abstract). In *FOCS*, pages 350–357, 1981.

[23] D. N. Duc, J. Park, H. Lee, and K. Kim. Enhancing security of EPCglobal Gen-2 RFID tag against traceability and cloning. *SCIS*, Jan. 2006.

[24] M. Farb, Y.-H. Lin, T. H.-J. Kim, J. McCune, and A. Perrig. Safeslinger: Easy-to-use and secure public-key exchange. In *MobiCom*, pages 417–428, 2013.

[25] S. Fouladgar and H. Afifi. An efficient delegation and transfer of ownership protocol for RFID tags. In *EURASIP Workshop*, 2007.

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

[27] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Crypto*, pages 10–18, 1984.

[28] O. Goldreich. *Foundations of Cryptography: Volume 1*. Cambridge University Press, New York, NY, USA, 2006.

[29] P. Golle, M. Jakobsson, A. Juels, and P. F. Syverson. Universal re-encryption for mixnets. In *CT-RSA*, pages 163–178, 2004.

[30] D. M. Hein, J. Wolkerstorfer, and N. Felber. ECC is ready for RFID - a proof in silicon. In *SAC*, pages 401–413, 2008.

[31] A. Henrici and P. Müller. Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. In *PerSec*, pages 149–153, 2004.

[32] E. Inc. EPC radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860MHz-960MHz version 1.1.0. EPCglobal Standards, Oct. 2007.

[33] A. Juels. Minimalist cryptography for low-cost RFID tags. In *SCN*, pages 149–164, 2004.

[34] A. Juels. "Yoking-proofs" for RFID tags. In *PerCom Workshops*, pages 138–143, 2004.

[35] A. Juels, R. Pappu, and B. Parno. Unidirectional key distribution across time and space with applications to RFID security. In *USENIX Security Symposium*, pages 75–90, July 2008.

[36] A. Juels and S. A. Weis. Defining strong privacy for RFID. *TISSEC*, 13(1):7:1–7:23, 2009.

[37] J. Katz and A. Y. Lindell. Aggregate message authentication codes. In *CT-RSA*, pages 155–169, 2008.

[38] M. O. Koutarou, K. Suzuki, and S. Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *RFID Privacy Workshop*, 2003.

[39] M. Langheinrich and R. Marti. Practical, minimalist cryptography for RFID privacy. *IEEE Systems Journal, Special Issue on RFID Technology*, 1(2):115–128, 2007.

[40] M. Langheinrich and R. Marti. RFID privacy using spatially distributed shared secrets. In *UCS*, pages 1–16, 2007.

[41] T. Li and G. Wang. Security analysis of two ultra-lightweight RFID authentication protocols. In *SEC*, pages 14–16, 2007.

[42] Y. Li, R. H. Deng, and E. Bertino. RFID security and privacy. *Synthesis Lectures on Information Security, Privacy, and Trust*, 4(3):1–157, 2013.

[43] Y. Li and X. Ding. Protecting RFID communications in supply chains. In *ASIACCS*, pages 234–241, 2007.

[44] C. H. Lim and T. Kwon. Strong and robust RFID authentication enabling perfect ownership transfer. In *ICICS*, pages 1–20, 2006.

[45] C.-C. Lin, Y.-C. Lai, J. D. Tygar, C.-K. Yang, and C.-L. Chiang. Coexistence proof using chain of timestamps for multiple RFID tags. In *APWeb/WAIM Workshops*, pages 634–643, 2007.

[46] B. Liu and C.-H. Chu. Security analysis of EPC-enabled RFID network. In *RFID-TA*, pages 239–244, 2010.

[47] R. E. Liu and A. Kumar. Leveraging information sharing to increase supply chain configurability. In *ICIS*, pages 523–537, 2003.

[48] D. A. Menascé. Security performance. *IEEE Internet Computing*, 7(3):84–87, 2003.

[49] D. Molnar, A. Soppera, and D. Wagner. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In *SAC*, pages 276–290, 2005.

[50] D. Molnar and D. Wagner. Privacy and security in library RFID: issues, practices, and architectures. In *CCS*, pages 210–219, 2004.

[51] G. Noubir, K. Vijayananda, and H. J. Nussbaumer. Signature-based method for run-time fault detection in communication protocols. *Computer Communications*, 21(5):405–421, 1998.

[52] M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *RFID Privacy Workshop*, 2003.

[53] K. Osaka, T. Takagi, K. Yamazaki, and O. Takahashi. An efficient and secure RFID security method with ownership transfer. In *CIS*, pages 778–787, 2006.

[54] Piramuthu. On existence proofs for multiple RFID tags. In *PERSER*, pages 317–320, 2006.

[55] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[56] J. Saito, K. Imamoto, and K. Sakurai. Reassignment scheme of an RFID tag's key for owner transfer. In *EUC Workshops*, pages 1303–1312, 2005.

[57] J. Saito and S. Kouichi. Grouping proof for RFID tags. In *AINA*, volume 2, pages 621–624, 2005.

[58] B. Schneier. Description of a new variable-length key, 64-bit block cipher (blowfish). In *FSE, Cambridge Security Workshop*, 1994.

[59] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.

[60] B. Song. RFID tag ownership transfer. In *RFIDSec*, Budapest, Hungary, July 2008.

[61] B. Song and C. J. Mitchell. RFID authentication protocol for low-cost tags. In *WiSec*, pages 140–147, 2008.

[62] T. van Deursen and S. Radomirović. Algebraic attacks on RFID protocols. In *WISTP*, 2009.

[63] T. Van Le, M. Burmester, and B. de Medeiros. Universally composable and forward-secure RFID authentication and authenticated key exchange. In *ASIACCS*.

[64] Wang, Y. Li, Z. Zhang, and Z. Cao. Two-level path authentication in EPCglobal Network. In *IEEE RFID*, pages 24–31, April 2012.

[65] S. Weis, S. Sarma, R. Rivest, and D. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In *SPC*, 2003.

[66] L. Xiao, X. Zhang, and S. A. Kubricht. Improving memory performance of sorting algorithms. *ACM Journal of Experimental Algorithmics*, 5:3, 2000.

[67] A. C. Yao, M. Yung, and Y. Zhao. Adaptive concurrent non-malleability with bare public-keys. *IACR Cryptology ePrint Archive*, 2010:107, 2010.

[68] D. Zanetti, L. Fellmann, and S. Čapkun. Privacy-preserving clone detection for RFID-enabled supply chains. In *IEEE RFID*, pages 37–44, 2010.