MUTUAL AUTHENTICATION PROTOCOLS

FOR

RFID SYSTEMS

By

ASRAR AHMED OMER

Masters of Science in Computer Science

Oklahoma State University

Tulsa, Oklahoma

2007

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2007

MUTUAL AUTHENTICATION PROTOCOLS

FOR

RFID SYSTEMS

Thesis Approved:

Dr. Johnson Thomas

Thesis Adviser

Dr Istvan Joyner

Dr. Debao Chen

Dr. A. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to thank all my committee members who helped me in completing my thesis. Specially, Dr Johnson Thomas, who was my inspiration, he helped me in every step of the way. I like to personally thank him for all of his guidance and help.

I would like to thank all my friends specially Mustafa Hilal Qureshi and ER Sabri who helped me keep up my morale and gave me lot of good memories.

I would like to thank from the bottom of my heart my brother Masroor Ahmad for his support and help to reach this goal. I would like to thank my beloved parents Mr. Mehboob Ahmed (Late) and Mrs. Khalida Mehboob for all their support, help and prayers.

In the end I would like to thank ALLAH for giving me the patience and guidance that was needed to complete my research.

TABLE OF CONTENTS

Chapter Page

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

NOMENCLATURE

C	Counter value
DoS	Denial of service attack
$f_k(x)$	Pseudo-random function with x as variable and k as secret seed
ID	static identification number
K	Master key
K1	first associated key value
K2	second associated key value
$K1_{last}$	first associated key value used at last attempt
$K2_{last}$	second associated key value used at last attempt
n	pseudo-random number
n'	pseudo-random number generated by attacker
PRF	Pseudo-random function
PRNG	Pseudo-random number generator
TIN	tag-index number
XOR	Exclusive-OR operation

CHAPTER 1

INTRODUCTION

1.1 RFID Overview

Radio Frequency Identification (RFID) is a type of automatic identification system. The purpose of an RFID system is to enable data to be transmitted to a portable device, called a tag, which is read by an RFID reader and processed according to the needs of a particular application. The data transmitted by the tag may provide identification or location information or specifics about the product tagged, such as price, color, date of purchase, etc. RFID aims to identify objects remotely, with neither physical nor visual contact.

RFID system consists of three components:

1. An antenna or coil
2. A transceiver (with decoder)
3. A transponder (RF tag) electronically programmed with unique information.

The antenna emits radio signals to activate the tag and read and write data to it. Antennas are the conduits between the tag and the transceiver, which controls the system's data acquisition and communication. When an RFID tag passes through the electromagnetic zone, it detects the reader's activation signal. The reader decodes the

data encoded in the tag's integrated circuit (silicon chip) and the data is passed to the host computer for processing.

The use of RFID is tracking applications first appeared during the 1980's even though RFID was developed by allied forces in WWII so radar operators could distinguish between friendly and enemy aircraft [16]. RFID systems have also been used for a few years in commercial applications, for example in contact-less smart cards used on public transport. However, the boom that RFID technology enjoys today is chiefly due to the standardization and development of low-cost devices, so-called tags. This new generation of RFID tags has opened the door to various applications. For example in supply chains, to locate people, to combat the counterfeiting of expensive items, to trace livestock, to label books in libraries, etc.

However, theses tags also bring with them security and privacy issues. Security issues rely on classic attacks, e.g., denial of service, traffic analysis, spoofing, impersonation of tags or channel eavesdropping. These attacks are rendered more practicable because of the tag's lack of computational and storage capacity.

RFID raises issues linked to privacy, in particular the problem of traceability of objects and thus indirectly of people. RFID tags would permit everybody to track people using only low-cost equipment. This is strengthened by the fact that tags can not be switched off, they can be easily hidden, their lifespan is not limited, and analyzing the collected data can be efficiently automated. RFID tags can be attached without knowledge of consumer and this is major concern for privacy advocacy groups. The potential for widespread dissemination, misuse, unauthorized access, and disclosure of

personal information about consumers would increase exponentially and will create a new source of privacy concern for the public.

1.2 RFID Components

RFID system is an information tracking system that consists of wireless tag T, wireless reader R, and back-end database, B as shown in Figure 1.1.

Tag: T is comprised of an IC chip and antenna, and sends information to the RFID reader in response to a wireless probe.

Reader: R is a device that transmits a radio frequency probe signal to T, receives the information sent by T, and sends the information to the back-end database, B.

Back-End: B is a secure server that has a database and manages various types of information related to each T, e.g., ID, reader location, read time, and temperature of sensor. B resolves the ID of T from the information sent by T through authenticated R.

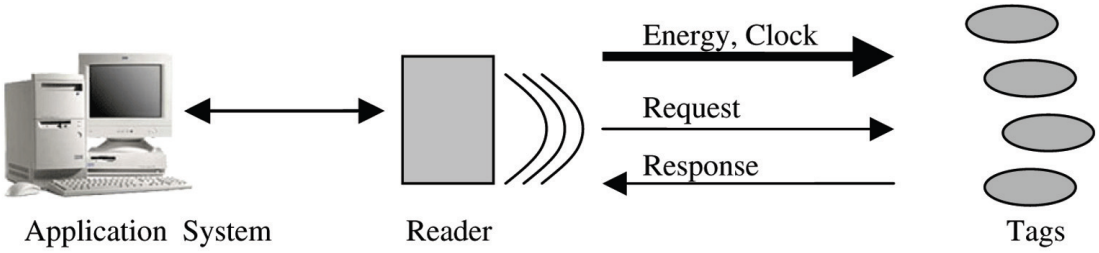

Figure 1.1: RFID System

As shown in the above figure when a reader probes a number of tags, it sends energy or signal in the form of a radio frequency. Tags on getting the signal energy wake up and perform the task within the timeframe given in the clock, as requested by the reader.

1.2.1 Transponders or RFID Tag:

Tags may either be actively or passively powered. Active tags contain an on-board power source, such as a battery, wile passive tags must be inductively powered via an RF signal from the reader. Active tags may be read from a greater distance than passive tags. Active tags may also record sensor readings or perform calculations in the absence of a reader. Passive tags can only operate in the presence of a reader and are inactive otherwise.

Tags are categorized into several types according to their physical characteristics and their applications. ISO/IEC categorizes RFID tags into *type* A and *type* B according to air interface since the characteristics of tags are mostly very different according to used radio frequency. On the other hand, EPC Global divides it into six categories, *Class 0-1, Class 2, Class 3, Class 4* and *Class 5* as a defacto standard. Class 0 and 1 are types of read only passive identity tags. Class 2 is passive too with additional functionality like memory or encryption. Class 3 is semi-passive tags and may support broadband. Class 4 is type of active tags and may be capable of broadband peer-to-peer communication with other active tags in the same frequency band and with readers. Class 5 is active tags and can support power Class 0-3 tags and communicate with Class 4 tags and with each other wirelessly. Tags can also be classified by their functionality [17]. Table 1 show five classes based on functionality defined by MIT Auto-ID Center.

Table 1: Tag Classification [17]

Class	Nickname	Memory	Power Source	Features
0	Anti-Shoplift Tags	None	Passive	Article Surveillance
1	EPC	Read-Only	Any	Identification Only

2	EPC	Read-Write	Any	Data Logging
3	Sensor Tags	Read-Write	Semi-Passive or Active	Environmental Sensors
4	Smart Dust	Read-Write	Active	Ad Hoc Networking

Universal deployment of RFID systems is nowadays mainly limited due to security and privacy concerns along with tag cost. Significant market penetration can be expected only if tags are priced below US$0.1 – 0.05 [3]. In this price range, tags come with following typical characteristics:

- Limited storage capacity.
- Limited computation power.
- Limited communication capabilities.
- No temper resistance.

This price barrier for low-cost tags restricts the range of gates in a tag number from 500 – 5000, and the number of gates for security purpose is limited to from 200 - 2000 [2]. Due to this limit, it is infeasible to use the existing cryptographic algorithms. [1].

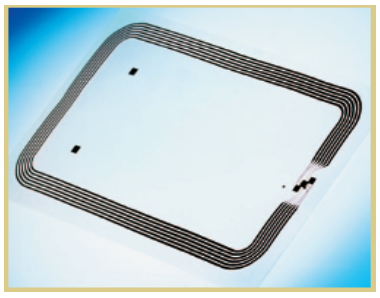

Figure 1.2: A Philips 1.Code RFID Tag [27]

1.2.2 Transceivers or RFID Reader:

A Reader may contain internal storage, processing power, or an interface to back-end databases to provide additional functionality. Readers may use tag contents as a look-up key into a database storing product information, tracking logs, or key management data.

The communication channel between tags and readers is generally considered insecure since the channel is based on the air interface. On the other hand, the communication channel between readers and back-end servers is considered as a secure channel. Readers can be either peripheral or a handheld device depending on the wireless network.

1.2.3 Back-end Servers (or Back-end Databases):

Back-end servers receive data from readers, enter the data into a database of their own, and provide access to the data in a number of forms that are useful to the sponsoring organization [17]. The back-end database may also perform functions on behalf of either the readers or tags. It is assumed that the communication channel between readers and back-end servers is secure channel like the existing VPN or SSL.

1.2.4 Operating Frequencies:

RFID tags and readers operate within several distinct frequency ranges, each of which is intended for specific application characteristic. According to the application purposes [17], table 1 shows the characteristics of frequencies that are available for RFID system.

Table 2: Frequency Classification [17]

Devices	Bandwidth	Typical Frequency	Application Example
Low	30 – 300 KHz	125 – 134 KHz	Short Range Applications: Live stock Identification, Antitheft Systems
High	3 – 30 MHz	13.56 MHz	Smart Card, Smart Card Label Applications, Baggage Tracking, Small Product Labeling
Very High	300 MHz – 3 GHz	U.S.A.: 902 -928 MHz or 2.45 GHz, EU: 865 -868 MHz or 5.8 GHz, Japan: 950 – 956 MHz, Korea: 908.5 – 914 MHz	Toll Collection Applications

1.3 Advantages of RFID System

The automated identification of objects with electromagnetic fields is the major purpose of RFID technology. It is expected that this technology will at least partly replace optical barcodes in the future. The potential benefits of a pervasive low-cost RFID system are enormous. World wide, over 5 billion barcodes are scanned daily [2]. The major advantages of RFID systems over optical identification with barcodes are:

- The operation without line-of-sight.
- The possibility to rewrite and modify data.
- The operation without any proper positioning.
- Tags can be scanned from distance of several meters.
- Faster than scanning barcodes.

A significant growth of the RFID market is predicted and a major driver for this rate is the falling prices of RFID – transponders.

There are various applications for low-cost tags such as logistics, point-of-sale checkouts, animal identification, item management in libraries, and waste management. More sophisticated RFID tags application includes health care, ticketing, road toll, electronic purse, access control for facilities, tracking people, key, RFID passports, anti-theft device and protection against counterfeiting. These RFID tags have the capability to replace magnetic stripe cards and classical contact smartcards.

Postal and courier mail services are expected to become the second largest market for RFD item level tagging following the retail sector.

1.4 Terminology and Basic Definitions

Some of the most commonly used terms are defined below, which will help in understanding the RFID system and its security models.

1.4.1 RFID System:

Such systems in which wireless devices transmit data and energy via radio frequency are called RFID systems.

1.4.2 Symmetric-Key Cryptography:

The signer and the verifier share a secret key, whereas the key exchange problem is solved between them. A single secret key is used for both encryption and decryption.

1.4.3 Asymmetric-Key Cryptography:

Signer has a pair of cryptographic keys – a public key and a private key. The private key is kept secret in the signer's environment, while the public key is widely distributed. A message encrypted with the public key can be decrypted only with the corresponding private key.

1.4.4 Authentication:

It is assurance of the identity of an entity at the other end of a communication channel. The protocol where one entity A is authenticated to entity B is called *unilateral authentication.* If both entities authenticate to each other, it's called *mutual authentication.*

1.4.5 Challenge-response Protocol:

In challenge-response protocols the verifier sends a challenge request to the claimant. This challenge can be a randomly chosen number or string which varies from one request to the other. The claimant "proves" its identity by manipulating the challenge using the secret which is associated with that entity. It is important not to show this secret to the verifier during the communication. After receiving the response from the claimant the verifier validates the response and can be sure whether the claimant knows the secret.

One-way Challenge-response Protocol: In this a timestamp mechanism is used. The signer A sends the encrypted timestamp t_A to the claimant B who decrypts it and verifies that the timestamp is acceptable.

$$A \rightarrow B: E_K (t_A)$$

Two-way Challenge-response Protocol: Its makes use of random numbers. In the case of two-way unilateral authentication, the claimant B must first send a random number r_B to the signer A who encrypts it and sends it back. Verification works by decrypting the response and comparing it with the random number sent.

$$A \leftarrow B: r_B$$

$$A \rightarrow B: E_K (r_B)$$

In the case of two-way mutual authentication, the entity A must encrypt the timestamp t_A and a randomly selected number r_A and send it to the second party B. Then, the random number r_A is encrypted and sent back to the originator who decrypts the message and compares the result with the send random number.

$$A \rightarrow B: E_K (t_A, r_A)$$

$$A \leftarrow B: E_K (r_A)$$

1.4.6 Hash Function:

The basic operation of hash functions is to map an element of larger domains to an element of smaller domains. This property is utilized in many non-cryptographic computer applications like storage allocation to improve performance. The purpose of hash functions in the cryptographic sense is to provide data integrity and message authentication.

A one-way hash function (OWHF) is a function which offers preimage and second preimage resistance. A collision resistant hash function CRHF is a function which is second preimage resistant and collision-freshness. Hash chain is a variant of hash functions and utilized in various areas.

1.4.7 Random Number Generator:

Random number generation is used in a wide variety of cryptographic operations, such a key generation and challenge-response protocols. A random number generator is a function that outputs a sequence of 0s and 1s such that at any point, the next bit cannot be predicted based on the previous bits. However, true random number generation is difficult to do on a computer, since computers are deterministic devices. Thus, if the same random generator is run twice, identical results are received. True random number generators are in use, but they can be difficult to build. They typically take input from something in the physical world, such as the rate of neutron emission from a radioactive substance or a user's idle mouse movements. Because of these difficulties, random number generation on a computer is usually only pseudo-random number generation. A pseudo-random number generator PRNG produces a sequence of bits that has a random looking distribution. With each different seed the pseudo-random number generator generate a different pseudo-random sequence. With a relatively small random seed a pseudo-random generator can produce a long apparently random string. Pseudo-random number generators are often based on cryptographic functions like block ciphers or stream ciphers.

1.4.8 Pseudo Random Functions:

A PRF is a deterministic function f: $\{0,1\}^n \rightarrow \{0,1\}^n$ which is efficient (i.e. computable in polynomial time) and takes two inputs *x, k* belongs $\{0,1\}^n$. We actually only consider *x* to be a variable and let *k* be a hidden random seed and function index, f $(x, k) = f_k (x)$.

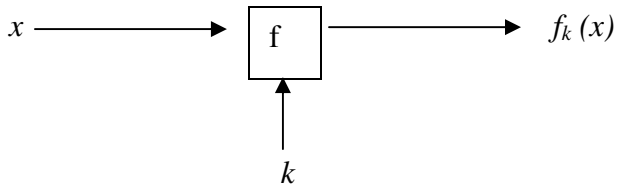

Figure 1.3: A Pseudo Random Function (PRF)

Most of the currently available protocols use hash functions, which are expensive in terms of chip cost, thus hindering the widespread use of this system. If we can reduce the cost by using simpler approaches than hash functions, the chip cost will be greatly reduced. Another problem seen in most protocols is database desynchronization when an attacker is able to change the values on either the database or at the tag end. Some attacks may even make the tag non-functional. Some protocols even suffer from scalability problems as the database has to compute for a particular tag.

Our approach uses primitive operations, pseudo-random generator (PRNG) and pseudo-random function (PRF), which can greatly reduce the cost. The first three protocols use a random number generator. Only one of these uses the PRNG on the tag side. Identifying the tag with their numbers (or Tag Identification Number) in the database can solve the scalability problem. By saving the previous values of keys in the database solves the problem of database desynchronization.

This thesis is organized as follows: In chapter 2, we look at the security and privacy issues, as well as the risks and threats currently faced by an RFID system. In chapter 3, review work done by other scientists and point out some of the deficiencies in their protocols. In chapter 4, we present the problem statement, that is, we describe the security problems investigated in this thesis. In chapter 5, we propose five new lightweight and ultra lightweight protocols for RFID systems. In chapters 6 and 7, we do

a security and performance analysis of all the proposed protocols. In chapter 8, we apply one of our protocols to a supply chain system. In the final chapter, we provide conclusions to the work done.

CHAPTER 2

SECURITY AND PRIVACY ISSUES

2.1 Security Issues

From the viewpoint of real world applications, the technical design of RFID readers and tags involve many risks. Existing RFID systems are vulnerable to many security risks and imply potential privacy problems, since it is very hard to implement the existing cryptographic algorithms due to the restricted computational power and the memory size of low-cost RFID tags. A common technology is used in both retail and library applications. Retail tags can be read at ten times the distance (20-30feet) of library tags (2-4feet). In addition, retail users of RFID will use the Electronic Product Code (EPC), a 96-bit number designed to uniquely label individual items. EPC uses will have access to the EPC Discovery Service, an aggregate database of tags collected from independent readers. Anyone with access to EPC Discovery can monitor or track the movement of a particular RFID-tagged item.

2.2 Privacy Issues

User privacy issues are considered as a big barrier to the proliferation of RFID system applications since the data of a tag can be transmitted by an illegal interrogation

without its bearer's notification. Two privacy issues are of major concern. One is the data leakage illegally from a tag. Another is the malicious tracking for the unique ID of a tag. A tag bearer has various objects that they do not want others to know including what they currently keep and what those objects are. If the tags are attached to those objects, the private information of tag bearers can be revealed regardless of their attention. The location privacy of tag bearers can be revealed through the response information from the tag although it is securely protected. In a RFID-labeled society, the value for commodities or products is mostly identified by the RFID. Thus, simple forgery such as copying information of a tag or even more sophisticated measures will be very attractive for malicious users and adversaries to disguise or impersonate.

2.3 Security Considerations

We consider the following as generally required security properties for RFID systems:

2.3.1 Confidentiality:

RFID tags must not get involved in processing personal data. In addition to it, data stored in a tag should not be gathered to trace the relationship between the tag and the tag bearer by illegitimate readers. The private information of a tag must be kept secure to guarantee user privacy. The tag information must be meaningless for its bearer even though it is eavesdropped by an unauthorized reader.

2.3.2 Authenticity:

The authenticity of a tag is at risk since the unique identifier of a tag can be spoofed or manipulated. The tags in general are not tamper resistant.

2.3.3 Availability:

Any RFID system can easily be disturbed by frequency jamming. However, denial-of-service attacks are also feasible on higher communication layers. The so called "RFID Blocker" exploits tag singulation (anti-collision) mechanisms to interrupt the communication of a reader with all or with specific tags.

2.3.4 Anonymity:

Although a tag's data is encrypted, the tag's unique identification information is exposed since the encrypted data is constant. An attacker can identify each T with its constant encrypted data. Therefore, it is important to make the tag's information anonymous.

2.3.5 Integrity:

Integrity in terms of the RFID environment as a security requirement is usually for data integrity between entities i.e. tags, readers, and back-end servers. This is due to the reason that the communication channel is not fault-tolerable and the data synchronization between entities can fail. Thus, integrity among entities must be guaranteed and data recovery mechanisms should be provided in case data loss occurs. In addition, if a tag's memory is rewritable, forgery is possible, so integrity for the tag's information also must be guaranteed.

2.3.6 Indistinguishability:

The value emitted by tag should not be such that the attacker can easily identify the tag.

2.3.7 Forward Security:

If the attacker is able to get the value from the tag, it should not give any past details.

2.4 Risks and Threats

The main risks and threats an RFID system can suffer are described below. These vary from system to system.

2.4.1 Physical Attacks:

In order to mount these attacks, it is necessary to manipulate tags physically, generally in a laboratory. Some examples of physical attacks are material removal through shaped charges or water etching, radiation imprinting, circuit disruption, etc. RFID tags offer little or no resilience against these attacks.

2.4.2 Denial of Service (DoS Attack):

A common example of this type of attack in RFID systems is the signal jamming of RF channels.

2.4.3 Counterfeiting:

These attacks consist of modifying the identity of an item, generally by means of tag manipulation.

2.4.4 Spoofing:

When an attacker is able to successfully impersonate a legitimate tag as, for example, in a man-in-the-middle attack.

2.4.5 Eavesdropping:

In this type of attacks, unintended recipients are able to intercept and read messages.

2.4.6 Database Desynchronization:

If the attacker is able to tamper with the responses from the tag and can create desynchronization of values at both tag and backend database.

2.4.7 Traffic Analysis:

In this attack, the person intercepts the messages and examines in order to extract information from patterns in communication. It can be performed even when the messages are encrypted and can not be decrypted. In general, the greater the number of messages observed, the more information can be inferred from the traffic.

2.5 RFID Standards

There exists a large variety of RFID systems and their main characteristics are defined by standards. Those standards typically describe the physical and the data link layers, covering aspects such as the air interface (frequency, coding, and modulations), communication protocol, bandwidth, anti-collision and security mechanisms.

RFID is a relatively heterogeneous radio technology with a significant number of associated standards. Figure 2.0 contains the most relevant technology standards.

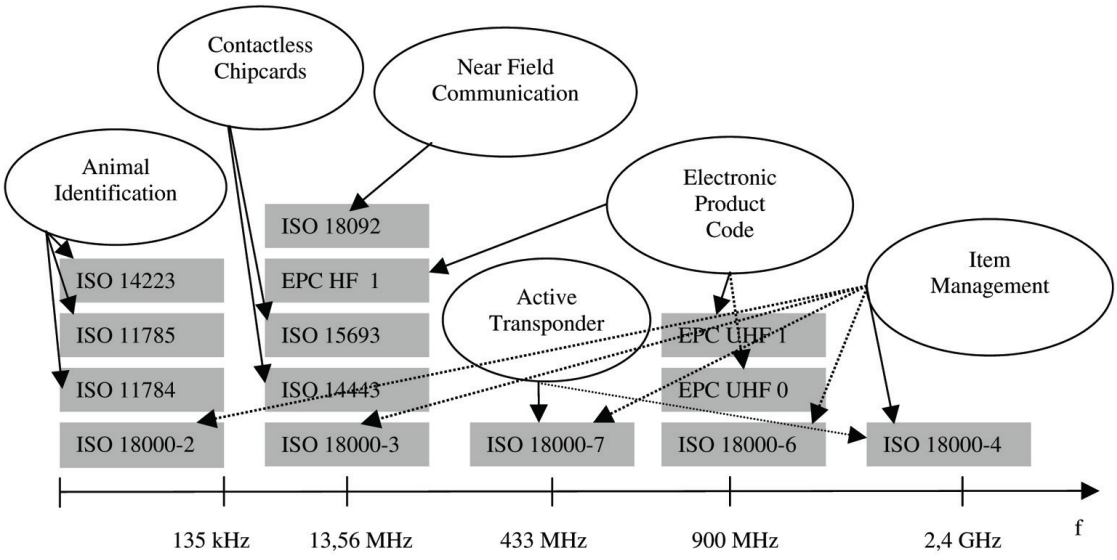

Figure 2.0: RFID technology standards and frequency bands [28]

2.5.1 Contactless Integrated Circuit Cards:

Contactless integrated circuit cards are special instances of identification cards as defined in ISO 7810. There are three types of contactless cards based on their communication range.

- Closed-coupled cards (ISO 10536). They operate at a very short distance to the reader (< 1cm).

- Proximity cards (ISO 14443). They operate at an approximate distance of 10cm of the reader. They usually possess a microprocessor and may be considered as high-end RFID transponders.
- Vicinity cards (ISO 15693). These cards have a range of up to 1 meter. They usually incorporate inexpensive state machines instead of microprocessors.

2.5.2 RFID in Animals:

ISO 11784, 1SO 11785 and ISO 14223 specify tags for animal identification in the frequency band below 135 kHz. The original standards defined only a fixed unique 64 bit identifier, but with the more recent ISO 14223 standard further read/write and write-protected data blocks are allowed.

2.5.3 Item Management:

ISO 18000 defines the air interface, collision detection mechanisms and the communication protocol for item tags in different frequency bands. Part 1 describes the reference architecture and parts 2 to 7 specify the system in different frequencies bands. Part 2, 3, 4, 5, 6, 7 specifies frequency (<134 kHz, 13.56 MHz, 2.45 GHz, 5.8 GHz, 900 MHz, 433 MHz) tags respectively.

2.5.4 Near-Field Communication (NFC):

NFC evolved from the RFID technology and is designed for interactions between tags and electronic devices in close proximity (<10cm). NFC is not designed for full

networking or transmission of large amounts of data, but should allow a convenient data exchange between cheap tags (e.g. smart labels) and electronic devices (e.g. PDA).

- NFCIP-1: The standards ETSI TS 102.190, ISO 18092 and ECMA 340 define identically the Near Field Communication Interface and Protocol. These protocols describe the air interface, initialization, collision avoidance, a frame format and a block oriented data exchange protocol with error handling. The communication modes can be either active or passive.
- NFCIP-2: The Near Field Communication Interface and Protocol – 2 (NFCIP – 2) specifies the communication mode selection mechanism (ECMA 352). This protocol deals with the situation that NFCIP-1, ISO 14443 and ISO 15693 devices all operate at 13.56 MHz, but with different protocols. Its specified that NFCIP-2 complaint devices can enter each of these three communication modes and are designed not to disturb other RF fields at the same frequency.

2.5.5 Electronic Product Code (EPC):

EPC was developed by the AutoID (Automatic identification) Center at MIT. The standardization is now within the responsibility of EPCglobal which is a joint venture between EAN International and the Uniform Code Council (UCC). The so-called EPC network is composed of five functional elements:

- EPC: The Electronic Product Code is a 96 bit number identifying the EPC version number, domains, object classes and individual instances.

- RFID System: An identification system which consists of RFID tags and readers. Tags can be of six different kinds (Class 0, 1, 2, 3, 4, 5) based on there functionality.
- Savant: The savant middleware offers “Processing Modules or Services” to reduce load and network traffic within the back-end systems.
- ONS: The Object Name Service is a networking service similar to the Domain Name Service (DNS). With ONS, the Electronic Product Code can be linked to detailed object information. The ONS servers return the IP address of the EPC information service which stores the associated information.
- PML: The Physical Markup Language is XML-based and provides a standardized representation of information from the EPC network.

CHAPTER 3

LITERATURE REVIEW

3.1 Privacy Protection Approaches

In this section we will discuss some of the approaches proposed by different scientist to protect the privacy of tag bearer.

3.1.1 Kill Command:

This solution was proposed by the Auto-ID center [18] and EPCglobal. In this scheme, each tag has a unique password, for example of 24 bits, which is programmed at the time of manufacture. Upon receiving the correct password, the tag will deactivate forever.

3.1.2 Faraday Cage:

Another way of protecting privacy of objects labeled with RFID tags is by isolating them from any kind of electromagnetic waves. This can be achieved by making what is known as a Faraday Cage (FC), a container made of metal mesh or foil that is impenetrable by radio signals (of certain frequencies). There are currently a number of companies that sell this type of solution [10].

3.1.3 Active Jamming:

Another way of obtaining isolation from electromagnetic waves, and an alternative to the FC approach, is by disturbing the radio channel, a method which is commonly known as active jamming of RF signals. This disturbance can be achieved with a device that actively broadcasts radio signals, so as to completely disrupt the radio channel, thus preventing the normal operation of RFID readers.

3.1.4 Blocker Tag:

If more than one tag answers a query sent by a reader, it detects a collision. The most, important singulation protocols are ALOHA (13.56 MHz) and the tree walking protocol (915 MHz). Juels [9] used this feature to propose a passive jamming approach based on the tree-walking singulation protocol, called blocker tag. A blocker tag simulates the full spectrum of possible serial numbers for tags.

3.2 Bill of Rights

In [16], Garfinkel proposed a so-called RFID bill of Rights, which adapts the principles of fair information practices to RFID systems deployment. This bill of rights consists of five guiding principles for RFID system creation and deployment. Users of RFID systems and purchasers of products containing RFID tags have:

- The right to know if a product contains an RFID tag.
- The right to have embedded RFID tags removed, deactivated, or destroyed when a product is purchased.

- The right to first-class RFID alternatives. Consumers should not lose other rights (such as the right to return a product or travel on a particular road) if they decide to opt-out of RFID or exercise an RFID tag's kill feature.
- The right to know what information is stored inside their RFID tags. If this information is incorrect, there must be means to correct or amend it.
- The right to know when, where and why an RFID tag is being read.

3.3 Security Protection Approaches

In this section we will discuss the approaches proposed to protect the security of the system.

3.3.1 Non-Cryptographic Primitives:

There are some solutions which do not use true cryptographic operations. These are purely based on primitive operations.

3.3.1.1 Key Permutation: Vajda *et al.* [3] proposed several lightweight authentication protocols for authenticating RFID tags to readers.

It's a challenge-response protocol (called Protocol 1), in which the tag and reader share a secret key, $k^{(0)}$. The reader randomly selects a uniform bitstring x to construct a challenge. The reader transmits $a^{(i)} = x^{(i)} \oplus k^{(i)}$ to the tag, where i is the i^{th} transaction between the reader and tag. $k^{(i)}$ is calculated by a permutation of $k^{(0)}$. Since the bitstring is selected randomly so the information passed, $a^{(i)}$, to the tag is random too. The tag uses its knowledge of $k^{(i)}$, to extract $x^{(i)}$ as follows:

$$a^{(i)} \oplus k^{(i)} \oplus k^{(0)} = x^{(i)} \oplus k^{(i)} \oplus k^{(i)} \oplus k^{(0)}$$

$= x^{(i)} \oplus k^{(0)}$

The tag then responds to the reader with $b^{(i)} = x^{(i)} \oplus k^{(0)}$. The reader verifies the correctness of the tag's response since it knows $x^{(i)}$ and $k^{(0)}$.

The protocol is considered broken when an adversary can send a valid $b^{(i)} = x^{(i)} \oplus k^{(0)}$ or learn the value of $k^{(0)}$ as seen in figure 3.1 below.

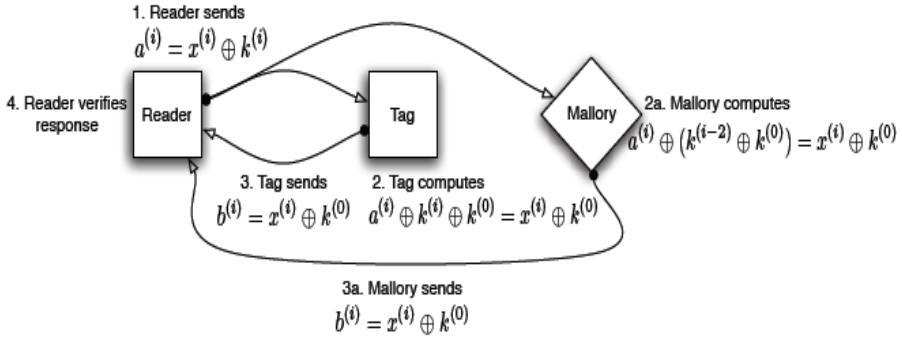

Figure 3.1: Steps 1 – 4 of VB Protocol 1 [13]

3.3.2 Hash Functions Schemes:

3.3.2.1 Hash-Lock Scheme: Weis *et al.* [2] proposed a simple hash-based protocol which enables to implement security at low cost.

In hash-lock scheme, a back-end server stores both keys *k* and metaID's as pairs in its database for all tags, where each tag has metaID = h(k) for its key. When a reader queries a tag, the tag transmits metaID to the reader as response. The reader sends metaID to the back-end server. Back-end server looks up the appropriate metaID and sends the corresponding key pair to the reader, which is transmitted to the tag. The tag hashes the key and compares it with the stored metaID. If those two values are matched, the tag sends its own ID to the reader.

This scheme requires implementing a hash function on the tag and managing keys on the back-end. Hash-lock scheme uses metaID as the unique ID of each tag for every read attempt. Thus, the data privacy of tag bearers is protected and the protocol can meet confidentiality. However, metaID is always constant so that attackers can eavesdrop it, identify each tag, and trace the tag. Therefore, location privacy of tag bearers is compromised.

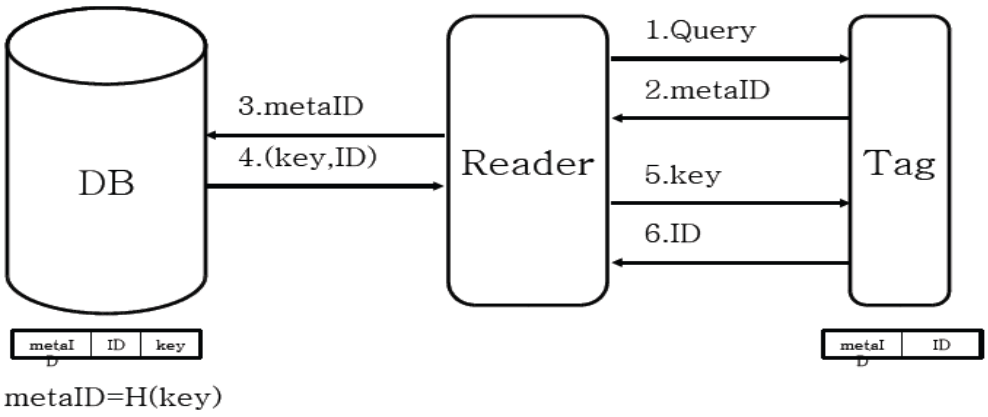

Figure 3.2: Hash-Locking: A reader unlocks a hash-locked tag [2].

Although this scheme is simple, it does not provide mutual authentication, suffers from the tracking problem, uses a hash function in the tag, the key is sent in plain text so forgery is possible, spoofing can be done, and is not forward secure [11].

3.3.2.2 Extended Hash-Lock Scheme: In extended hash-lock scheme [2], they proposed another method to overcome the tracing problem.

This is an extension of the hash lock type scheme. It requires the tag to have a hash function and a pseudo-random generator. The tag picks random number R uniformly and calculates c = hash ($ID_k||R$) as the tag's unique identification for every session. The

tag then sends *c* and *R* to the reader. The reader sends the data to the back-end database. The back-end server calculates the hash function for each ID stored in the database using the input as the received *R* and ID_k of each tag. The back-end server then identifies the ID_k that is related to the received *c* and sends the ID_k to the reader.

The tag output changes with each access, so this scheme deters tracking. This scheme is also strong for the replay attack. However, the tag can be traced if the tag's ID is exposed. In addition, an adversary can query a tag to get a tag's valid message pair (*c, R*). Later on, the attacker can impersonate that tag to legitimate reader. The response from the reader will identify the tag. Also, the implementation issue for the random number generator is still an issue.

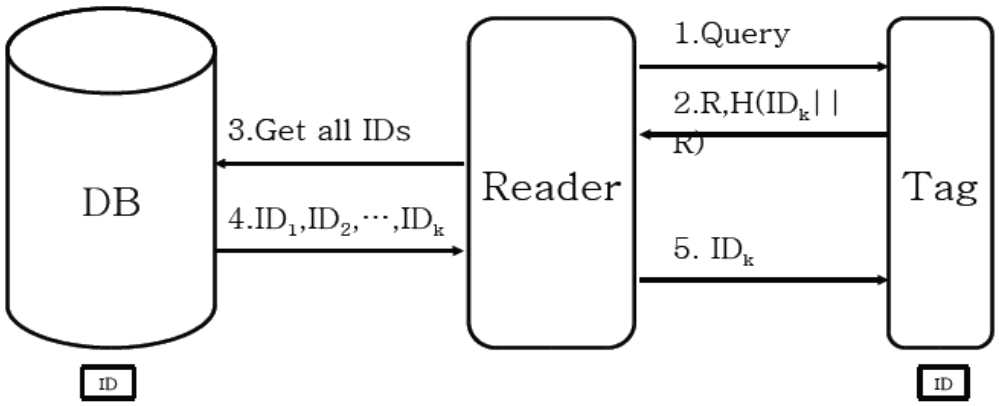

Figure 3.3: Randomized Hash-Locking: A reader unlocks a tag whose ID is *k* in the randomize hash-lock scheme [2].

Although this scheme deters tracking, it suffers from a high time complexity for tag identification. It also uses a hash function on the tag, the ID is sent in plain text so forgery is possible, does not provide mutual authentication and is not forward secure.

3.3.2.3 Hash-based Varying Identifier: This scheme was proposed by Henrici and Muller [12]. This scheme also adopts a hash function and a random number generator, but a pseudo random number is generated by a back-end server and transmitted to the tag for every interrogation to make the tag's queried identifier random and preserve location privacy.

In this protocol, RFID-tag needs to contain fields for the following entries:

DB-ID	ID	TID	LST

The Back-End database needs to contain a table with the following entries for each record:

HID (Hash of current ID)	ID (Current ID)	TID (Last Trans Number)	LST (Last Successful Tran Numb)	AE (Assoc DB)	DATA (A ref to Tag data)

Step 1: Reader sends query to a tag.

Step 2: Tag increases its transaction number (TID) by one and sends the h (ID), h (TID $\oplus$ ID), and ΔTID = TID – LST back to the reader.

Step 3: Reader sends this information back to the backend database indicated in DB-ID field.

Step 4: In the backend database, record with HID = h (ID) is selected. Calculate TID^* = LST + ΔTID. If h ($TID^* \oplus$ ID) matches h (TID $\oplus$ ID) and TID^* > TID, then the message is valid. A random number RND is generated. With this RND, a new ID is generated performing ID^* = RND $\oplus$ ID and HID = h (ID^*) and is stored in the new record row. The AE-field is updated in both rows so that they can reference to each other. The TID^* is stored in the TID field and in the LST field of the new row.

Step 5: Now a reply message containing RND and a hash h (RND ⊕ TID* ⊕ ID) is created and send to the reader which forwards the message to the tag.

Step 6: Tag verifies by calculating h (RND ⊕ TID ⊕ ID) and if it's same as sent by the database, then it updates its stored ID to the value RND ⊕ ID and sets its LST to the TID value.

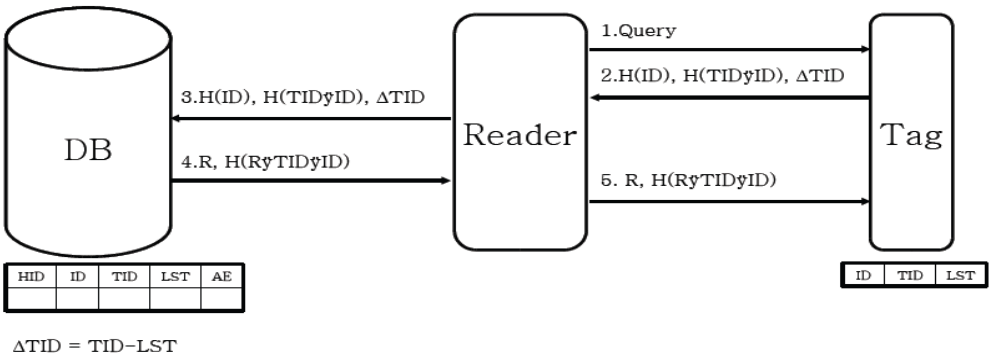

Figure 3.4: Hash-based Enhancement Using Varying Identifiers [12].

This protocol can be compromised including an attack based on the non-randomness of transmitted information, refreshment avoidance, and database desynchronization. It uses a hash function on the tag which increases the cost. This scheme does provide mutual authentication, but suffers from tracking and the desynchronization problem [11].

3.3.2.4 Hash Chain-based Scheme: Another authentication protocol was proposed by Okubo *et al.* [1] based on hash-chains, which renew the secret information contained in the tag, protects the user's location privacy and anonymity. When a tag is requested by a reader, it sends a hash of its current identifier and then renews it using a second hash function. Initially tag has initial information s_1. In the i-th transaction with the reader, the

RFID tag sends $a_i = G(s_i)$ to the reader and renews the secret $s_{i+1} = H(s_i)$ as determined from the previous secret s_i. Where H and G are hash functions. The reader sends a_i to the back-end server. The back-end server database maintains a list of pairs (ID, s_1), where s_1 is the initial secret information and is different for each tag. Then the back-end database calculates $a_i' = G(H^i(s_1))$ for each s_1 in the list, and checks if $a_i = a_i'$. If it finds, it returns the ID, which is a pair of a_i'.

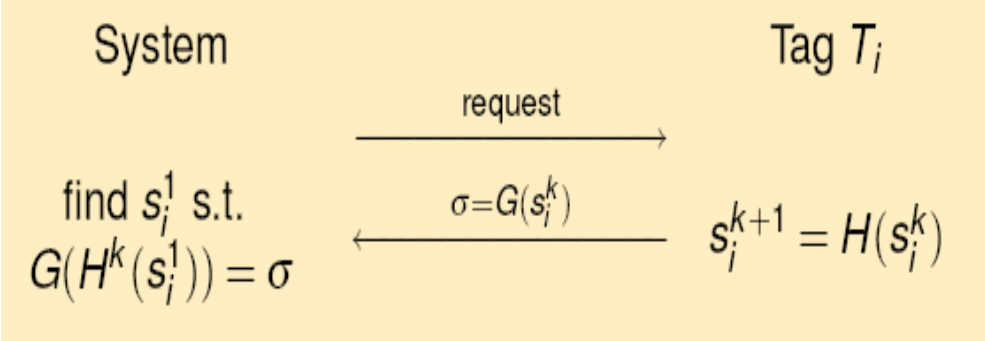

Figure 3.5 Hash Chain Scheme [1]

This protocol does not provide mutual authentication, uses two hash functions on the tag side and suffers from the scalability problem.

3.3.2.5 Mutual Authentication Scheme based on Synchronized Secret: Lee *et al.* [21] proposed a mutual authentication scheme based on primitive operations and hash functions.

The secret key (k) is shared between the tag and the back-end server. Database at back-end server has fields IDR, K, and K_{last}, which saves the ID, the current *k*, the preceding *k* (the previous secret information which is replaced by the current *k*), respectively.

Step 1: Reader generates and saves a new pseudorandom number *s* and sends to it tag.

Step 2: Tag generates a new pseudorandom number r_1 and sends to reader. Then it calculates $r_2 = h\,(r_1 \oplus k \oplus s)$, and sends it to the reader.

Step 3: Reader sends r_1, r_2 and s back to the back-end server.

Step 4: Back-end server searches for k' from the fields K and K_{last} of the table, which satisfies the following equation: $h\,(r_1 \oplus k' \oplus s) \stackrel{?}{=} r_2$.

Step 5: If k' is found in the field K of record, then $K_{last} = k'$ and $K = h\,(k')$. If k' is found in the field K_{last} of record, nothing is done.

Step 6: Back-end server now calculates $r_3' = h\,(r_2 \oplus k' \oplus s)$, and sends to the reader. Reader transfers r_3' to the tag.

Step 7: Tag test the following equation: $r_3' \stackrel{?}{=} r_3$. If it comes out true then $k = h(k)$.

This protocol uses hash function and a pseudo-random number generator (PRNG) on tag which greatly increases the price of tag. This protocol suffers from the scalability problem too.

From the above discussed protocols, we can see that all of them use either a pseudo-random number generator (PRNG) or the hash function on the tag or they use both making the tag expensive. Hence, we need another approach to reduce the cost by making use of simple operations and less expensive alternatives so that this technology can be widely used.

CHAPTER 4

PROBLEM STATEMENT

The reason that we cannot use well-known authentication protocols comes from the fact that such protocols do not preserve the privacy of the tag. In other words, the reader can check whether or not the identity claimed by the tag is true, but he cannot guess it himself: the tag must send his identity which in turn allows an adversary to track him.

Asymmetric cryptography could easily solve this problem: the tag encrypts his identity with the public key of the reader. Thus, no eavesdropper is able to identify the tag. Unfortunately, asymmetric cryptography is too heavy to be implemented within a tag.

In symmetric cryptography, the problem remains that both tag and reader need to share a common secret-key instead of a public-key. In RFID systems, tags are not tamper-resistant. Therefore an attacker who tampers with a tag can track its past events, if the person had access to its previous interactions with the reader, e.g., from the readers' log files. This is possible in those cases in which the tagged item was used for temporary purposes for sometime and then returned back. Using a common key for all the tags would be weak from a security standpoint: an attacker who

tampers with one tag, e.g., her own tag, would also be able to attack all the other tags in the system. Another approach consists of using a unique key for each tag, such that only the reader knows all these keys. However, this approach suffers from an expensive time complexity on the readers' side. Indeed, because only symmetric cryptography functions can be used, the system needs to explore its entire database in order to retrieve the identity of the tag it queries. If n is the number of tags managed by the system, O (n) cryptographic operations are required in order to identify one tag. The advantage of the system over an attacker is that the system knows in which subset of identifiers it needs to search while the attacker has to explore the full range of identifiers.

One of the major problems faced by current protocols today is the high cost of tags that makes them unusable for item level tagging. This is because using cryptographic operations on tags needs more computational power and memory. The goal of this thesis is to present a suite of ultra-light protocols that can be used for item level tagging. Security and performance analysis of the proposed protocol are done to see if it holds to the major privacy and security issues. Finally the proposed approach is applied to a supply chain system.

Instead of using cryptographic operation like hash function or block ciphers on tags which require expensive tags, we will make use of basic bitwise operations, pseudo-random number generator (PRNG) and pseudo-random function (PRF) to achieve the same level of privacy and security. Hence cheap tags can be used. Table 3 shows the number of logical gates needed for implementing various hash functions and AES encryption.

Table 3: Core Comparison [23, 24]

SOLUTIONS	IMPLEMENTATION	GATE COUNTING
HASH	MD5 Helion [23]	16K Gates
	Fast SHA-1 Helion [23]	20K – 23K Gates
	Fast SHA-256 Helion [23]	23K – 26K Gates
AES Unit	JetAES Tiny [24]	4370 Gates
	Feldhofer [25]	3595 Gates
	JetAES Standard [24]	8970 Gates
PRNG	TRNG Certicom [30]	22K Gates
PRF	SSG [31]	1435 Gates

The second problem is the high time complexity of tag identification which makes is unusable for high end systems and gives rise to the scalability problem. This problem can be solved by issuing each tag with a tag index number that is stored in the database as well. Instead of going through the entire database, each tag will be identified with its tag index number. However this tag index number will be updated on each successful mutual authentication which will make the tag untraceable.

The third problem which is present in some protocols is the desynchronization of databases, which will lead to the problem of denial of service. The desynchronization problem happens when either the reader updates its values and the tag does not and vice versa. In case, if the tag updates its values, then that tag becomes non-functional. However if the reader updates its values and not the tag, this problem can be solved by storing the previous secret key in the database.

CHAPTER 5

PROPSED METHODOLOGY

In this thesis, we propose five protocols. The first is an ultra-light mutual authentication protocol between RFID readers and tags based on PRN and primitive operations, the second and third protocols are variants of first one, the fourth is a light mutual authentication protocol based on PRF and master keys, the fifth is similar to fourth one with a difference of using only one master key for all the tags instead of using a master key for each tag. We propose multiple protocols as they provide different levels of security and costs for implementation, thus making it easier to choose the right protocol for the right environment. The motivation behind protocol 1 is to provide the cheapest solution without using a random number generator on the tag side. The motivation behind protocol 2 is to provide more security compared to the first one, but this comes with the cost of a using random number generator on the tag side. The motivation behind protocol 3 is to achieve the same level of security as protocol 2 without the use of the random number generator on the tag side. The motivation for protocol 4 is to provide more security compared to the first, second and third protocols, but this comes with the cost of using PRF on both sides. The motivation for protocol 5 is to provide the same level of security as protocol 4 without the use of a master key

for each tag. In protocol 5, we make use of one master key for the whole system, instead of using a master key for each tag.

5.1 Protocol 1

5.1.1 Assumptions:

In the first protocol, all the costly computing operations are done by the reader. We assume that readers are devices with enough computing power to generate random numbers and to perform any cryptographic operations. Communication must be initiated by readers due to the fact that low-cost tags are passive. We consider that the communication channel between the reader and the back-end database is secure. Therefore we consider both reader and back-end database as one entity.

All tags are supplied with tag-index number (TIN) which is the index of the table (a row) where all the information about the tag is stored in the database. Each tag has an associated key which is divided in two parts (K = K1|| K2). Tag identification number (ID) which holds the information about the product to which it's attached is also stored permanently in it. K1 and K2 values changes during authentication.

the Tag stores the following data in it.

TIN	K1	K2	ID

The Back-end database stores the following data for each tag in the database.

TIN	K1	K2	$K1_{last}$	$K2_{last}$	ID

5.1.2 Authentication:

In this protocol the reader generates the random number by making use of PRNG. We describe the process of our authentication as follows:

Step 1: Reader generates a random number *n* utilizing PRNG, and sends it to a tag.

Step 2: Tag will create three messages A, B and C as follows and sends them back to reader:

$$A = K1 \oplus n, \qquad B = K2 \oplus n, \qquad C = TIN \oplus n$$

Step 3: *Tag Identification:* From message C, as reader already knows the random number *n,* reader will get the tag index number (TIN) for that particular tag as follow:

$$C \oplus n \Rightarrow TIN \oplus n \oplus n \Rightarrow TIN$$

Step 4: *Tag Authentication:* Using this TIN, reader will look up in the database to find the record for that particular tag. Making use of values of K1, K2, $K1_{last}$, $K2_{last}$ in the database for that particular tag we will authenticate the tag. We got two cases here for authentication:

Case 1:Both values stored in K1 and K2 can be used as follows:

$$A \oplus K1 \Rightarrow K1 \oplus n \oplus K1 \Rightarrow n, \qquad B \oplus K2 \Rightarrow K2 \oplus n \oplus K2 \Rightarrow n$$

If the output from both messages A and B after XOR ($\oplus$) with K1 and K2 respectively yields *n*, which is known to the reader, it authenticates that the message came from a valid tag. This means that the tag exists in the database and it's a new authentication process.

Case 2: Or values stored in $K1_{last}$ and $K2_{last}$ can be used as follows:

$$A \oplus K1_{last} \Rightarrow K1 \oplus n \oplus K1_{last} \Rightarrow n$$

$$B \oplus K2_{last} \Rightarrow K2 \oplus n \oplus K2_{last} \Rightarrow n$$

If the output from both messages A and B after XOR ($\oplus$) with $K1_{last}$ and $K2_{last}$ respectively yields *n*, this means that the tag exists in the database, however, back-end database already updated its K1 and K2 values at the previous authentication process but the tag didn't.

Step 5: *Update Reader:* Back-end database updates information of tag.

In case 1, where K1 and K2 values are used to authenticate the tag, values are updated as follows:

$$K1_{last} = K1, \qquad K2_{last} = K2$$

$$K1' = K1 \oplus n, \qquad K2' = K2 \oplus n$$

These values of K1′ and K2′ are stored in K1 and K2 respectively in the database.

In case 2, where $K1_{last}$ and $K2_{last}$ values are used, we do not update the values in the database. This means that the tag is trying to use the previous authentication values showing either tag didn't update its values or some adversary is trying to hack the system.

Step 6: Reader generates a new message D as follows and sends it to tag:

$$D = TIN \oplus K1 \oplus K2$$

Step 7: *Reader Authentication:* As K1 and K2 are known to tag, it will use those values to get TIN from message D. If that TIN is same as TIN stored in tag, it validates that the message came from a legitimate reader thus giving us mutual authentication as follows:

$$D \oplus K1 \oplus K2 \Rightarrow TIN \oplus K1 \oplus K2 \oplus K1 \oplus K2 \Rightarrow TIN$$

Step 8: *Update Tag:* After mutual authentication, tag also updates its values of K1 and K2 as follows:

$$K1' = K1 \oplus n, \qquad K2' = K2 \oplus n$$

These values of K1′ and K2′ are stored in K1 and K2 respectively in the tag.

Step 9: Tag will create a message E in which it will send its ID to the reader, releasing its information about the product as follows:

$$E = TIN \oplus ID \oplus n$$

Modifications to the above described protocol can be achieved by making use of other primitive bitwise operations like AND ($\wedge$), and OR ($\vee$) to make it more complex.

The process of authentication is shown in figure 5.1 on the next page.

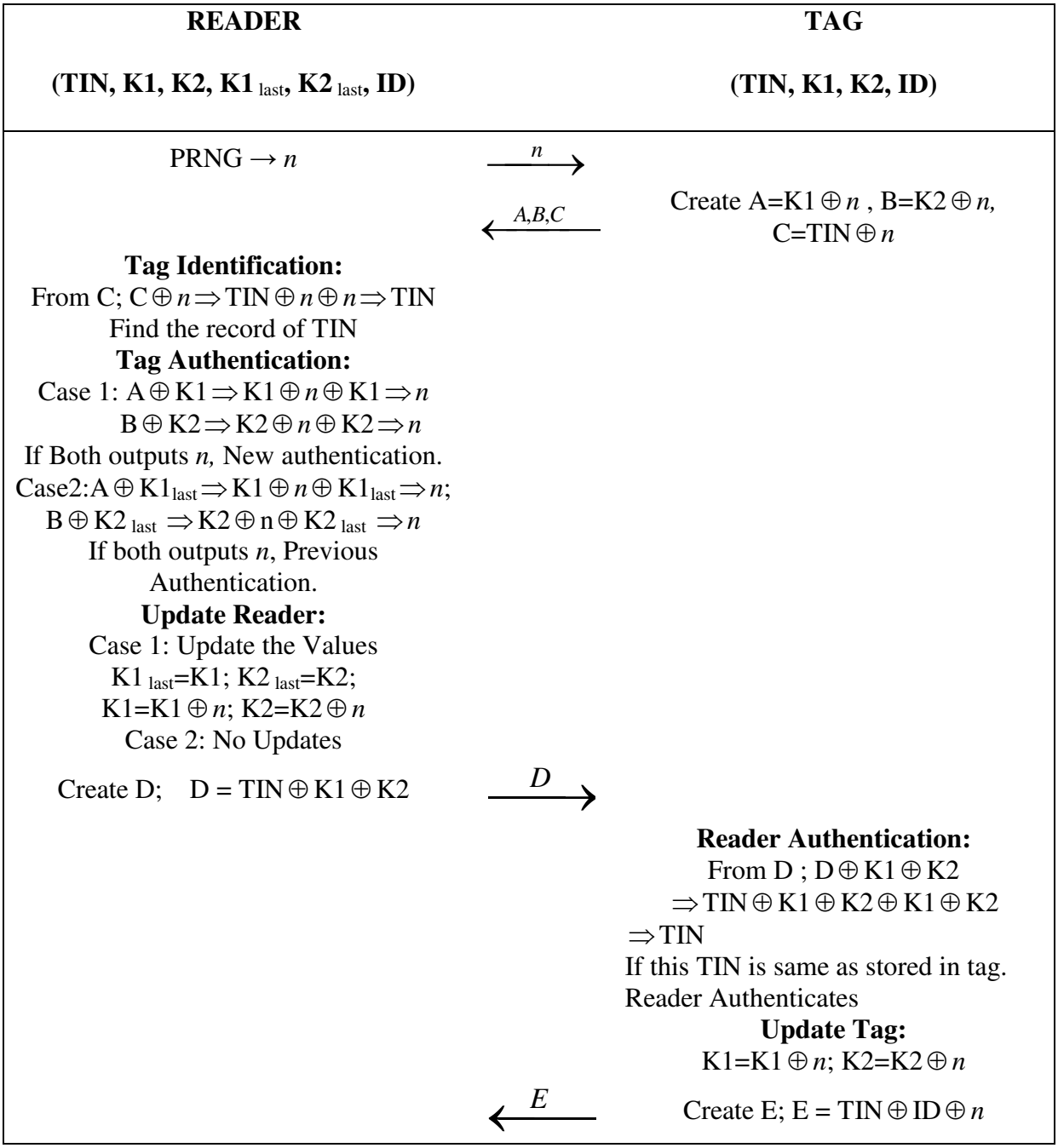

Figure 5.1: Protocol 1 (PRNG on Reader Side)

5.2 Protocol 2

5.2.1 Assumptions:

This is a variant of the first protocol with a small difference. In this protocol, we make use of PRNG on both reader and tag side. All other assumptions made in protocol 1 apply here also.

5.2.2 Authentication:

We describe the process of authentication as follows:

Step 1: Reader generates a random number *n1* utilizing PRNG, and sends it to a tag.

Step 2: Tag generates a random number *n2,* creates three messages A, B and C; and then sends them back to reader as follows:

$$A = K1 \oplus n2, \qquad B = K2 \oplus n2, \qquad C = TIN \oplus n1$$

Step 3: *Tag Identification:* From message C, as reader already knows the random number *n1,* reader will get the tag index number (TIN) for that particular tag as follows:

$$C \oplus n1 \Rightarrow TIN \oplus n1 \oplus n1 \Rightarrow TIN$$

Step 4: *Tag Authentication:* Using this TIN, reader will look up in the database to find the record for that particular tag. Making use of values of K1, K2, $K1_{last}$, $K2_{last}$ in the database for that particular tag we will authenticate the tag. We get two cases here for authentication:

Case 1:Both values stored in K1 and K2 of record can be used as follows:

$$A \oplus K1 \Rightarrow K1 \oplus n2 \oplus K1 \Rightarrow n2, \qquad B \oplus K2 \Rightarrow K2 \oplus n2 \oplus K2 \Rightarrow n2$$

If the output from both messages A and B after XOR ($\oplus$) with K1 and K2 respectively yields *n2*, it authenticates that the message came from a valid tag. This means that the tag exists in the database and it's a new authentication process.

Case 2: Or values stored in $K1_{last}$ and $K2_{last}$ of record can be used as follows:

$$A \oplus K1_{last} \Rightarrow K1 \oplus n2 \oplus K1_{last} \Rightarrow n2,$$

$$B \oplus K2_{last} \Rightarrow K2 \oplus n2 \oplus K2_{last} \Rightarrow n2$$

If the output from both messages A and B after XOR ($\oplus$) with $K1_{last}$ and $K2_{last}$ respectively yields *n2*, this means that the tag exists in the database, however, the back-end database had already updated its K1 and K2 values at the previous authentication process but the tag didn't.

Step 5: *Update Reader:* Back-end database updates information about the tag as follows:

In case 1, where K1 and K2 values are used to authenticate the tag, values are updated as follows:

$$K1_{last} = K1, \qquad K2_{last} = K2$$

$$K1' = K1 \oplus n2, \qquad K2' = K2 \oplus n2$$

These values of $K1'$ and $K2'$ are stored in K1 and K2 respectively in the database.

In case 2, where $K1_{last}$ and $K2_{last}$ values are used, we do not update the values in the database. This means that the tag is trying to use the previous authentication values showing either tag didn't update its values or some adversary is trying to hack the system.

Step 6: Reader will create a new message D as follows and sends it to tag:

$$D = TIN \oplus K1 \oplus K2 \oplus n2$$

Step 7: *Reader Authentication:* As *n2*, K1 and K2 are known to tag, it will use those values to get TIN from message D. If this TIN is same as the one stored in tag, it will

validate that the message came from a legitimate reader thus giving us mutual authentication as follows:

$$D \oplus K1 \oplus K2 \oplus n2 \Rightarrow TIN \oplus K1 \oplus K2 \oplus n2 \oplus K1 \oplus K2 \oplus n2 \Rightarrow TIN$$

Step 8: *Update Tag:* After mutual authentication, the tag also updates its values of K1 and K2 as follows:

$$K1' = K1 \oplus n2, \qquad K2' = K2 \oplus n2$$

These values of K1′ and K2′ are stored in K1 and K2 respectively in the tag.

Step 9: Tag will create a message E in which it will send its ID to the reader, releasing its information about the product as follows:

$$E = TIN \oplus ID \oplus n1 \oplus n2$$

The process of authentication is shown in figure 5.2 on the next page.

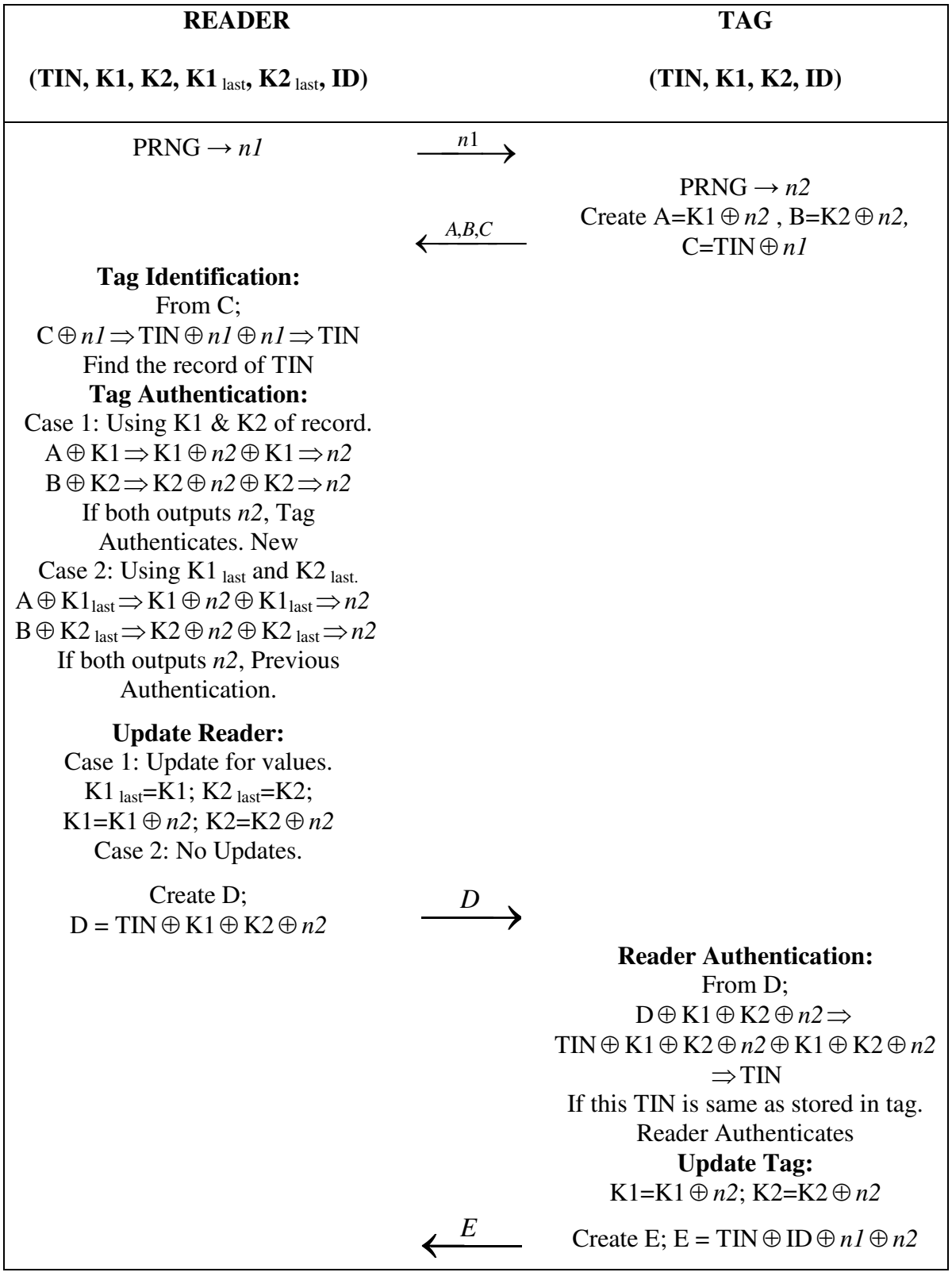

Figure 5.2: Protocol 2 (PRNG on Reader and Tag Side)

5.3 Protocol 3

5.3.1 Assumptions:

This protocol uses PRNG on the reader side and updating is done after mutual authentication. Only one previous key value is stored rather than two in this protocol. All other assumptions made in protocol 1 apply here.

5.3.2 Authentication:

We describe the process of authentication as follows:

Step 1: Reader sends a message "hello" to the tag.

Step 2: Tag sends back its tag index number (TIN) in response.

Step 3: *Tag Identification:* Reader looks up the database and finds the record for that TIN. Then it creates a random number *n* from PRNG. It creates messages A, B and sends them to tag.

$$A = K1 \oplus n, \qquad B = K2 \oplus n$$

Step 4: *Reader Authentication:* From messages A and B, tag will use the stored values of K1 and K2 to get the random number *n* as follows:

$$A \oplus K1 = K1 \oplus n \oplus K1 \Rightarrow n, \qquad B \oplus K2 = K2 \oplus n \oplus K2 \Rightarrow n$$

If both *n* are the same, then messages A and B came from an authentic reader.

Step 5: Tag will create message C, D and send them to the reader as below:

$$C = K1 \oplus n, \qquad D = ID \oplus n$$

Message D will contain the ID of the product to which that tag is attached, which can be retrieved by the reader easily.

Step 6: *Tag Authentication:* Making use of values of K1 and $K1_{last}$ in the database for that particular tag we will authenticate the tag. We can have two cases for tag authentication.

Case 1: Value stored in K1 of record is used to authenticate as below:

$$C \oplus K1 \Rightarrow K1 \oplus n \oplus K1 \Rightarrow n$$

If *n* is the same as that generated by the tag, it authenticates that the message came from a valid tag. This means that the tag exists in the database and it's a new authentication process.

Case 2: OR value stored in $K1_{last}$ of record is used to authenticate as below:

$$C \oplus K1_{last} \Rightarrow K1 \oplus n \oplus K1_{last} \Rightarrow n$$

If *n* is the same as that generated by the tag, this means that the tag exists in the database, however, the back-end database has already updated its K1 value at the previous authentication process but the tag did not.

Step 7: *Update Reader:* The Back-end database will update information about the tag as follows:

In case 1, where K1 was used to authenticate the tag, values will be updated as follows:

$$K1_{last} = K1, \qquad K1' = K1 \oplus n, \qquad K2' = K2 \oplus n$$

These values of K1′ and K2′ are stored in K1 and K2 respectively in the database.

In case 2, where the $K1_{last}$ value is used, we do not update the values in the database. This means a tag is trying to use the previous authentication values showing either the tag didn't' update or some adversary is trying to hack the system.

Step 8: Reader will create message F as follows and sends it to tag:

$$F = TIN \oplus K1 \oplus K2 \oplus n$$

Step 9: *Reader Authentication:* As K1 and K2 are known to the tag; it will use those values to get TIN from message F. If this TIN is the same as the one stored in the tag, it will validate that the message came from a legitimate reader.

$$F \oplus K1 \oplus K2 \oplus n \Rightarrow TIN \oplus K1 \oplus K2 \oplus n \oplus K1 \oplus K2 \oplus n \Rightarrow TIN$$

This protocol has an advantage in that it authenticates the reader twice.

Step 10: *Tag Update:* and tag will update its values too as follows:

$$K1' = K1 \oplus n, \qquad K2' = K2 \oplus n$$

These values of K1′ and K2′ are stored in K1 and K2 respectively in the tag.

The process of the authentication is shown in figure 5.3 on the next page.

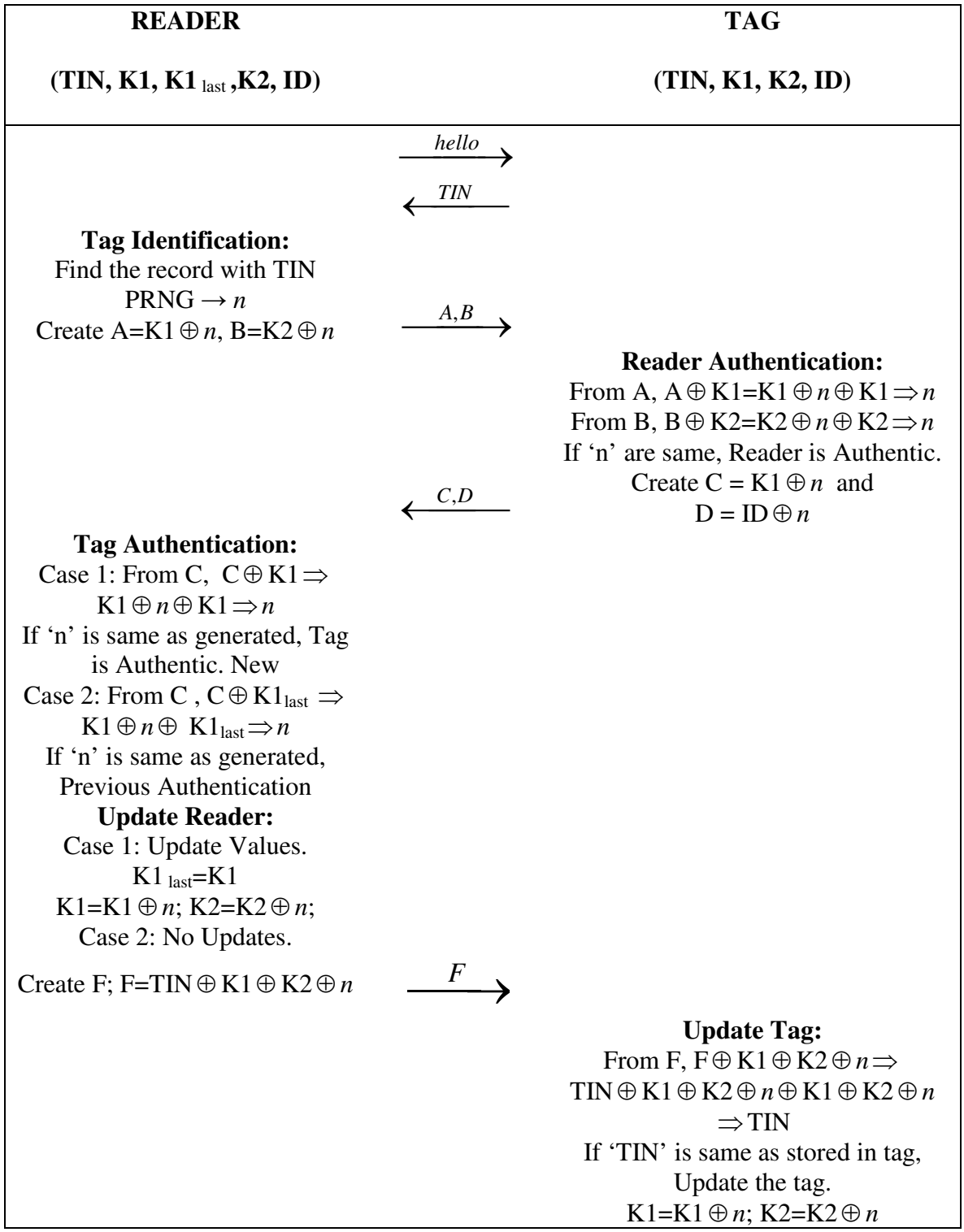

Figure 5.3: Protocol 3 (PRNG on Reader Side and Updating after Authentication)

5.4 Protocol 4

5.4.1 Assumptions:

All tags are supplied with a Master Key (K) and ID. The master Key (K) generates the seed (k) for all PRF as shown in the figure below. The PRF has two inputs, one is a secret seed (k) and other is a variable x. We can create different values from a PRF by chaining either k or x. When we use variable value as 1, we use it to update the master key. When we use variable values 2 and 3, we use it as pseudo-random numbers.

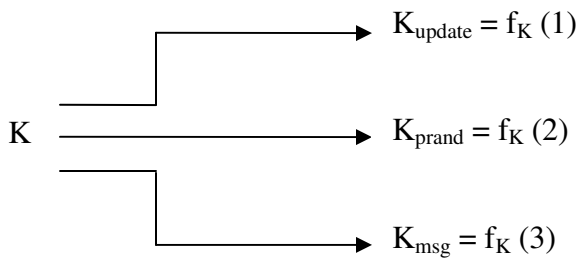

Figure 5.4 Pseudo Random Seed production from Master Key

The Tag stores the following data.

K	ID

The Back-end database stores the following data for each tag in the database.

K	K_{last}	ID

5.4.2 Authentication:

We describe the process of authentication as follows:

Step 1: Reader sends a message “hello” to tag.

Step 2: Tag generates pseudorandom numbers *n1* and *n2* using PRF with master Key as one of the inputs. Then it creates message α using PRF with *n1* and *n2* as input as shown in figure 5.5 and sends this message to reader as follows:

$$n1 = f_k(x) \Rightarrow f_k(3) = F(K, 3)$$

$$n2 = f_k(x) \Rightarrow f_k(2) = F(K, 2)$$

$$\alpha = F(n1, n2)$$

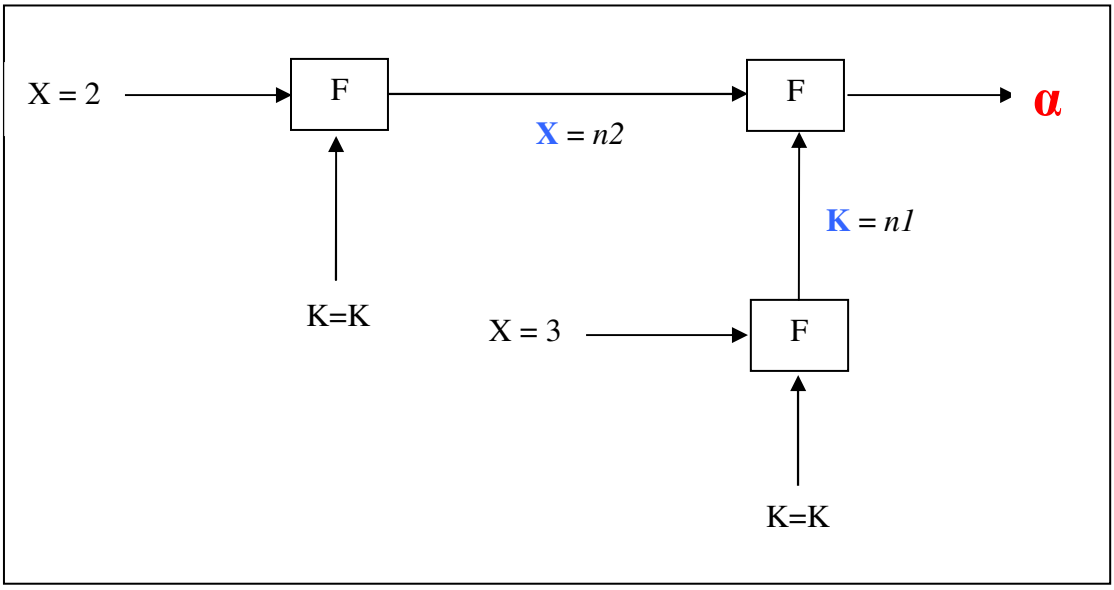

Figure 5.5: Creating Message α from PRF

Step 3: *Tag Authentication:* Reader will use the values of K and K_{last} in the database to carry out an exhaustive search to find that tag whose response is the same as the message received. We have two cases for tag authentication.

Case 1: K is used to authenticate the tag as below:

$$F(f_k(3), f_k(2)) \stackrel{?}{=} \alpha$$

Reader will apply the PRF on the stored value of master key along with variables and compare it with the message received. If the PRF value comes out to be the same, that means that tag exits in the database and it's a new authentication.

Case 2: OR K_{last} is used to authenticate the tag as below:

$$F(f_{klast}(3), f_{klast}(2)) \stackrel{?}{=} \alpha$$

This means that the back-end database already updated its K value at the previous authentication but the tag did not. This will catch the replay attack and will prevent the attacker from desynchronizing the database.

Step 4: *Update Reader:* The Back-end database will update information about the tag as follows:

In case 1, where K was used to authenticate the tag, values will be updated as follows:

$$K_{last} = K; \qquad K = f_k(x) \Rightarrow f_k(1) = F(K, 1)$$

In case 2, where K_{last} was used, we do not update the values in the database. This means a tag is trying to use the previous authentication value showing either the tag did not update last time or some adversary is trying to hack the system.

Step 5: Reader generates pseudo-random numbers *n3* and *n4* using PRF and newly generated master key K as one of the inputs. Then it creates message β using PRF as shown in figure 5.6 with *n3* and *n4* as input and sends this message to the tag as follows:

$$n3 = f_k(x) \Rightarrow f_k(3) = F(K, 3)$$

$$n4 = f_k(x) \Rightarrow f_k(2) = F(K, 2)$$

$$\beta = F(n3, n4)$$

Step 6: *Reader Authentication:* The Tag will compute the new value for the master key and will use this value to check if the message is the same as that sent by the reader as shown below:

$$K' = f_k(x) \Rightarrow f_k(1) = F(K, 1)$$

$$F\left(f_{k'}(3), f_{k'}(2)\right) \stackrel{?}{=} \beta$$

If the value comes out the same as the message received, this means the reader is legitimate, thus giving mutual authentication.

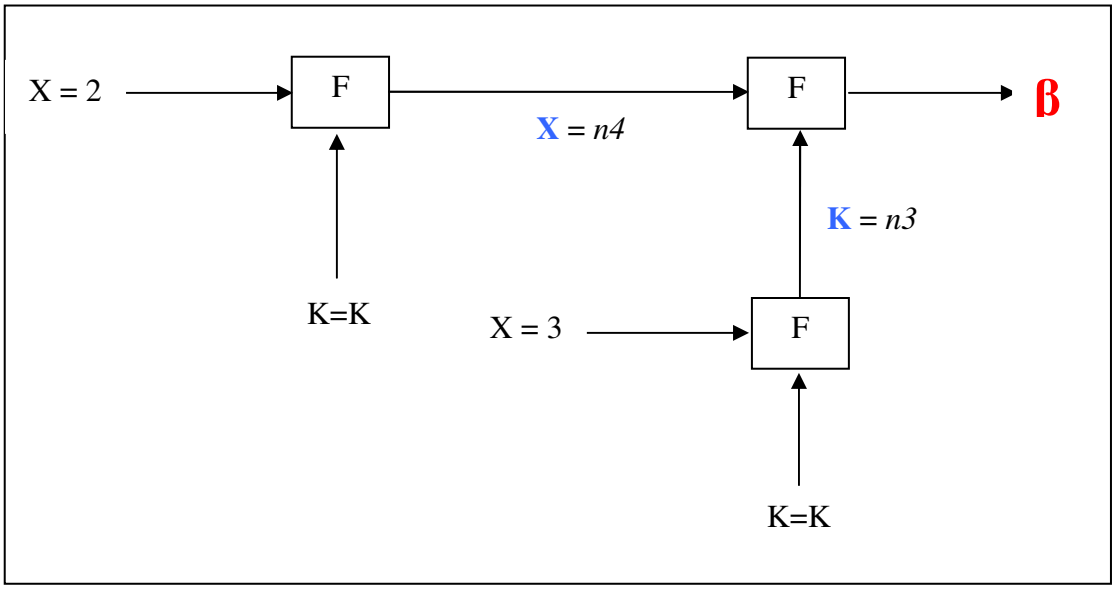

Figure 5.6: Creating Message β from PRF

Step 7: *Update Tag:* Tag will update its values as below:

$$K = f_k(x) \Rightarrow f_k(1) = F(K, 1)$$

Step 8: Tag will create message γ using the values of *n1, n2* and ID as below:

$$\gamma = ID \oplus n1 \oplus n2$$

The process of authentication is shown in figure 5.7 on the next page.

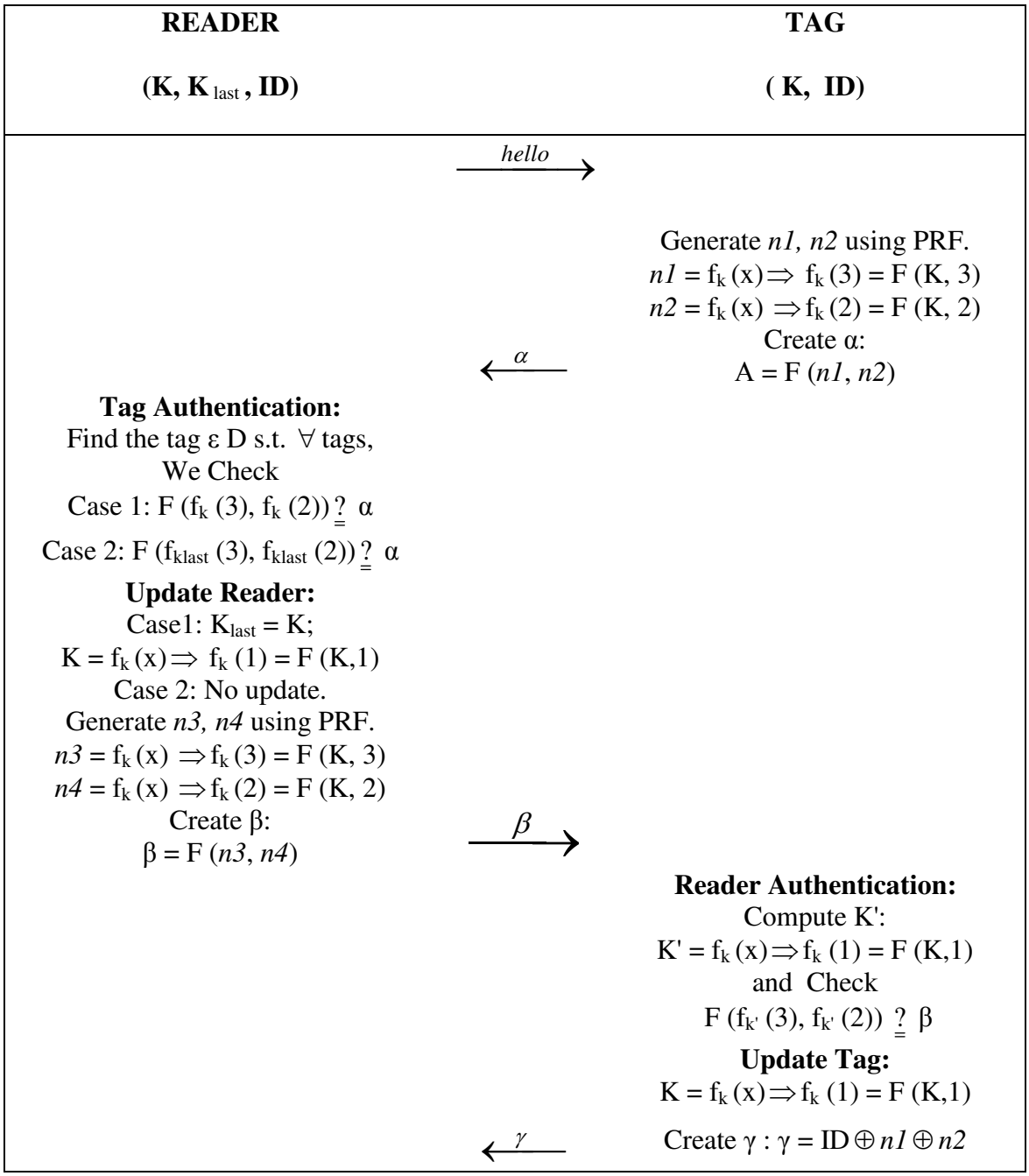

Figure 5.7: Protocol 4 (Using Tag Master Key)

5.5 Protocol 5

5.5.1 Assumptions:

In this protocol, instead of using one master key for each tag, we use one master key for the whole system. This protocol is similar to the last one. Each tag is supplied with its tag number (TN), counter value (C), master key (K) and product ID.

A Tag has to store the following data.

K	TN	C	ID

The Back-end database has to store the following data for each tag in the database, where C_{last} is the value of C used last time.

K	TN	C	C_{last}	ID

5.5.2 Authentication:

We describe the process of authentication as follows:

Step 1: Reader sends a message “hello” to tag.

Step 2: Tag generates pseudorandom numbers *n1* using PRF with tag number TN as X and master Key as K in the input. Then it creates a message α as shown in figure 5.8 using PRF with *n1,* previously generated, and C as input and sends this message to the reader as follows:

$$n1 = f_k(x) \Rightarrow f_k(TN) = F(K, TN)$$

$$\alpha = F(n1, C)$$

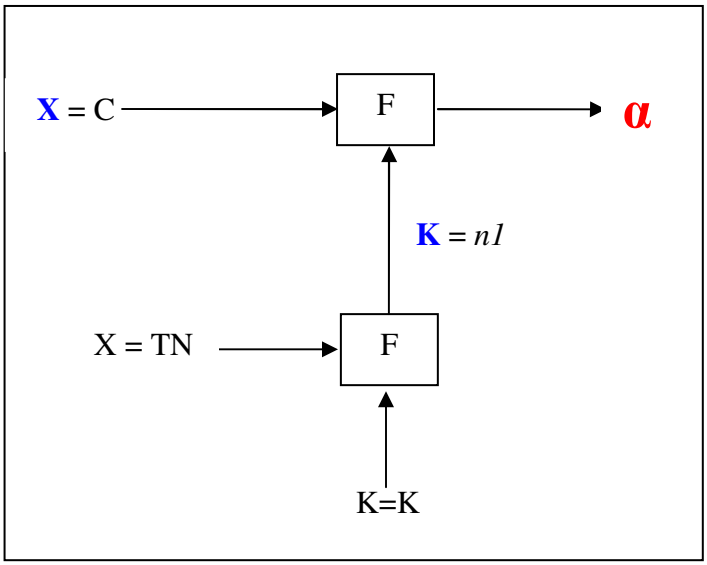

Figure 5.8: Creating Message α from PRF

Step 3: *Tag Authentication:* Reader will use the values of K, TN, C and C_{last} in the database to carry out an exhaustive search to find that tag whose response is the same as the message received. We have two cases for tag authentication.

Case 1: K, TN and C are used to authenticate the tag as below:

$$F(f_k(TN), C) \stackrel{?}{=} \alpha$$

This means that tag exits in the database and it's a new authentication.

Case 2: OR K, TN and C_{last} are used to authenticate the tag as below:

$$F(f_{klast}(TN), C_{klast}) \stackrel{?}{=} \alpha$$

This means that the back-end database already updated its C value at the previous authentication but the tag did not. This will catch the replay attack and will prevent the attacker from desynchronizing the database.

Step 4: *Update Reader:* The Back-end database will update information about the tag as follows:

In case 1, where K, TN and C were used to authenticate the tag, values will be updated as follows:

$$C_{last} = C; \qquad C = C + 1$$

In case 2, where K, TN and C_{last} were used, we do not update the values in the database. This means a tag is trying to use the previous authentication values, showing either tag did not update last time or some adversary is trying to hack the system.

Step 5: Reader generates pseudo-random numbers *n2* using PRF with master key K and tag number TN. Then it creates message β as shown in figure 5.9 using PRF with *n2* and C as input and sends this message to tag as follows:

$$n2 = f_k(x) \Rightarrow f_k(TN) = F(K, TN)$$

$$\beta = F(n3, C)$$

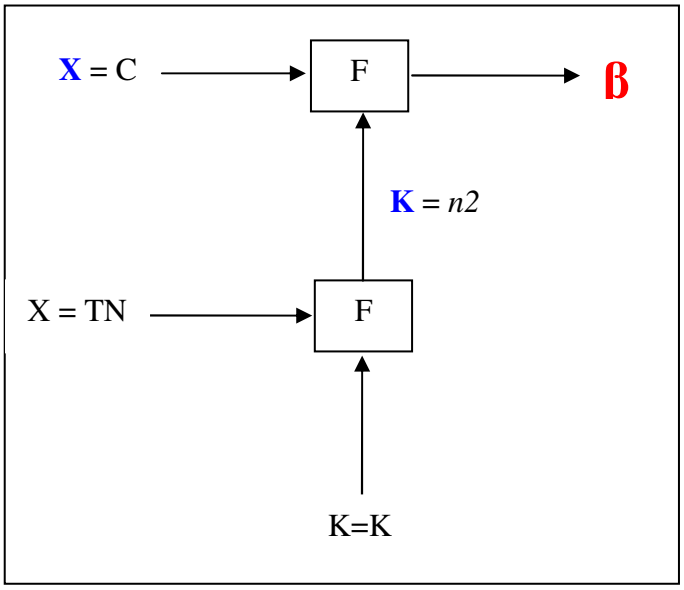

Figure 5.9: Creating Message β from PRF

Step 6: *Reader Authentication:* Tag will compute the new value for counter and will use this value to check if the message is the same as sent by the reader as shown below:

$$C' = C + 1$$

$$F(f_k(TN), C') \stackrel{?}{=} \beta$$

If the value comes out the same as the message received, that means the reader is legitimate, thus giving mutual authentication.

Step 7: *Update Tag:* Tag will update its values as below:

$$C = C + 1$$

Step 8: Tag will create the message γ using the values of *n1, n2* and ID as below:

$$\gamma = ID \oplus n1$$

The process of authentication is shown in figure 5.10 on next page.

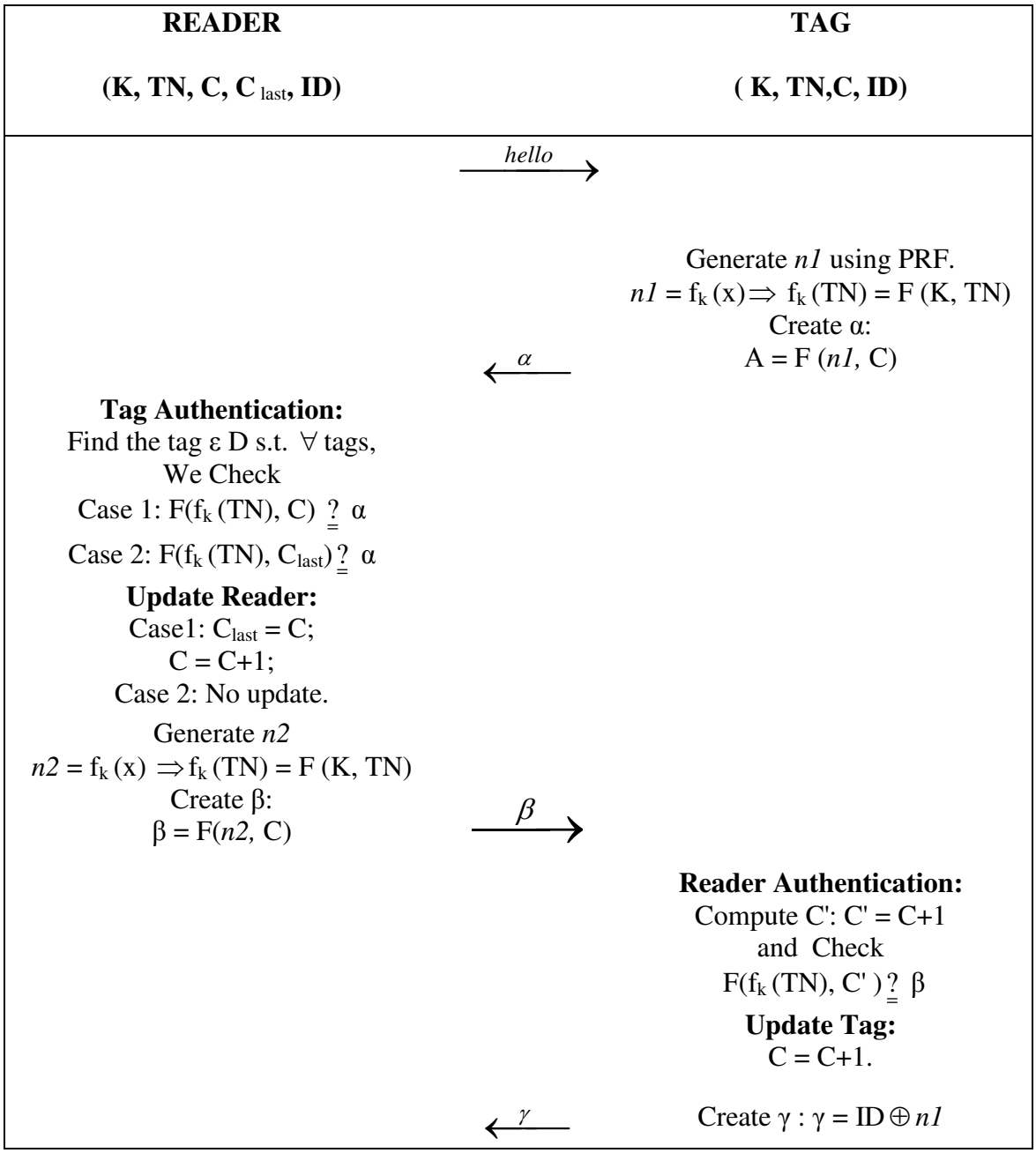

Figure 5.10: Protocol 5 (Using Universal Master Key)

CHAPTER 6

SECURITY ANALYSIS

6.1 Security Analysis of Protocol 1

Data Integrity: A part of the tag memory is rewritable, modifications are possible. In this part of the memory, the tag stores the *Tag-Index Number* (TIN), and shared secret keys (K1 and K2) associated with itself. If an attacker does succeed in modifying this part of the memory, then the reader would not recognize the tag and would lead to database desynchronization problem.

An attacker can obtain the random nonce *n* created by legitimate reader. Then, he creates its own random nonce *n'* and sends it to the tag impersonating as a legitimate reader. The tag sends the messages A', B', and C' to the reader which is intercepted by the attacker. From this, the attacker retrieves TIN, K1 and K2. Then using the random nonce *n* created by the legitimate reader, it will create messages A, B, and C and send them to the reader. The reader authenticates the tag and creates a message D and sends it to the tag which is intercepted by the attacker. Instead of sending this message D created by the legitimate reader, the attacker instead sends D' to the tag which does not authenticates the reader. In this case only the tag is authenticated and Keys are updated at the reader side only and not on the tag side giving rise to database desynchronization problems.

Another scenario is in which the attacker eavesdrops the messages and at the end can suppress the message D from being sent to the tag. In this case, values at the reader side will update and not on the tag side which can give rise to the database desynchronization problem.

Another scenario is in which the attacker instead of sending messages A, B, C to the reader, retrieves the values of K1, K2, TIN and *n* and use them to create message D. Then attacker sends this message to the tag, which authenticates it, updates the values on the tag side. In this case, tag side values are updated and not the reader side values, leading to the desynchronization problem. In this case, this tag will become useless. This case is however not possible due to the reason that the channel from tag to reader is much harder to eavesdrop than the channel from reader to tag.

This shows that by manipulating message D, this protocol can give rise to a problem. However any change to messages A, B or C does not have any effect because if those messages are changed, it will not authenticate the tag.

Mutual Authentication: This protocol is designed to provide both tag-to-reader authentication, which is achieved by message A, B, C and reader-to-tag authentication, obtained by message D.

Forward Security: This protocol provides Forward Security. Its the property that security of message sent today will be valid tomorrow i.e. data transmitted today will still be secure even if the secret tag information is revealed by tampering in the future. A future security compromise on an RFID tag will not reveal data previously transmitted. If the attacker is able to get the data from the tag, he cannot trace the data back through past

events in which the tag was involved. The adversary, who only eavesdrops on the tag output, cannot associate the current output with past output. Forward security requires that old keys be unpredictable from new keys i.e. it's unfeasible to compute previous keys and outputs from the current key.

Since Key updating is done during the mutual authentication process using random numbers, it makes it impossible for the attacker to guess the values in future. Hence a future security compromise on an RFID tag will not reveal previous values of the shared secret key.

Replay Attack: The key (K1 & K2) freshness for each successful read attempt prevents reply attacks.

An eavesdropper could store the messages interchanged between the reader and the tag during different protocol runs. Then, he could try to impersonate as a tag, replaying the message A, B, C to the reader seen in any of the protocol runs. He could try to impersonate as a reader too, by replaying the message D to the tag. It seems that this could cause the loss of synchronization between the database and the tag, but this is not the case because after the successful read attempt in this mutual authentication protocol, the Key (K) is updated, which makes the previously used messages invalid.

Replay attack is also prevented because the $K1_{last}$ and $K2_{last}$ store the previous last successful Keys. If the attacker tries to replay the messages A, B and C, using the Keys K1 and K2, it will be detected and no action will be taken by the reader. If the attacker tries to replay the message D to the tag, it will not authenticate the tag and no update will be done.

Man-in-the-middle Attack: A man-in-the-middle attack is possible. An attacker can impersonate as a legitimate reader and get the information from the tag, so he can impersonate as the legitimate tag responding to the reader. Thus, the attacker easily can be authenticated by the legitimate reader before the next session. As the attacker can easily make messages A, B and C; hence the man-in-the-middle attack is possible.

However we assume a restriction on the ability of an attacker to perform man-in-the-middle attack as the attacker would run the risk of being discovered since the attack would have to take place in a monitored environment.

Data Confidentiality: All the information about the item to which the tag is attached is stored in TagID. This Tag ID is kept secure which guarantees user privacy. The tag ID is send to the reader in secure form. The tag sends it in the message E, where the ID is exclusive-ORed with the TIN, then the result is exclusive-ORed with the random number *n,* created by the reader. This ID can be retrieved by an eavesdropper from message E easily, therefore this protocol do not provide us with data confidentiality.

Tag Anonymity: This protocol does not provide tag anonymity. As the TIN is sent back to the reader in message C, which will always be constant. If we modify TIN on each successful read attempt, then in case of man-in-the-middle attack, that tag cannot be identified.

Indistinguishability: The values emitted by the tag should not be such that the attacker can easily identify the tag. The operations used in this protocol makes the data transmitted between tag and reader indistinguishable. All the messages passed to and fro between tag and reader looks identical to the eavesdropper.

Forgery Resistance: This protocol does not prevent forgery. If an attacker is able to retrieve the values of TIN, K1, K2 and ID from a legitimate tag, then he can simply copy the information to make a clone tag.

However we assume a restriction on the ability of an attacker to record those values as he would run the risk of being discovered since the attack would have to take place in a controlled environment.

Data Recovery: This protocol does not provide data recovery. In case where the messages A, B and C are blocked from the tag, that data cannot be retrieved. In such case, the reader will not update the values. In case the message D is blocked or modified, the reader would have updated the values and not the tag. When the reader queries the tag next time, it will send the same message D, which will update the values on the tag side too.

6.2 Security Analysis of Protocol 2

Data Integrity: In this protocol, the attacker has to send a random nonce *n1* to the tag. The tag will generate a random nonce *n2,* then create messages A, B, C and send them to the attacker. This way attacker can retrieve the TIN from message C. Since messages A and B are using the values of K1, K2 and the random number generated by the tag, the attacker cannot retrieve the values of K1, K2 and *n2* at this point. If the attacker intercepts the message D from the reader, he can obtain *n2* by exclusive-ORing of message D with TIN, A and B.

If the message D is suppressed or altered, then the tag will not know or will not authenticate the reader respectively. Since message D is created after the reader updates its values, the values on the tag side will be different causing data integrity compromise.

If messages A and B from the tag are suppressed or altered by the attacker, then the reader will not authenticate the tag and it will not generate message D from which the attacker retrieves *n2.* This means that the attacker has no control over updating the values on the reader side.

The effect of changing the values at the reader side, do nothing as this protocol catches such ambiguity thus providing us data integrity.

Mutual Authentication: Mutual authentication is achieved by messages A, B, C, and D. Tag-to-reader authentication is achieved from messages A, B, and C. whereas reader-to-tag authentication is achieved from message D.

Forward Security: This protocol provides forward security i.e. data transmitted today will still be secure even if secret tag information is revealed by tampering in the future. Forward security requires that old keys be unpredictable from new keys. As Key updating is done during mutual authentication using the random number generated by the tag, it is impossible for an attacker to guess the values or to make an association between the current and past outputs. Thus the contents of memory in the tag do not give any hint to detecting past outputs.

Replay Attack: The key (K1 & K2) freshness for each successful read attempt prevents reply attacks. Replaying messages A, B, C to the reader will cause no harm as these messages will be unable to authenticate the attacker as a legitimate tag because once the messages were used, the values of K1, K2 are updated at the reader end making these

messages invalid. Replaying message D to the tag will have no effect either as the message will be unable to authenticate the attacker as a legitimate reader.

Man-in-the-middle Attack: A man-in-the-middle attack is possible. If the attacker is able to create messages A, B and C, then he can impersonate as a legitimate tag and make the legitimate reader to authenticate it. An attacker can retrieve random nonce *n2,* from message D and then use it to make those messages.

However we assume a restriction on the ability of an attacker to perform a man-in-the-middle attack as the attacker would run the risk of being discovered since the attack would have to take place in a monitored environment.

Data Confidentiality: This protocol does not provide user data confidentiality. Tag sends its ID in the message E, where the ID is exclusive-ORed with the TIN, and then the result is exclusive-ORed with the random number *n1* and *n2*. This hides the tag ID from a nearby eavesdropper equipped with an RFID reader who listens to the message E. An attacker who already knows the values of TIN, *n1* and *n2*, will get the ID from the message E.

Tag Anonymity: This protocol does not provide Tag anonymity. The TIN is sent back to the reader in message C, which will always be constant. If we modify the TIN on each successful read attempt, then in case of man-in-the-middle, that tag cannot be identified.

Indistinguishability: The operations used in this protocol makes the data transmitted between tag and reader indistinguishable. All the messages passed to and fro between tag and reader looks identical to the eavesdropper.

Forgery Resistance: This protocol does not prevent forgery. If an attacker is able to retrieve the values of TIN, K1, K2 and ID from a legitimate tag, then he can simply copy the information to make a clone tag.

We also assume a restriction on the ability of an attacker to record those values as he would run the risk of being discovered since the attack would have to take place in a monitored environment.

Data Recovery: This protocol provides data recovery. If the attacker blocks the messages A, B, and C from the tag, he cannot create message D. The message D can only be created using random nonce *n2,* which can be retrieved only from message D from the reader. Thus changing the values on the tag side only is not possible. In case the message D is blocked or modified, the reader would have updated the values and not the tag. When the reader queries the tag next time, it will send the same message D, which will update the values on the tag side too.

6.3 Security Analysis of Protocol 3

Data Integrity: This protocol provides data integrity (information related to tag) i.e. TIN, K1 and K2.

Modifying the values only at the reader end is possible. After mutual authentication, the reader updates its values first. Hence, if the message F from reader is blocked or modified, then the tag will not know or will not validate the message respectively causing data integrity compromise which can lead to database desynchronization. However this ambiguity will be caught by this protocol in the next run

as it will fall under case 2, which will not update the values on the reader side and just replay the previous message F.

Modifying the values on the tag side only is not possible. If the attacker blocks or modifies the message C to the reader, then the reader will not know or validate the tag respectively. An attacker cannot create message F on its own, as it needs a random number *n* generated by a legitimate reader, which he can not retrieve.

Mutual Authentication: This protocol provides mutual authentication in which messages A and B provides reader-to-tag authentication, and message C provides tag-to-reader authentication.

Forward Security: This protocol provides forward security i.e. data transmitted today will still be secure even if the secret tag information is revealed by tampering in the future. Since forward security requires that old keys be unpredictable from new keys i.e. it is unfeasible to compute previous keys and outputs from the current key. As Key updating is done using the random number generated by the reader, after mutual authentication it's impossible for an attacker to guess the values or to make an association between the current and past outputs. Thus contents of memory in the tag do not provide any hint on detecting past outputs.

Replay Attack: This protocol prevents replay attacks because key (K1 & K2) refreshing takes place after mutual authentication.

An attacker could store the messages interchanged between the reader and the tag (different protocol runs). Then he could try to impersonate a reader, by replaying the messages A, B and F to the tag. It may appear that the tag will authenticate the attacker as a legitimate reader, however, this is not possible as the values of K1, K2 would have to

be updated in the last mutual authentication run, which makes the previously used messages invalid.

An attacker could try to impersonate as a tag too, replaying the message C to the reader seen in any of the protocol runs. This attack is prevented because the $K1_{last}$ stores the previous last successful value of K1. If the attacker tries to replay the messages C, it will be detected and no action will be taken by the reader.

Man-in-the-middle Attack: A man-in-the-middle attack is possible. An attacker can impersonate as a legitimate reader and get the information from the tag, so that he can impersonate as the legitimate tag responding to the reader. Thus, the attacker can easily be authenticated by the legitimate reader before the next session. The attacker can retrieve random nonce *n,* from message F and then he can make a message C. Thus, the man-in-the-middle attack is possible.

We assume a restriction on the ability of an attacker to perform MITM attack by blocking the messages, as the attacker would run the risk of being discovered since the attack would have to take place in a monitored environment.

Data Confidentiality: This protocol does not provide user data confidentiality. The tag sends its ID in the message D, where the ID is exclusive-ORed with the random number *n* generated by a legitimate reader. This hides the tag ID from a nearby eavesdropper equipped with an RFID reader who listens to the message D. An attacker, who wants to retrieve the ID from message D will have to wait till message F is created so that he can get *n.*

Tag Anonymity: This protocol does not provide Tag anonymity. As the TIN is sent back to the reader in response to a reader's message *'hello'*, which will always be constant. If

we modify TIN on each successful read attempt, then in case of MIMA, that tag cannot be identified.

Indistinguishability: The operations used in this protocol makes the data transmitted between tag and reader indistinguishable. All the messages passed to and fro between tag and reader looks identical to the eavesdropper.

Forgery Resistance: This protocol does not prevent forgery. If an attacker is able to retrieve the values of TIN, K1, K2 and ID from a legitimate tag, then he can simply copy the information to make a clone tag.

We also assume a restriction on the ability of an attacker to record those values as he would run the risk of being discovered since the attack would have to take place in a monitored environment.

Data Recovery: This protocol provides data recovery. In case the message F is blocked or modified, the reader would have updated the values and not the tag. When the reader queries tag next time, it will send the same message F, which will update the values on the tag side too. Changing the values on the tag side alone is not possible as message C cannot be created without knowing the random nonce *n.*

6.4 Security Analysis of Protocol 4

Data Integrity: This protocol provides data integrity for both the tag as well as the item to which that tag is attached. In this protocol an attacker is unable to modify any values either on the tag or the reader side making the tag data secure.

If the attacker blocks message α, then he cannot create β on its own. The attacker may intercept the message α from the tag, modify it and send α' to the reader. This will have no effect because this message will not authenticate the tag to the reader.

If the attacker blocks the message β from going to the tag, then the reader would have updated and not the tag. On the next authentication step, the reader would recognize this because it has stored the last value of K.

If the attacker intercepts the message β from the reader, modifies it, and sends β' to the tag, it will not authenticate the reader to the tag as the values have to be the same. In this case, the tag will not update the values and will not send the message γ to the attacker. On the next authentication step, this tag will update its values.

If the attacker intercepts the message γ, then the attacker has to guess random numbers *n1* and *n2* created by the tag. Since *n1*and *n2* are created from PRF using master key K, those values have changed during the last run since it was updated and so the value is of no use to the attacker.

If the attacker sends the message 'hello' to the tag, it will generate its random number *n1* and *n2* from the PRF making use of master key as the secret hidden seed. The tag will then create a message α, which is sent back to the attacker. From this message, the attacker cannot retrieve anything. At this point the attacker has to guess three things, first he has to guess the master key K, second, the PRF used by the tag to generate *n1* and *n2,* and thirdly, guess the PRF used to generate the message α.

Similarly, the message β generated by the reader makes no sense to the attacker. To retrieve anything from this message, the attacker has to guess the new master key K,

guess the PRF used to generate *n3, n4* and then guess the PRF used to generate the message β.

If the attacker is able to guess those PRFs, then he has to guess the master key K at a certain point in time too because they are updated at each successful authentication.

Mutual Authentication: This protocol is designed to provide both tag-to-reader authentication, which is achieved by message α and reader-to-tag authentication, obtained by message β.

Forward Security: This protocol is forward secure. Since the tags do not store any historic data, even if the attacker succeeds in guessing the PRF, he will not be able to retrieve any past information about the tag because the master key K value is updated on each authentication. It will be unfeasible to compute previous keys and outputs from the current key.

Replay Attack: The master key K freshness for each successful authentication prevents from reply attacks.

An eavesdropper could store the messages α and β between the reader and the tag during different protocol runs. Then he could try to impersonate a tag, replaying the message α to the reader. He could try to impersonate a reader too, by replaying the message β to the tag. It may appear that this could cause the loss of synchronization between the database and the tag, but this is not possible because the master key K value is updated after each successful authentication making the previously used messages invalid.

Man-in-the-middle Attack: A man-in-the-middle attack is not possible. An attacker can impersonate as a legitimate reader and get the information from the tag, so he can

impersonate as the legitimate tag responding to the reader. Thus, the attacker can easily be authenticated by the legitimate reader before the next session. As the attacker cannot create message α, the man-in-the-middle attack is not possible.

Data Confidentiality: All the information about the item to which the tag is attached is stored in the ID. This ID is kept secure which guarantees user privacy. This ID is sent to the reader in secure form. The tag sends it in the message γ in which the ID is exclusive-ORed with *n1*, then the result exclusive-ORed with *n2* making the data more secure and meaningless to the attacker.

Tag Anonymity: During each successful mutual authentication, the master key is updated. This makes the tag partially anonymous.

If the attacker sends the message 'hello' at time t1 to get the message α, and then tries again at time t2 to get the same message α, this way the attacker can track the tag. However if the legitimate reader reads between time t1 and t2, then the attacker cannot track the tag because the master key value would have changed by that time.

Indistinguishability: The operations used in this protocol makes the data transmitted between the tag and the reader indistinguishable. All the messages passed to and fro between the tag and the reader looks identical to the eavesdropper.

Forgery Resistance: If an attacker is able to retrieve the value of K from a legitimate tag, then he can simply copy the information to make a clone tag. For an attacker to retrieve ID from message γ, he has to retrieve random number *n1* and *n2* created by the tag using PRF making use of master key K. He cannot retrieve those numbers from message α. Thus the real value of tag i.e. the ID can not be retrieved by the attacker; this makes

copying the values to the clone tag useless. This protocol provides protection against forgery.

Data Recovery: This protocol provides data recovery. In case the message β is blocked or modified, the reader would have updated the values and not the tag. When the reader queries tag next time, it will send the same message β, which will update the values on the tag side too.

6.5 Security Analysis of Protocol 5

Data Integrity: This protocol provides data integrity for both tag as well as the item to which that tag is attached. In this protocol the attacker is unable to modify any values either on the tag or the reader side making the tag data secure. Data integrity of the item to which the tag is attached is not compromised either, because the information about the item is stored in the ID, which is sent in the secure form.

If the attacker blocks the message α, then he cannot create β on its own. If the attacker intercepts the message α from the tag, modifies it and sends α' to the reader, then this message will not authenticate the tag to the reader.

If the attacker blocks the message β from reaching the tag, then only in this case the reader side will have updated the values for that particular tag, but not the tag side giving rise to desynchronization of the database. However, this is not the case as the last value of C is stored in the database and for the next authentication; it can recognize that tag easily making use of case 2.

If the attacker intercepts message β from the reader, modifies it, and sends β' to the tag, it will not authenticate the reader to the tag. In this case, the tag will not update the values and will not send the message γ to the attacker.

If the attacker sends the message 'hello' to the tag, it will generate its random number *n* from the PRF making use of the master key as the secret hidden seed. The tag will then create a message α, which is sent back to the attacker. From this message, the attacker cannot retrieve anything. At this point attacker has to guess four things, first he has to guess the master key K, second, the TN of the tag, third, the PRF used by the tag to generate *n1,* and fourth, the value of counter C to get anything out of message α.

Similarly, the message β generated by the reader, makes no sense to the attacker. To retrieve anything from this message, the attacker has to guess the master key K, guess the PRF used to generate *n2* and guess the counter C used to create message β.

If the attacker intercepts the message γ, then the attacker has to guess the random number *n1* created by the tag. Since *n1* is generated using the PRF with master key K, *n1* cannot be retrieved.

Mutual Authentication: This protocol is designed to provide both tag-to-reader authentication, which is achieved by message α and reader-to-tag authentication, obtained by message β.

Forward Security: Since the tags do not store any historic data, even if the attacker succeeds in guessing the PRF, he will not be able to retrieve any past information about the tag because the counter C value is updated on each mutual authentication. An attacker cannot guess the previous outputs from the tag as the counter values C changes on each

mutual authentication. Thus it is hard for an attacker to guess what the output was back in time.

Replay Attack: The counter value C freshness for each successful authentication prevents from reply attacks. An eavesdropper could store the messages α and β between the reader and the tag during different protocol runs. Then, he could try to impersonate a tag, replaying the message α to the reader. If that happens it will fall under case 2 of reader authentication in which no update will be done and the reader will create the same message β again.

An eavesdropper could try to impersonate a reader too, by replaying the message β to the tag. If that happens, the tag is not going to authenticate that response as the value of the counter was changed on the last authentication making this message invalid.

Man-in-the-middle Attack: A man-in-the-middle attack is not possible. An attacker can impersonate as a legitimate reader and get the information from tag, so he can impersonate as the legitimate tag responding to the reader. Thus, the attacker easily can be authenticated by the legitimate reader before the next session. As the attacker can not create message α, so man-in-the-middle attack is not possible.

Data Confidentiality: All the information about the item to which the tag is attached is stored in the ID. This ID is kept secure which guarantees user privacy. This ID is sent to the reader in secure form. The tag sends it in the message γ in which the ID is exclusive-ORed with *n1* making the data more secure and meaningless to the attacker.

Tag Anonymity: During each successful mutual authentication, counter value C is updated which makes the tag partially anonymous.

If the attacker sends the message 'hello' at time t1 to get the message α, and then tries again at time t2 to get the same message α in this manner the attacker can track the tag. However if the legitimate reader reads between times t1 and t2, then the attacker cannot track the tag because the counter value would have changed by that time.

Indistinguishability: The operations used in this protocol makes the data transmitted between tag and reader indistinguishable. All the messages passed to and fro between the tag and the reader looks identical to the eavesdropper.

Forgery Resistance: If an attacker is able to retrieve the value of K, TN and C from a legitimate tag, then he can simply copy the information to make a clone tag. For an attacker to retrieve the ID from message γ, he has to retrieve random number *n1* generated by the tag using PRF making use of master key K. He cannot retrieve that number from message α. Thus the real value of the tag i.e. the ID can not be retrieved by the attacker. This makes copying the values to the clone tag useless. This protocol provides protection against forgery.

Data Recovery: This protocol provides data recovery. In case the message β is blocked or modified, the reader would have updated the values and not the tag. When the reader queries the tag next time, it will send the same message β, which will update the values on the tag side too.

All the proposed protocols in this thesis are compared with each other in table 4 for security analysis.

Table 4: Comparison of Security Requirements between Proposed Protocols

Protocol	P 1	P 2	P 3	P 4	P 5
User Data confidentiality	X	X	X	O	O
Tag Anonymity	X	X	X	Δ	Δ

Data Integrity	X	O	O	O	O
Mutual Authentication	O	O	O	O	O
Forward Security	O	O	O	O	O
Man-in-the-middle Attack	X	X	X	O	O
Replay Attack	O	O	O	O	O
Forgery Resistance	X	X	X	O	O
Indistinguishability	O	O	O	O	O
Data Recovery	X	O	O	O	O

†† Notation: O Satisfied Δ Partially Satisfied X Not Satisfied

All the proposed protocols in this thesis are compared with other protocols in table 5.

Table 5: Comparison of Security Requirements with Other Protocols

Protocol	**HLS**	**EHLS**	**HBIV**	**MAP**	**P 1**	**P 2**	**P 3**	**P 4**	**P 5**
User Data confidentiality	X	Δ	Δ	O	X	X	X	O	O
Tag Anonymity	X	Δ	Δ	O	X	X	X	Δ	Δ
Data Integrity	Δ	Δ	O	O	X	O	O	O	O
Mutual Authentication	Δ	Δ	Δ	O	O	O	O	O	O
Forward Security	Δ	Δ	O	O	O	O	O	O	O
Man-in-the-middle Attack	Δ	Δ	X	O	X	X	X	O	O
Replay Attack	Δ	Δ	O	O	O	O	O	O	O
Forgery Resistance	X	X	X	O	X	X	X	O	O
Indistinguishability	X	X	X	O	O	O	O	O	O
Data Recovery	X	X	O	O	X	O	O	O	O

†† Notation: O Satisfied Δ Partially Satisfied X Not Satisfied

From the table 5, we can see that the Mutual authentication protocol proposed by Yang [26] satisfies all the security requirements. However, that protocol uses a hash function on the tag which makes them much more expensive as compared to our proposed protocols. We have tried to come up with the same level of security without using expensive hash functions and making use of primitive operations and pseudo-random-functions(PRF).

CHAPTER 7

PERFORMANCE ANALYSIS

It is important to carefully analyze the performance of the proposed scheme, to show that it can be safely implemented even in low-cost tags. It is assumed that the connection between the reader and the database is secure. Moreover, the readers and databases are devices with non-limited computing and storing capabilities. Due to these reasons we can collapse the notion of the reader and the back-end database into single entity (R+B). Therefore, in the performance analysis of our protocol, we consider the reader and database form a single entity.

7.1 Performance Analysis of Protocol 1

Computation Overhead: In this protocol, the tag only needs XOR operation whereas the reader needs XOR operation and PRNG. This protocol provides the minimal computation load on both the tag and reader side. Low-cost RFID tags are very limited devices, with only a small amounts of memory, and very constrained computationally (only between 200 and 2000 logic gates can be devoted to security-based tasks). Additionally, one of the main drawbacks that hash-based solutions have is that the load on the server-side (R+B) is proportional to the number of tags. Our proposal have completely solved this problem by using *Tag-Index Number* (TIN) that allows a tag to be univocally identified.

Storage Overhead: We assume that the sizes of all components are L bits. Our protocol is based on pseudonyms, concretely on an L-bit TIN, so each tag has to store it. For the implementation of our protocol, each tag should have an associated key of length 2L, which is used for mutual authentication. Moreover, the tag has to store a unique identification number (ID) of length L. Thus the tag needs a memory size of 4L bits. However the reader needs memory size of 6L due to additional storage cost of $K1_{last}$ and $K2_{last}$.

Communication Overhead: The proposed protocol accomplishes mutual authentication between tag (T) and reader (R+B), requiring only three rounds. Taking into account that low cost tags are passive, and that the communication can only be initiated by a reader, three rounds may be considered as a reasonable number for mutual authentication in RFID environments. Therefore the proposed protocol is feasible and practical for a low-cost RFID environment.

7.2 Performance analysis for Protocol 2

Computation Overhead: In this protocol, the tag needs a PRNG and XOR operation whereas the reader needs XOR operation and PRNG. This protocol has an extra overhead of generating PRNG on the tag side. However the tag is identified easily making use of TIN in the database.

Storage Overhead: We assume that the sizes of all components are L bits. This protocol needs L-bit TIN, each tag should have an associated key of length 2L and it has to store a unique identification number (ID) of length L. Thus the tag needs the memory size of 4L

bits. However the reader needs memory size of 6L bits due to additional storage cost of $K1_{last}$ and $K2_{last}$.

Communication Overhead: The proposed protocol accomplishes mutual authentication between tag (T) and reader (R), requiring only three rounds. Therefore the proposed protocol is feasible and practical for low-cost RFID environment.

7.3 Performance analysis for Protocol 3

Computation Overhead: In this protocol tag only needs XOR operation whereas the reader needs XOR operation and a PRNG. Since the tag is identified in the database using TIN, we don't have to go through the whole database to find the tag and compute its identity as done in other protocols.

Storage Overhead: We assume that the sizes of all components are L bits. This protocol needs L-bit TIN, each tag should have an associated key of length 2L and it has to store a unique identification number (ID) of length L. Thus the tag needs a memory size of 4L bits. However the reader needs memory size of 5L bits due to additional storage cost of $K1_{last}$.

Communication Overhead: The proposed protocol accomplishes mutual authentication between tag (T) and reader (R), requiring only four rounds. Therefore the proposed protocol is feasible and practical for low-cost RFID environment.

7.4 Performance analysis for Protocol 4

Computation Overhead: In this protocol, tag needs PRF operation whereas the reader needs PRF operation too.

When a reader sends a message 'hello' to the tag, it will generate two random numbers *n1* and *n2* from the PRF making use of the master Key and will respond with message α to the reader. Then the system has to carry out an exhaustive search to find that tag whose response is same as the message received. Therefore the system's workload is linear to the number of tags. If such a tag is found, then updating the reader side values takes place. The Reader will generate new random number *n3* and *n4* from the PRF, which it uses to make message β. The tag will then authenticate the reader.

Storage Overhead: We assume that the sizes of all components are L bits. For each tag, this protocol needs master key of length L and it has to store a unique identification number (ID) of length L. Thus the tag needs a memory size of 2L bits. Reader needs a memory size of 3L bits because of added K_{last}.

Communication Overhead: The proposed protocol accomplishes mutual authentication between tag (T) and reader (R), requiring only three rounds making it feasible and practical for low-cost RFID environment.

7.5 Performance analysis for Protocol 5

Computation Overhead: In this protocol, tag and reader both needs PRF operations.

When a reader sends a message 'hello' to the tag, it will generate a random numbers *n1* from the PRF making use of the master Key and the tag number TN, and will respond with message α to the reader. Then the system has to carry out an exhaustive search to find that tag whose response is the same as the message received. Therefore the system's workload is linear to the number of tags. If such tag is found, then updating the

reader side values takes place. The Reader will generate a new random number *n2* from the PRF, create message β and send it to the tag. The tag will then authenticate the reader.

Storage Overhead: We assume that the sizes of all components are L bits. For each tag, this protocol needs master key of length L, tag number TN, counter value C, and it has to store a unique identification number (ID) of length L. Thus the tag needs a memory size of 4L bits. The Reader needs the memory size of 5L bits because of added $\mathbf{C}_{last}$.

Communication Overhead: The proposed protocol accomplishes mutual authentication between tag (T) and reader (R), requiring only three rounds making it feasible and practical for low-cost RFID environment.

All the proposed protocols in this thesis are compared with each other for their computational loads and memory requirements in table 6 as shown below.

Table 6: Computational Loads and Memory Requirement for Proposed Protocols

Protocol	**Entity**	**P 1**	**P 2**	**P 3**	**P 4**	**P 5**
PRNG Operation	T	¬	1	¬	¬	¬
	R	1	1	1	¬	¬
PRF Operation	T	¬	¬	¬	6	2
	R	¬	¬	¬	$2n + 4$	$2n + 1$
Exclusive-OR Operations	T	11	13	9	2	3
	R	8	12	9	¬	$2n + 1$
Authentication Steps		3	3	4	3	3
Required Memory	T	4L	4L	4L	2L	4L
	R	6L	6L	5L	3L	5L
Identification Computation	R	O(1)	O(1)	O(1)	O(n)	O(n)

†† Notation: ¬ Not Required n Number of Tags L Size of Memory

Table 7 shows a comparison made by Yang [26] of the security requirements of different proposals. Our protocols are added to that table.

Table 7: Comparison of Computational Load and Memory Requirement with Other Protocols

Protocols	**Entity**	**HLS**	**EHLS**	**HBVI**	**MAP**	**P 1**	**P 2**	**P 3**	**P 4**	**P 5**
Hash Operations	T	1	2	3	2	¬	¬	¬	¬	¬
	B	¬	N	3	$2n$	¬	¬	¬	¬	¬
Keyed Hash Operation	R	¬	¬	¬	1	¬	¬	¬	¬	¬
	B	¬	¬	¬	1	¬	¬	¬	¬	¬
PRNG Operation	T	¬	1	¬	¬	¬	1	¬	¬	¬
	R	¬	¬	¬	1	1	1	1	¬	¬
	B	¬	¬	1	¬	¬	¬	¬	¬	¬
Basic Operations	T	¬	¬	¬	4	11	13	9	2	3
	R+B	¬	¬	¬	$2(n+1)$	8	12	9	¬	$2n+1$
Number of Encryption	B	¬	¬	¬	1	¬	¬	¬	¬	¬
Number of Decryption	R	¬	¬	¬	1	¬	¬	¬	¬	¬
Authentication Steps		6	5	5	5	3	3	4	3	3
Required Memory	T	$1\frac{1}{2}L$	1L	3L	$2\frac{1}{2}L$	4L	4L	4L	2L	4L
	R+B	$2\frac{1}{2}L$	$1\frac{1}{2}L$	9L	$9\frac{1}{2}L$	6L	6L	5L	3L	5L
Identification Computation	R	O(n)	O(n)	O(n)	O(n)	O(1)	O(1)	O(1)	O(n)	O(n)

†† Notation: ¬ Not Required n Number of Tags L Size of Memory

CHAPTER 8

APPLICATION OF PROPOSOD SECURO RFID PROTOCOLS

To reduce the cost of RFID tags, bulky data about products is stored in backend databases and accessible through the internet. Only a minimum amount of information such as product IDs and light-weight security primitives are stored in the RFID tags. These tags are attached to containers, pallets, and/or items. A networked RFID system proposed by EPCGlobal including Tag, Reader, Savant, Electronic Product Code Information System (EPCIS) and Object Name Service (ONS) is shown in the figure 8.1.

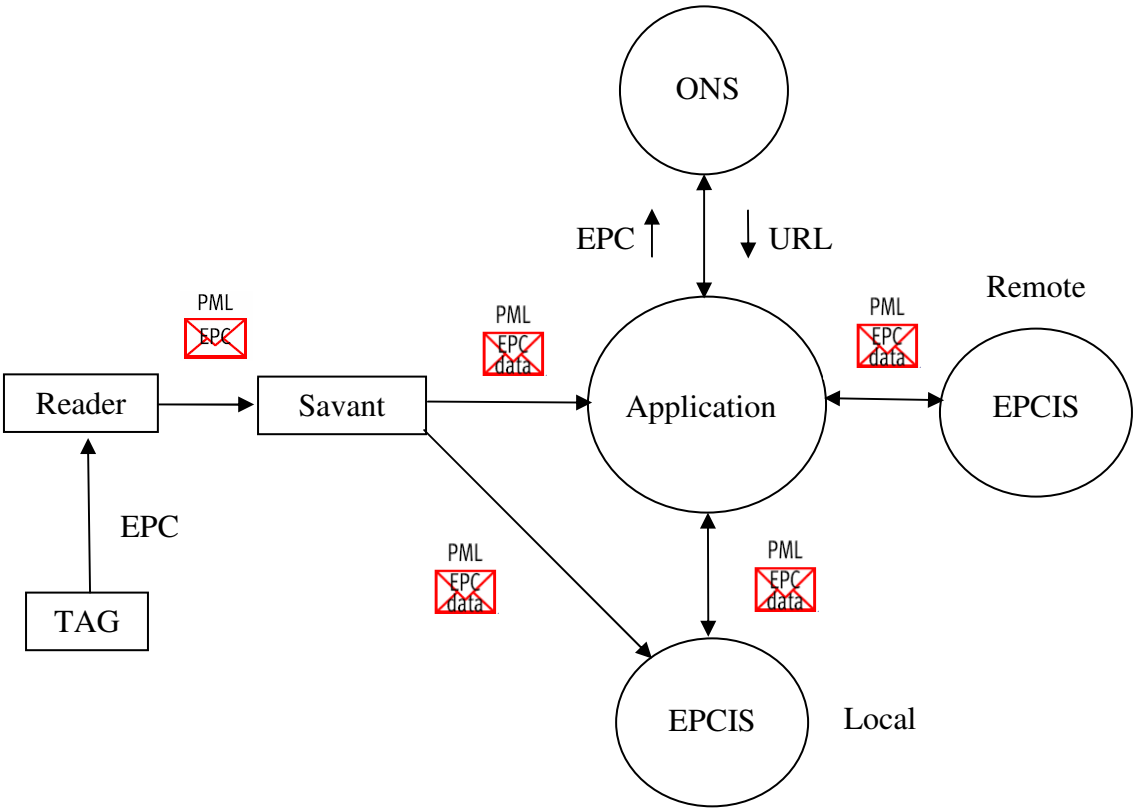

Fig 8.1 Networked RFID

8.1 Supply Chain Application

In supply chain systems, a supply chain partner uses RFID readers to collect product information from RFID tags. The collected information is then sent to savant system for further interpretation and process. Meanwhile, a supply chain information flow can take place between supply chain partners through internet connections as shown in the figure 8.2.

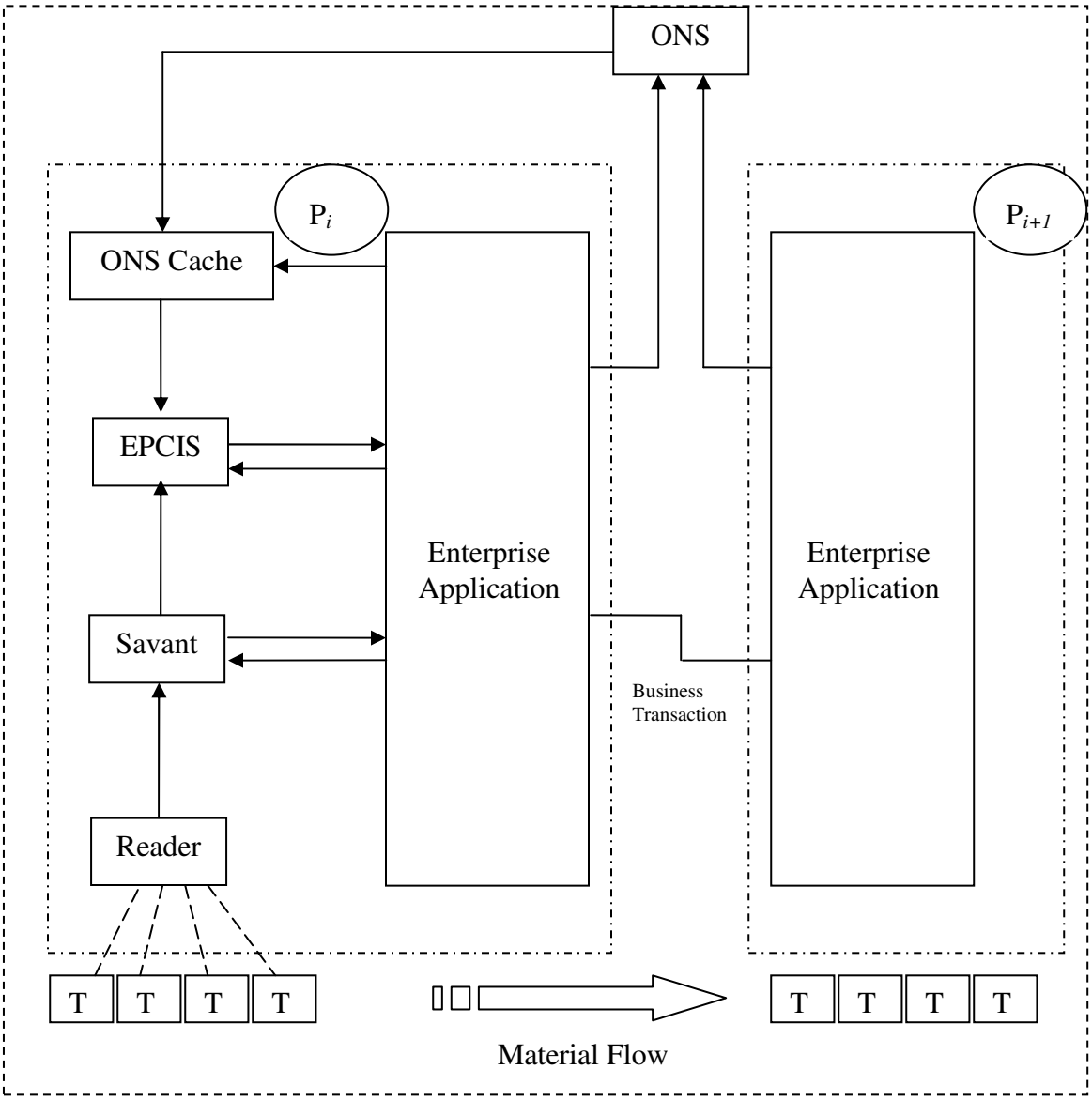

Fig 8.2 Networked RFID Systems in Supply Chain

8.1.1 System Model:

Yingjiu [29] proposed a protocol that is applied on a supply chain system. In that protocol they used a hash function on the tag. They have shown how to implement their protocol when a number of partners are engaged for a particular batch of tags. We have used the same strategy and have applied our protocol to that situation. We have also proved that the security requirements as proposed by them for supply chain are satisfied by our protocol. In our model, instead of using hash functions on tags, we have used PRF, which makes the tags much cheaper. We have proposed that the tag information should be transmitted to the next partner instead of the previous partner as proposed in their protocol.

We consider a supply chain consisting of N partners denoted by P_1, P_2 ... P_N. Material flow of items between the partners is equipped with RFID tags. It originates from P_1 and is shipped along the supply chain in the sequence of P_1, P_2...P_i, P_{i+1} ... P_N. When the flow arrives at P_1, it has to read and update all RFID tags.

We assume that every partner has limited knowledge of its local neighborhood in the whole supply chain. Namely, for all $1 \leq i \leq N$, partner P_i is aware of its subsequent peer P_{i+1} and for all $1 \leq_i \leq N$, P_i is aware of its preceding peer P_{i-1}.

We do not consider physical attacks on legitimate readers, tags or tag-item attachment and denial of service attacks in this supply chain.

8.1.2 System Setup:

We consider using protocol 4 with some modifications for this application as its more secure than the other proposed protocols. Master keys transmission between the

partners is done using public key encryption. Let *l* be the maximum length of RFID tag's ID. If partner P_i is the first to start the supply chain then master keys for all the tags are generated and stored. When the tags leave P_i facility, all the updated master keys and other data are transmitted to P_{i+1}.

8.1.2.1 Tag Initialization: P_1 is responsible for RFID tag initialization. The data pertaining to tag includes master key K and ID. If P_1 is the starting point in the supply chain then P_1 will generate a master key and assign this value to K in the tag. P_1 will store the item serial number or EPC, to which the tag is supposed to attach, in the ID field.

8.1.2.2 Database Initialization: Using protocol 4 for this application, we modify the database by adding field S in the database which is a binary bit. 's=1' to mean that the corresponding RFID tag has been processed. Otherwise we set it to 0.

K	K_{last}	ID	S

Since P_1 is the originator of the supply chain, it initializes D_1 after setting up RFID tags. It will assign the value of the master key generated for that tag to field K, K_{last} will have the same value, EPC or serial number of item to field ID and s=0 and will do it for all the tags.

Each partner P_i maintains a database D_i in its local system. D_i contains all RFID information with respect to that shipment. Each tuple in the database corresponds to a tag. For convenience, the j-th entry in the database, (k_j, k_{lastj} , id_j, s_j), is denoted by d_j. D_i is represented by $\{d_1, d_2 \ldots d_n\}$, where *n* is the number of tags for the current shipment. Initially, D_i is empty.

To process the incoming material flow, P_i either receives or downloads all updated master Keys and ID of D_{i-1} from P_{i-1} through a secure communication channel. P_i will set all the $s_j = 0$, $(1 \leq j \leq n)$.

8.1.2.3 RFID Read/Write Protocol: Our Protocol 4 works as follows in this case.

Reader Protocol: The ultimate goal of this protocol for P_i's reader is to extract the tag's ID and retrieve its corresponding record from the database. Our protocol described below shows the interaction between one tag and a reader.

Step 1: Reader → Tag: The reader sends message 'hello' to the tag.

Step 2: Tag → Reader: Tag generates *n1, n2*, then create message α using master key and sends α to the reader.

Step 3: Tag Authentication: Using the master key K and K_{last}, the reader computes all possible responses for all unmarked tags in the database D_i. Then the reader searches from the computation results. If match is found, it sets s=1, which means it's a legitimate tag being present in the database and it's being processed.

Step 4: Used in Write Protocol.(see below)

Step 5: Used in Write Protocol.

Step 6: Reader Authentication: Using the new master key, the tag computes the response. If the response is the same as the message β, it authenticates the reader.

Step 7: Used in Write Protocol.

Step 8: Tag → Reader: Tag creates message γ and sends it to reader, which includes the ID of the item.

Write Protocol: The write process is to update the tag's master key so that it can be accessed securely by the authorized readers of the next partner P_{i+1}. In essence, the reader of P_i writes K_{i+1} to an RFID tag. The protocol is as follows:

Step 4: Update Reader: New master key is generated using the old master key.

Step 5: Reader → Tag: Reader generates *n3, n4,* then creates message β using the master key and sends β to the tag.

Step 7: Update Tag: New master key is generated using the old master key.

8.1.3 Security Requirements in Supply Chain:

8.1.3.1 Visibility: In a supply chain, tracking of RFID tags should be provided. It should also provide the information about the last partner who has processed it. It allows the partners to track and monitor the progress of material flow without inefficient bar code scanning. While the supply chain partners are trusted and should be provided with supply chain visibility, however unauthorized readers should be prevented from understanding any tag's content and from tracking the movement of material flow. The following can be concluded from the above protocol.

- Without knowledge of the master key, no reader is able to obtain the tag's identity.
- Without knowledge of the master key, no reader is able to determine whether two tags belong to the same material flow.

8.1.3.2 Authoritative Access: RFID tags are only accessible by authorized readers of partners P_i. Only authorized readers are able to interpret the responses and extract their

identities, whereas a malicious reader obtains no meaningful information from its interrogation. We summarize the security with respect to authoritative access in the following statement.

- Consider an RFID tag delivered by partner P_{i-1} to partner P_i. Only P_i's reader is able to read the tag's ID. Furthermore, only P_i's reader is able to write to this tag.

8.1.3.2 Authenticity of Tags: Only legitimate RFID tags delivered by P_{i-1} will be accepted by P_i readers eliminating the replay and cloning attacks. Note that the supply chain reader is unable to distinguish between the original tag and a cloned malicious tag. The authenticity of tags in our protocol is summarized in the following statement.

- It is computationally infeasible for an attacker, without the knowledge of a master key, to find out a pair *n1* and *n2* to make a valid message α.

8.1.3.4 Unlinkablility: It should be unfeasible for the rouge reader to determine whether its interrogations are upon the same tag in inbound and outbound flow. In supply chain, a correlation of inbound flow and outbound flow reveals critical information about the company. Following can be concluded from the above protocol.

- Given a response t_1 from a tag prior to being processed by partner P_i and a response t_2 from a tag after being processed by P_i, it is unfeasible for a rouge reader to determine whether t_1 and t_2 are from the same tag. In other words, the tags are unlinkable for unauthorized readers.

CHAPTER 8

CONCLUSION AND FUTURE WORK

In this thesis we have investigated the security issues and requirements of RFID systems, and have proposed Ultra-Light weight (Protocol 1, Protocol 2, and Protocol 3) and Light weight (Protocol 4, Protocol5) protocols. From the security and performance analysis done in the previous chapters we come up with the following conclusions.

1. Ultra-Light weight protocols using primitive operations and pseudo-random number generator (PRNG) can provide the same level of security and performance without the use of expensive hash functions, symmetric encryption, and at much reduced cost.
2. Ultra-Light weight protocols are highly robust. In ultra-light weight protocols, use of tag-index number (TIN) reduces the time complexity for identifying the tags in the database.
3. In Light weight protocols, storing the previous value of shared key prevents the desynchronization problem.
4. Light weight protocols using pseudo-random functions (PRF) can provide the same level of security with the exception of total tag anonymity and data recovery at much reduced cost.

The protocol suite proposed in this thesis can work as efficiently and it's as secure as proposed by other people and fits the low-cost RFID system environment.

These protocols can be used for item level tagging depending on the environment. Ultra-Lightweight protocols can easily work in a controlled environment without the presence of an active attacker. Since these protocols are cheap to implement and they don't suffer from the scalability problem, they are best in such environment. These protocols can easily prevent eavesdropper and other attacks as shown. Light-weight protocols can be used in an environment where an active attacker is present. They can provide security for the item, however anonymity is partially fulfilled.

In the proposed protocols, we were only able to provide partial anonymity. Total anonymity can be added to these protocols at the cost of a random number generator on the tag side, as the response from tag is always the same in our case, which can increase the price of a tag. Total anonymity is left for future work.

REFERENCES

[1] Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita., "Cryptographic approach to "privacy-friendly" tags", In *RFID Privacy Workshop*, MIT, Massachusetts, USA, November 2003.

[2] Stephen A. Weis., Sanjay E. Sarma., Ronald L. Rivest., and Daniel W. Eengels., "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems", *Internationl Conference on Security in Pervasive Computin - SPC,* pp.454-469, March 2003.

[3] Istvan Vajda and Leevente Buttyan. "Lightweight authentication protocols for low-cost RFID tags". *In Second Workshop on Security in Ubiquitous Computing – Ubicomp 2003,* USA, October 2003.

[4] Ari Juels., "RFID Security and Privacy: A Research Survey". *IEEE Journal on Selected Areas in Communications,* 24(2), pp 381-394, Feb. 2006.

[5] Gildas Avoine., Etienne Dysli. And Phillippe Oechslin., "Reducing Time Complexity in RFID Systems". In B. Preneel and S. Taavares, editors*, Selected Areas in Cryptography – SAC 2005.*

[6] Mikko Lehtonen., Thorsten Staaake., Florian Michaheelles., and Elgar Fleisch. "From Identification to Authentication – A Review of RFID Product Authentication Techniques" *Workshop on RFID Security,* July 2006.

[7] Tasssos Dimitriou. "A Lightweight RFID Protocol to protect against Traceability and Cloning attacks". *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm,* September 2005.

[8] Auto-ID Center, "860MHz–960MHz Class I Radio Frequency Identification Tag Radio Frequency& Logical Communication Interface Specification Recommended Standard, Version 1.0.0", *Technical Report, MIT-AUTOID-TR-007,* November 2002.

[9] Ari Juels., Ronald Rivest., and Michaeel Szydlo., "The blocker tag: selective blocking of RFID tags for consumer privacy", *ACM Conference on Computer and Communications Security – ACM CCS,* pp. 103 -111, October 2003.

[10] mCloak for RFID tags, http://www.mobilecloak.com/mobilecloak/index.html [last accessed – April 22, 2007]

[11] Gildas Avoine., "Radio frequency identification: adversary model and attacks on existing protocols", *Technical Report LASEC – REPORT – 2005 – 001,* EPFL, Lusanne, Switzerland, September 2005.

[12] Dirk Henrici., and Paul Muller., "Hash-based Enhancement of Location Privacy for Radio-Frequency Identification Devices using Varying Identifiers", *Workshop on Pervasive Computing and Communications Security*, 2004

[13] Benessa Defend., Kevin fu., and Ari Juels, "Cryptanalysis of Two Lightweight RFID Authentication Schemes" , *PERCOMW* , pg 211-216. March 2007

[14] Ari Juels., "Minimalist cryptography for low-cost RFID tags". *Security of Communication Networks,* volume 3352 of LNCS, pp 149 – 164. Springer-Verlag, 2004.

[15] David Molnar., and David Wagner, "Privacy and security in library RFID: issues, practices, and architectures", *Conference on Computer and Communication Security – ACM CCS,* pp. 210 – 219, October 2004.

[16] Simson L. Garfinkel, Ari Juels, Ravi Pappu, "RFID Privacy: An Overview of Problems and Proposed Solutions," *IEEE Security and Privacy*, vol. 03, no. 3, pp. 34-43, May/Jun, 2005.

[17] S. Shepard., RFID Radio Frequency Identification, *MacGraw-Hill*, 2005.

[18] Auto-ID Center. 900 MHz class 0 radio frequency (RF) identification tag specification. Draft, March 2003.

[19] P. Peris-Lopez., J. C. Hernandez-Castro., J. M. Estevez-Tapiador., and A. Ribagorda. "LMAP: A Real Lightweight Mutual Authentication Protocol for Low-Cost RFID tags". *Proc. Of 2nd Workshop on RFID Security,* July 2006.

[20] Tieyan Li., and Guilin Wang. "Security Analysis of Two Ultra-Lightweight RFID Authentication Protocols". *IFIP SEC 2007*, May 2007.

[21] Sangshin Lee., Tomoyuki Asano., and Kwangjo Kim. "RFID Mutual Authentication Scheme based on Synchronized Secret Information". *Symposium on Cryptography and Information Security*, January 2006

[22] K. Yksel., J.P. Kaps., and B. Sunar. "Universal hash functions for emerging ultra-low-power networks". In *Proc. Of CNDS' 04,* 2004.

[23] Datasheet Helion Technology. High Performance MD5. Fast SHA-1. Fast SHA-256. Hash core for ASIC, 2005.

[24] JetStream Media Technologies. JetAES Tiny, Standard : Low Gate Count Low Data Rate AES Cores, October 2006.

[25] M. Feldhofer., S.Dominikus., and J. Wolkerstorfer. "Strong authentication for RFID systems using the AES algorithm". *In Proc. Of CHES' 04*, volume 3156 of LNCS, pg 357- 370, 2004.

[26] J. Yang., J. Park., H. Lee., K. Ren., and K. Kim. "Mutual authentication protocol for low-cost RFID". *Encrypt Workshop on RFID and Lightweight Crypto,* July 2005.

[27] M.R. Rieback., B. Crispo., and A. S. Tanenbaum. "The Evolution of RFID Security". *IEEE Pervasive Computing,* Volume 5, Issue 1, page 62-69 , 2006.

[28] H. Knospe., and H. Pohl. "RFID Security". *Information Secuirty Technical Report*, Volume 9, pg 39-50, Nov-Dec 2004.

[29] Yingjiu Li., and Xuhua Ding. "Protecting RFID Communications in Supply Chains". *ACM Symposium on InformAtion, Computer, and Communication Secuirty,* pages 234 – 241, Singapore, March 20-22, 2007.

[30] Certicom Suite B TRNG IP Core, http://www.certicom.com/download/aid-690/Suite%20B%20TRNG%20IP%20Core.pdf [last accessed – Nov 20, 2007]

[31] HangRok Lee., and DoWon Hong. "The tag authentication scheme using self-shrinking generator on RFID system". *Transactions on Engineering, Computing, and Technology,* Vol 18, pages 52 – 57, 2006.

VITA

Asrar Ahmed Omer

Candidate for the Degree of

Master of Science

Thesis: MUTUAL AUTHENTICATION PROTOCOLS FOR RFID SYSTEMS

Major Field: Computer Science

Biographical:

Personal Data: Born in Sialkot, Pakistan, September 1974 to Mr. and Mrs. Mehboob Ahmed

Education: Received my Matriculation degree from Crescent Model School, Lahore, Pakistan, in 1990. Received my Pre-Engineering from Government College, Lahore, Pakistan, in 1992. Received my Bs. Chemical Engineering from Punjab University, Lahore, Pakistan, in 1999. Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Tulsa, Oklahoma in December, 2007.

Professional Memberships: Member of American Computer Society (ACM).

Name: Asrar Ahmed Omer Date of Degree: December, 2007

Institution: Oklahoma State University Location: Tulsa, Oklahoma

Title of Study: MUTUAL AUTHENTICATION PROTOCOLS FOR RFID SYSTEMS

Pages in Study: 96 Candidate for the Degree of Master of Science

Major Field: Computer Science

Radio-Frequency Identification Devices (RFID) is emerging as a pervasive computing technology with numerous applications. Current low-cost RFID tags are highly resource-constrained and cannot support complex security mechanisms. Hence they have potential risks and may violate the privacy of their bearers. The challenge in providing security for low-cost RFID tags lies in that they are computationally weak devices, unable to perform even basic symmetric-key cryptographic operations as proposed in currently available protocols.

In this thesis we have analyzed the security issues and requirements for a RFID system. We have proposed a suite of lightweight mutual authentication protocols for low-cost RFID tags which offer an adequate level of security at much reduced cost. We also compare our proposed protocols with those proposed by others. Furthermore we apply our proposed protocol to secure a supply chain management system.

ADVISER'S APPROVAL: Dr. Johnson Thomas