CLONING PREVENTION PROTOCOL FOR RFID

by

Jignasa Shah

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
December 2010

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "CLONING PREVENTION PROTOCOL FOR RFID" by Jignasa Shah in partial fulfillment of the requirements for the degree of Master of Computer Science.

Dated: December 9, 2010

Supervisor: _____

Readers: _____

_____

DALHOUSIE UNIVERSITY

DATE: December 9, 2010

AUTHOR: Jignasa Shah

TITLE: Cloning Prevention Protocol For RFID

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: MCSc CONVOCATION: May YEAR: 2011

_____

Signature of Author

DEDICATION PAGE

This thesis is dedicated to my parents, who have taught me that knowledge is the greatest strength.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Radio Frequency Identification (RFID) is an emerging area under ubiquitous computing. RFID benefits include multiple read/write, longer read range and no requirement for line of sight. Due to security and privacy issues, RFID is not as popular as it should be. Tag cloning is one of the biggest threats to RFID systems. Easy access to RFID tags allows an attacker to replicate the tags and insert duplicate tags into the system. An RFID tag cloning attack can lead to access control or financial frauds in areas like supply chain management and government issued IDs.

In this thesis, a cloning prevention protocol is proposed. It uses light weight functions such as Pseudo Random Number Generator (PRNG) and compare function. A 3-way handshake with a secret key, frequency hopping mechanism and dynamic fake ID makes this protocol a secure authentication mechanism.

LIST OF ABBREVIATIONS USED

RFID	Radio Frequency Identification
ID	Identification
DoS	Denial-of-service
ISO	International Standards Organization
EPC	Electronic Product Code
Gen	Generation
KHz	Kilo Hertz
MHz	Mega Hertz
GHz	Giga Hertz
LF	Low Frequency
HF	High Frequency
UHF	Ultra High Frequency
PRNG	Pseudo Random Number Generator
SC1	Secret Code part 1
SC2	Secret Code part 2
SC3	Secret Code part 3
T->R	Tag to Reader
R->T	Reader to Tag
EAN	European Article Number
UCC	Uniform Commercial Code
EDL	Enhanced Driving Licenses
TID	Tag Identification
LFSR	Linear Feedback Shift Register
PUF	Physically Unclonable Function
PDA	Personal Digital Assistant
IEC	International Electrotechnical Commission
XOR	Exclusive OR
PRN	Pseudo Random Number
CRC	Cyclic Redundancy Check

PRNG	Pseudo Random Number Generator
AIA	Abstract of Integer Arithmetic
PIN	Personal Identification Number
Cur.	Current
Prev.	Previous
SC	Secret Code
AIA	Abstract of Integer Arithmetic

ACKNOWLEDGEMENTS

I am sincerely grateful to my supervisor Dr. Srinivas Sampalli, who provided his support and encouragement during my entire thesis. His patience and knowledge has allowed me to think outside the box.

This project could not be possible without the financial support from Industry Canada. I would also like to thank Dalhousie University, Faculty of Graduate Studies and Faculty of Computer Science for providing me with an opportunity to work in a great environment and computer lab.

I am grateful to Dr. Nur Zincir-Heywood and Dr. Denis Riordan, who have taken the time to read my thesis and provided me with helpful comments and suggestions.

All members of WISE RFID team have shared their knowledge and participated in group meetings to provide feedback on my work. Thanks to all WISE RFID team members for their support.

My family and friends made my life cheerful during this experience. My beloved husband, Kuntal Shah has always provided me with continuous encouragement and guidance.

CHAPTER 1 INTRODUCTION

Radio Frequency Identification (RFID) is the technology to identify objects by radio waves. RFID ideas and applications date back to 1939. It was used to identify World War II aircrafts [2]. The modern RFID that we see in applications was patented in 1983 [1, 62]. RFID is an emerging technology under ubiquitous computing which covers a wide range of different applications. From supply chain management to government issued IDs such as passports and currency notes, asset tracking to security and access control, RFID tags are driving the networking revolution.

RFID systems consist of tiny integrated circuits equipped with antennas (RFID tags) that communicate with their reading devices (RFID readers) using electromagnetic fields at one of the several standard radio frequencies. The reader is connected with the middleware which is responsible for converting the raw tag data into meaningful information. It is also responsible for resolving reader collisions and for ensuring that multiple reads or bad reads are eliminated. The middleware is connected with a back-end system which basically handles information coming from the reader. It is used to perform the assigned task depending on the application.

When an RFID tag passes through the field of a reader antenna, it detects the activation signal. This activates the RFID chip, and transmits the information on its microchip to be picked up by the reader antenna. Figure 1.1 shows a typical RFID system. The scanning antenna sends out radio frequency (RF) signals in a relatively short range. This RF radiation does two things. First it provides the passive RFID tag with the energy to communicate. Second, it provides a carrier for communicating with the RFID tag [4].

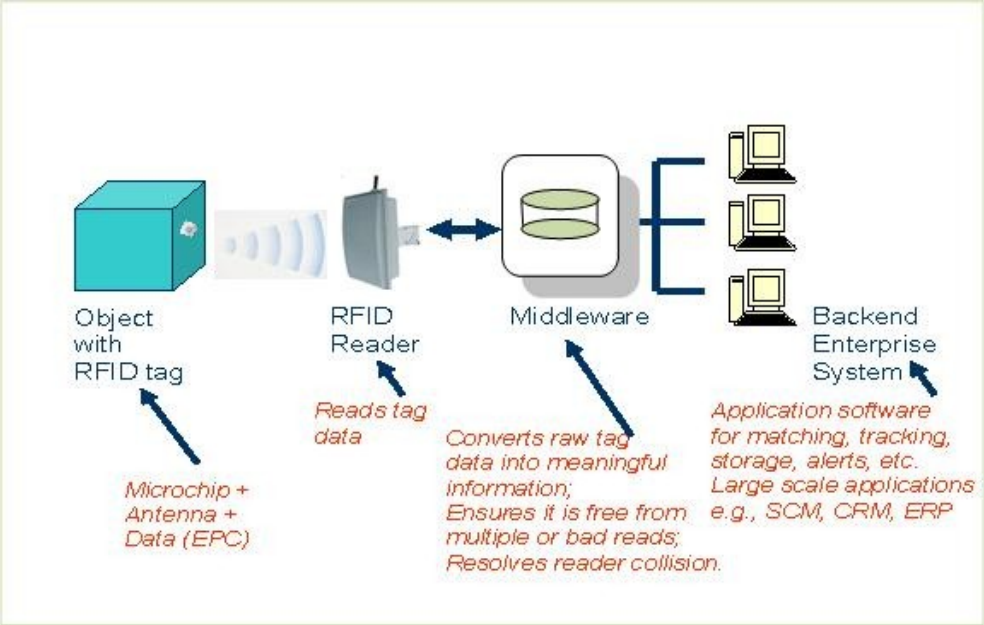

Figure 1.1 RFID Basic Components [4]

Many authentication and security protocols and cryptographic solutions have been deployed to make this process secure and trustworthy. But due to many weaknesses and vulnerabilities, different types of attacks can be launched on this system. The tag, reader, air and back-end system are the main areas where attacks can be launched. The attacks such as unauthorized data modification, eavesdropping, tag cloning and Denial-of-service (DoS) are some of the main risks [4]. Security solutions for other systems such as mobile devices, embedded systems, firewall and virtual private networks also cannot be used directly due to limitation on the tag side. Due to applications and usage in passport, access control, currency notes and other sensitive areas, it is necessary to develop an RFID system which is resistant to all major attacks [14,18]. Many solutions have been proposed to mitigate attacks. But still attackers find loopholes or perform reverse engineering to launch these attacks. To illustrate, let's take an example of unauthorized tag reading. The usual process for tag reading is when a tag comes in the vicinity of the reader; it sends signal back to the reader with Electronic Product Code (EPC) which tells weather a tag is valid or not. Due to low computational power on the tag, there are few efficient methods to detect whether the reader is legitimate or not. In this situation, an

unauthorized reader can come in the vicinity of the tag and read the information from the tag. Once the attacker has enough information from the legitimate tag, he/she can clone the tag and use this tag in malicious activities.

Tag cloning is one of the biggest challenges in RFID which leads to counterfeiting of products or to gain access to the secure areas using cloned access cards. An attacker can replicate the tags to insert the counterfeit products into the supply chain. Memory constraints on passive RFID tags prevent us from implementing highly secure encryption algorithms.

The goal of this research is to develop a lightweight protocol to prevent tag cloning. It uses a combination of frequency hopping, secret key for a 3-way handshake and a dynamic fake ID. This prevents cloning by not providing or revealing tag information to the unauthorized reader. While this protocol mainly aims to prevent the cloning attack, it also helps in preventing other attacks such as the kill command attack, eavesdropping/unauthorized reading, unauthorized data modification, de-synchronization attack and replay attack. Simulation results of the protocol indicate that the protocol prevents cloning by preventing unauthorized reading of the tag.

The rest of the thesis is organized as follows.

Chapter 2 of this report presents details on RFID, attacks on RFID, and previous work done on RFID cloning prevention and detection. Chapter 3 presents working of the protocol. Chapter 4 presents analysis of the protocol and simulation results of different attack scenarios. Chapter 5 provides concluding remarks.

CHAPTER 2 BACKGROUND

2.1 RFID COMPONENTS AND TYPES

RFID tags can be passive, active or battery-assisted passive. Passive tags are energized from the electromagnetic field generated by the reader, while active tags are equipped with a battery to communicate with the reader. Semi-passive or battery-assisted passive tags are equipped with a battery which is used only to run microchip but it communicates by harvesting energy from electromagnetic field generated by a reader [56].

There are many standards developed for RFID which drive the architecture and working fundamentals for each type. International Organization for Standards (ISO) and EPCGlobal™ are the two leading organizations which develop industry standards. It is worth noting that within North America most of the research is done on EPCGlobal™ standards [58]. EPCGlobal™ was an effort by the Auto ID Centre which was initially started from researchers at MIT.

While ISO has many standards, some of the notable standards mentioned by many researchers include 14443, 11784, 11785, 15693 and 18000. These standards also have variations depending on the release. On the other side EPCGlobal™ standards divide tags into classes [59]. Table 2.1 shows class structure by EPCGlobal™.

Table 2.1 EPC Class Structure [5]

Class	Details
Class 0	Read Only Passive Tags
Class 1	Write Once Read Many Passive Tags
Class 1 Gen. 2	Multiple Read/Write
Class 2	Re-writeable Passive Tags

Class	Details
Class 3	Semi Passive Tags
Class 4	Active Tags
Class 5	Readers

It is important to note that the most popular tag is Class 1 Gen. 2 tags due to its acceptance within ISO standards. ISO Standard 18000-6c is the same as EPC Class 1 Gen2 tag. For each standard, there is a different frequency associated with it. Passive tags usually work on 125 kHz (LF), 860-960MHz (UHF) and 13.56 MHz (HF). On the other hand, active tags work on 433 MHz (UHF) and 2.45 GHz (Microwave). 915 MHz (UHF) can be used for either passive or active tags [6].

A typical Passive RFID tag of EPC Class 1 Gen 2 has a memory of 2048 bits. RFID tag is divided into three major components, namely, the RF module, logic circuits and memory. The RF module consists of the power regulator, rectifier, voltage reference module and backscattering module. Rectifier and voltage reference are used to generate power from the radio signals which are controlled by the power regulator. Backscattering is the component to reply to the reader [6].

Logic circuits comprise of clock extraction and demodulator. Clock extraction extracts time from the reader and the demodulator converts signals from the reader. The main part of memory contains three sections. There are many standards in EPC and EPC 96 is one of them. First part of EPC 96 consists of the header, EPC manager, object class and serial number. The header contains European Article Number - Uniform Commercial Code (EAN.UCC) scheme of 8 bits which can be either European or North American standards for partition. The EPC Manager is unique to each manufacturer (28 bits) and object class is unique to the object manufacturer (24 bits). The serial number is used to identify the item uniquely, which is given highest 36 bits. The second part of EPC 96 scheme contains 224 bits, which are used for status control, kill command and information about read only cells. The third part contains 1728 bits, which is divided into four memory

banks for efficiency. These four memory banks are customizable by the user. Within this space there can also be a field of TID which is the tag ID given by the user [6].

2.2 RFID ATTACKS

Figure 2 outlines different types of attacks on each component such as on a tag, wireless medium, back-end database, side channel and reader. The type of attacks on the tag and the wireless medium are more than the others. Tables 2.2 – 2.6 provide basic descriptions of these attacks.

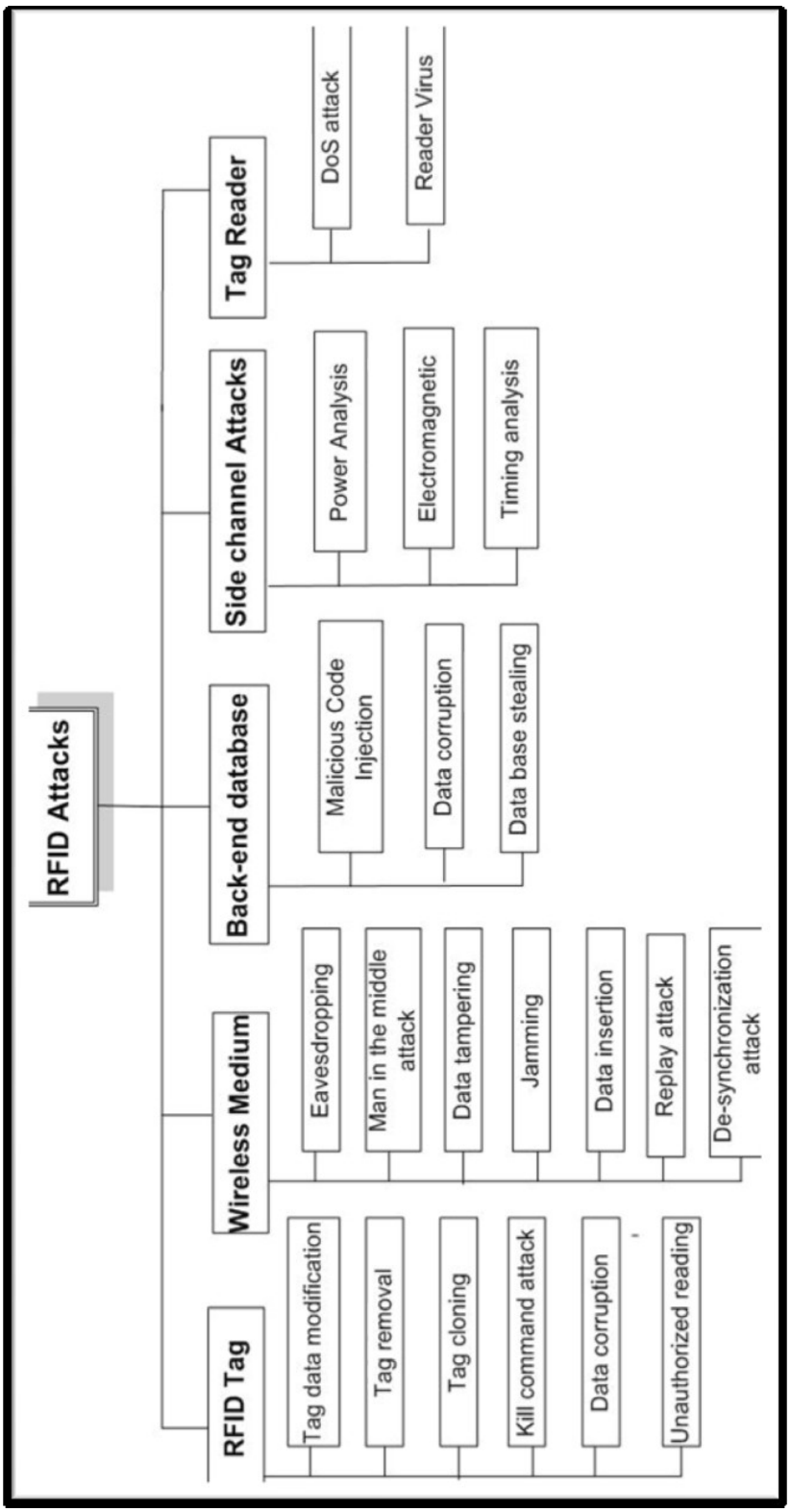

Figure 2.1 Classification of Threats and Vulnerabilities in RFID [8, 21]

Table 2.2 List of Attacks on a Tag

Attack	Description
Tag Data Modification	Data on the tag is replaced with other meaningful data in order to break data integrity.
Tag Removal	Physical removal or destruction of a tag.
Tag Cloning	Create an exact replica of a tag.
Kill Command Attack	Disable a tag by issuing kill command.
Data Corruption	Modify data on the tag such that the information is corrupt.
Unauthorized Tag Reading	Data on the tag is read by an unauthorized reader.

Table 2.3 List of Attacks on a Reader

Attack	Description
DoS on Reader	Bring down the reader performance by many requests.
Reader Virus	A virus that is stored on a tag and triggers when the reader reads the tag. Virus replicates itself, modifies the reader settings and eventually modifies the data in the SQL back-end [7].

Table 2.4 List of Attacks in a Wireless Medium

Attack	Description
Man-in-the-middle Attack	Impersonate an RFID tag to the reader or impersonate a reader to the RFID tag.
Eavesdropping	Capture a wireless transmission between the tag and reader.
De-synchronization Attack	Destroy the synchronization between the tag and the reader such that they cannot communicate further.
Signal Jamming	Generating an electromagnetic signal in the same range as the reader in order to prevent tags from communicating with readers.
Replay Attack	Collect data during eavesdropping and use the same data to communicate with the reader.
Data Tampering	Modify tag data.

Table 2.5 List of Attacks on a Back-End Database

Attack	Description
SQL injection/Malware injection	A virus or worm attacks on the back-end database which may compromise information [55].
Back-end Data Corruption	Change values in the back-end database such that information is no longer usable.

Attack	Description
Database Stealing	An intruder steals the data stored on a back-end database. Make an unauthorized image of the database to attack in future.

Table 2.6 List of Side-Channel Attacks

Attack	Description
Power Analysis Attack	An attacker observes changes in a power consumption and tries to relate it with the internal state of a crypto-system [57].
Timing Attack	Extract information, based on the variations in the rate of computation of a target device [57].

One of the biggest areas of crime is government issued IDs. In the United States, passports and enhanced driving licenses (EDL) now contain an implanted RFID chip. Australia and Sweden are also using passports with RFID [14]. Australia introduced its ePassport system in October 2005, but due to security issues they now issue new ePassports beginning in May 2009 with improved security features. This new N series ePassport contains holder's digitized photograph, name, gender, date of birth, nationality, passport number and passport expiry date which are core to anyone's identity [13]. Britain introduced its biometric passport for further security measures, but in a really short span computer hackers were able to crack it for exploitation [15,16,17].

Researchers tested the United States' passport card and two Washington State EDLs. In their findings the first discovery was stated as "EDLs do not carry tag-unique or even

system-unique TIDs, but instead bear generic manufacturer codes" [18]. They also mentioned that government departments are working to include this in the future. Tag Identifier (TID) is a unique identifier to prevent cloning in EPC tags. Lack of TID or inappropriate assignment of TID may be susceptible to other attacks including data modification. Another important finding was that these tags (EPC Gen 2) are possible to read/write as far as 50 meters under ideal conditions and as low as 2 meters. The protective sleeve is the only mechanism to prevent this [18, 20].

Many companies are now putting RFID on parts to identify the components included in that machine. Boeing and Airbus are two of the frontiers in this business. There are around three million parts in Boeing 777 and an RFID tag on each part allows them to scan for missing parts and part history such as how old the part is, was it refurbished or was it used on another plane and other related information. Any tag data tampering or unauthorized modification can prove to be dangerous to many lives associated with that machine [19].

Even with the replacement of barcode systems with an RFID into areas such as transportation businesses, billing systems, customs and border protection systems, still rely heavily on the information retrieved from the tags. If an intruder changes the product code, it can have many implications on billing and customs duties [12].

Tag has an access password, which allows only an authorized person to change information on the tag. This access password is not big enough to sustain against today's password breaking programs. It is very easy to perform a simple brute force attack and get the access password. Once the attacker has this information, he/she can modify the information on the tag and perform a data modification attack or data integrity failure. The same thing can happen with the kill password. The kill password is also not big enough which allows it to be easily broken, disabling the tag forever. The other kind of attack is a DoS attack in which the attacker can perform a simple signal jamming and interrupt the communication. Several more attacks are possible on existing RFID systems

like a de-synchronization attack, eavesdropping and man-in-the-middle. An RFID virus has also been developed [7].

Due to the limitation of computational power, complex cryptographic algorithm and other security protocols are not possible to be implemented on the tag. Use of light weight functions and providing security solution to all of these attacks is a real challenge for researchers around the world.

2.3 CLONING DETECTION AND PREVENTION METHODS

When the information on a valid tag is replicated to another tag, it creates a fake identity. This attack is referred to as a cloning attack. Cloning attacks are common in financial transactions, government IDs and expensive object tracking. Classic example is MiFare Classis Crypto-1 tag protocol. It is a stream cipher based technology used in about 200 million RFID chips worldwide [11]. Crypto-1 authentication protocol is a mixture of a random number generator, 48-bit linear feedback shift register (LFSR) and an algebraic calculator, which provides a high algebraic immunity. Crypto 1 cipher has a limit of 48 bit computation. However, researchers soon discovered that high algebraic immunity does not help much as there were already many direct algebraic attacks in existence. Researchers found that Crypto-1 was not sufficiently effective for very long [11]. This section provides explanation of different approaches for cloning detection and prevention mechanisms developed by researchers [8].

Most cloning prevention methods are based on two techniques. The first technique uses a mutual authentication handshake between the tag and the reader before the reader can retrieve the EPC data from the tag. Commonly used handshaking procedure involves a challenge text sent by the reader to which the tag responds by encrypting the challenge text using a shared secret key and sending it back to the reader. Thus the reader verifies the authenticity of the tag. Some systems use a bidirectional authentication procedure in

which the reader is also authenticated by the tag. The second technique uses a cryptographic protocol in which messages between the tag and the reader are encrypted using hash functions and secret keys shared between the tag and the reader.

The following is the set of techniques available in the literature. It should be noted that many tag cloning prevention methods also inherently present methods to detect cloning.

Deckard [22] is based on anomaly-based system to detect tag cloning. The idea is to monitor users' behavior on a daily basis and compare it with current activity in real time. Since each individual's behavior is different and almost unique we can build a different profile for each user. The RFID tag's audit record is used to build a profile for normal behavior, which can be used to determine when there is a significant change or deviation in normal behavior. When an anomalous behavior is detected, it triggers an alarm and flags a security breach.

Kim and Kim [23] propose an application level anti-counterfeiting solution to track a product during its life cycle. Their proposal uses mobile RFID devices like mobile phones and PDAs equipped with RFID readers to continuously track and trace the product. This allows easy detection of counterfeit products in distribution channels. The authors suggest that addition of a watermarking scheme will be beneficial.

Lehtonen [24,25] proposes a cloning detection method that includes probabilistic analysis and historic traces of a tag to achieve the goal. The paper defines traces as the set of events for a single tag obtained throughout the tracking system. The assumption is that if there is a cloned tag then it should have its traces in the genuine tag traces. If that is the case then the author's hypothesis suggests that machine learning techniques can be used to understand abnormal behavior of cloned traces.

Ilic *et al* [26] propose a synchronous secrets method for anti-counterfeiting with cloned tags. Due to the cost factor, cryptographic tags are not a solution for the supply chain. The synchronous secret method detects de-synchronization, thus preventing further

distribution of counterfeit products. The product goes through multiple steps from the manufacturer to distributors to hospitals and they are authenticated throughout the process. All readers are connected to a back-end server that keeps information about secret keys for all tags. All genuine tags are scanned at least once at every phase in supply chain. This system is based on a web service that uses a synchronized secret. Each RFID tag has a secret key Kx that is stored on tag as well as on the back-end database. On every reader a new random secret key Kx+1 is generated and which is again stored on tag and the back-end database.

Danev *et al* [27] proposes the use of physical layer fingerprints of RFID transponders for cloning detection. The paper proposes a method based on spectral features of response signal and modulation shape generated by transponders for in and out reader signals. The authors have tested it on fifty RFID smart cards and found uniqueness of patterns in the physical layer fingerprints.

Since RFID tags have limited hardware, Bolotny and Robins [28] propose a physically unclonable function (PUF)-based approach that provides hardware support for privacy and security. PUFs also supply exponential number of keys for tags, which is a viable solution to key distribution problem. PUF tags are also part of cloning prevention as tags have hardware applications of access control and authenticity verification.

Yamamoto *et al* [29] propose a method around the standard of logical memory map of ISO/IEC 18000-6 type C. An RFID tag's memory is typically divided into four memory banks. The idea proposed in this method is based on a scheme called the bank level locking scheme. In this scheme, the idea is to lock down one of the sections in the memory bank. The authors propose the idea of recording write journal into a specially designed area of the memory where only the tag itself can write. Readers can only read that area but cannot write. Whenever there is any write operation on the tag, it writes it into the write journal part of the tag write only area. This allows tracking of tampering; if a malicious user tries to tamper with the data then it will be written into the tag write only

part and upon reading by a genuine reader it will be retrieved. Middleware will read the write journal and detect any tampering if done.

Han and Chao-Hsein [30] propose a digital watermarking scheme. Digital watermarking in multimedia is a method of putting information on the document, picture or video such that it can be protected from manipulation and duplication. This technology is promising for RFID tamper detection. This method of watermarking is used in two things. One is tamper detection in RFID tags and second is tamper detection in the RFID data stream. The method proposed in the paper has some unique features. For example, the embedded watermark is imperceptible. It also prevents illegal embedding and verification; so only an authorized person who has a key can embed, extract and verify watermarks. Furthermore, it uses blind verification in which the original unmarked data should not be required for watermark verification. Watermarks can also help in getting information like the location at which it is altered. It also helps in detecting which kind of alteration has been done.

In addition to the above detection methods, there are some excellent survey papers in the literature [31-39]. Wasnioski [33] suggests an ideal intrusion detection system structure where signature based or anomaly based architecture can be dispatched. It comprises of data stream filled with tags that are read by sensors or the readers and fed into a database system. The user application will then either automatically or manually query the database to analyze any results or security purpose interrogation.

The technical report by Koscher *et al* [35] is dedicated to a survey of Class 1 Gen 2 EPC tag cloning. At the time of publishing the paper they expected Class 1 Gen 2 tags to be a worldwide success on passports and driver licenses and some risk analysis was made. They show that cloning of government issued IDs can pose a risk to the entire EPCGlobal^{TM} system.

Lehtonen et al's survey [36] shows that there is no silver bullet solution that can be applied to every application. It has compared many approaches but at the end it indicates

that authentication needs to be specific to the application area. Also, it indicates that further research is needed in the area of offline authentication and network issues.

With focus on man-in-the-middle attack, the paper by Anderson [37] argues that even though security is a coordinated effort, cracking is usually done by several intruders and the weakest link is exploited. It analyzes vulnerabilities in large-scale infrastructures where many security aspects come together.

This white paper [38] is a report by US border alliance that has an excellent classification showing which issues are addressed and which issues are still on the table. Important conclusions indicate that random ID numbers do not protect location privacy but addresses data privacy and increase confidence in privacy protection. Another surprising result is that physical shielding was found to address a number of concerns.

The Smart Card Alliance is a non-profit, multi-industry association working towards smart card technology adoption and use. The white paper by the alliance [39] includes many aspects of industry use but at the end it seems to point to hardware improvement in the card and encryption technology with shorter read range for addressing privacy concerns.

Lehtonen et al [53] propose methods to detect tag cloning after a tag has been cloned and introduced in the system. It relies on a simple method to raise an alarm if two tags having the same identity (secret key or serial no, etc) have been introduced in the system.

Dimitriou [40] proposes a simple protocol using one-way hash function and random session identifiers. It uses standard cryptographic hash function. Hash value of the tag ID is used for authentication which is refreshed periodically.

Chiu and Li [41] introduce an improved scheme for pedigree. It uses a re-writable tag with three cells which are used to store hashed value of barcode and signatures. It works in combination with 2D barcodes and prevents large scale counterfeits.

Konidala et al [42] propose tag-reader mutual authentication scheme. Pad generation function is used which takes two 16-bit random numbers each, from the tag and the manufacturer, and utilizes the access and kill passwords, to generate two 16-bit Pads. Here both tag and reader can generate same values as access and kill password is shared between tag and reader which are used to XOR the two 16-bit chunks (APwdM, APwdL).

Choi et al [44] propose an anti-cloning protocol in accordance with EPCGlobal^{TM} Class 1 Gen 2 tags. A unique serial number, encrypted EPC, access password, kill password, 32-bit PRNG, 16-bit CRC, 128-bit symmetric encryption scheme, XOR, concatenate function ad extraction function are used to make a guessing attack much harder. In the proposed scheme, a tag does not encrypt an EPC but stores an encrypted EPC which allows for only a few XOR or PRNG operations so that it can even be applicable to passive tags.

The paper by Kwak et al [45] proposes a light-weight and secure authentication protocol based on a random number generator and abstract of integer arithmetic (AIA), which generates a secret key pool from the subset of the remainders and the carries of the integer multiplication. While requiring only 82 bits of RAM, 20 bits of ROM and 300-400 logic gates, the proposed protocol can satisfy security requirements for RFID system while requiring a small amount of resources.

Abawajy [46] proposes a tag authentication scheme called TagAuth which is used to prevent tag cloning. This authentication mechanism relies on the high security and computation at the back-end side and allows for less computation in the tag itself. It uses tag EPC ID, a secret key stored in tag and a virtual ID. This virtual tag ID is only communicated to the reader. The back-end server decrypts the virtual tag to get the valid EPC ID.

Duc et al [34, 47] introduce synchronization based protocol to protect against cloning. The protocol uses simple functions such as PRNG and checksum codes or CRCs to add

more security to the message being sent by tag to the reader. This scheme uses EPC ID, the PIN and a secret key to generate the message sent to the reader. The protocol has only one round of communication which reduces computational and communicational burden on RFID tag.

Mitra [49] presents a new encryption technique against tracking and cloning. The tag rotates pseudonyms from time to time in order to respond to a query. An authentic reader would share the pseudonyms in advance such that it would be the only reader to track it consistently. If an attacker reader is querying constantly then reader would either slow down the response or it will go to sleep. However, these techniques would require lot of calculation for an RFID chip.

Tuyls and Batina [50] propose a protocol for RFID cloning prevention using Physical Unclonable Functions. It uses off-line authentication method in combination with this. A Physical Unclonable Function is a function that maps challenges to responses and that is embodied in a physical object.

Juels [52] proposes a technique which protects basic cloning attacks or skimming attacks. The method uses PIN to perform reader-to-tag authentication. This paper mostly concerns with identifying a valid reader rather than the tag.

Over the time, many researchers came up with different ideas to improve RFID security. Due to physical constrain of the tag memory and processing power, many of them are not possible to implement at this stage with the current hardware available. Many papers discuss about using cryptographic functions to achieve security, which is not possible to implement on the current passive RFID tags. Active tags on the other hand carries more memory and battery power, which make them more secure then the passive tags but price difference between the two is vast which makes passive tags more economical to use in supply chain and many other areas.

CHAPTER 3 PROPOSED PROTOCOL

Tag cloning is one of the biggest challenges in RFID. It leads to the counterfeiting of products or an unauthorized access to the secure areas using cloned access cards. An attacker can replicate tags to insert the counterfeit products into the supply chain. Memory constrains on the passive RFID tags prevent researchers from implementing the secure encryption algorithms.

Chien [60] classifies RFID security protocols in four categories, full fledged (symmetric encryption, cryptographic one-way function, public key algorithms), simple (random number generator, one-way hashing function on tags), light weight (random number generator, cyclic redundancy check (CRC) but not hash function) and ultra light weight (simple bitwise operations like XOR, AND, OR on tags). The proposed protocol presented here uses pseudo random number generator and basic bitwise operations, therefore it falls under the light weight RFID security protocol.

The main idea of the protocol is tag authenticating the reader before providing tag information. It uses a combination of frequency hopping, secret key for 3-way handshake and a fake ID. This 3-way handshake is used to confirm the identity of the tag and the reader to each other. This is achieved via a random symmetric secret key, pseudo random number generator and synchronization between the tag and the reader. In addition, the protocol is using frequency hopping which uses a random number generator to make it more secure. Frequency hopping may help in preventing eavesdropping to some extent. The tag ID is never sent in an open medium. The tag and the reader communicate using a fake ID. The fake ID changes after every transaction to avoid vulnerability.

The protocol assumes that the tag and the reader are physically secure. Hardware modifications may be required on the tag to implement the protocol. This protocol is designed with keeping UHF tags in mind, which has a wider frequency band. The average breakdown of this band is made of 50 channels, which are 500 kHz apart [54]. Current RFID systems use a spread spectrum system. The coded sequence is used to

switch from one frequency to another in a fixed interval. In a near term, this spreading looks random but in a long run distribution, it looks evenly distributed over the hop set [54]. In this protocol instead of setting this frequency list on the tag and reader side, it is generated using pseudo random number generator on the tag and the reader side as needed. This makes difficult for intruders to guess the next frequency.

3.1 NOTATIONS

The tag and the reader have a database which stores information about a tag. Tables 3.1 and 3.2 show the information stored on the tag and the reader side.

Table 3.1 Tag Database with an Example Entry

Tag ID	Incomplete	Cur. Fake ID	Prev. Fake ID	Cur. Frequency	Prev. Frequency	Cur. SC
21545	0	101	405	15	26	1356

Table 3.2 Reader Database with an Example Entry

Tag ID	Incomplete	Cur. Fake ID	Prev. Fake ID	Cur. Frequency	Prev. Frequency	Cur. SC	Prev. SC
21545	0	101	405	15	26	1356	12
68454	1	254	123	3	30	125	3025

Tag ID

A tag ID is the real ID of a tag. The real ID is never sent during communication. This way it is protected from an intruder. It can also be used to distinguish between a real tag and a cloned tag.

Incomplete

Incomplete is a flag value which represents the status of the last communication. It can have values of 0, 1 or 2. Zero represents that the last communication was successful and start the new communication as a normal protocol scenario. One represents that the last communication was incomplete. Hence, use an error recovery mechanism in the next communication. Two represents the last two communications were failed and takes steps accordingly.

Current Fake ID

In this protocol, a real tag ID is never sent in an open medium. It uses a fake ID during communication. A fake ID is a randomly generated tag ID on both the tag and the reader side. The reader has a database of the real tag ID and the associated fake ID.

Previous Fake ID

A previous fake ID is a fake ID used in the previous communication. It is used in certain attack recovery scenarios and certain data loss recovery scenarios.

Current frequency

A Current frequency is the most recent frequency used in the communication. It is stored in the database to generate the next frequency using PRNG.

Previous Frequency

A previous frequency is a frequency used before the current frequency. It is stored in the database and used in certain de-synchronization scenarios.

Current SC

A secret code is used to authenticate the tag and the reader. The protocol performs a 3-way handshake to verify that the tag is talking with the correct reader and the reader is talking with the correct tag. A secret code is divided in three parts *SC1*, *SC2* and *SC3* which is used in different steps of an authentication process.

Previous SC

A previous SC is the secret code generated before the current SC. It is stored in the database and it is used in some error recovery scenarios.

Seed Values

In addition to the above fields, the tag and the reader also store pre-shared values of the seed to generate a new frequency and a new SC.

3.2 PROTOCOL OVERVIEW

The protocol has three main steps: the first is an initial communication, the second is a 3-way handshake and the third is the end of communication. In the initial phase, the tag sends a *fake ID* to the reader upon receiving the *hello* signal from the reader. In the second phase the 3-way handshake has 2 secrets; the first is secret code and the second is a random frequency both generated using PRNG. Both PRNG functions are independent and may have different seed values and length. The tag and the reader authenticate each other before sharing information. This method can also prevent eavesdropping because every message is sent on a different frequency. For an intruder it is difficult to guess the next frequency. In the same way, unauthorized data modification and kill command are also hard to execute. To overcome the de-synchronization attack, protocol stores the recent frequency. If the tag or the reader is not getting a response on the expected frequency then it looks for the next or last frequency and adjusts the frequency to match the other side. In the last phase of the communication the tag and the reader updates the value for the *fake ID*.

F0 is a pre-defined frequency on which all tags and reader start communicating. In figure 3.1, *Fi+1* is a new frequency generated using the PRNG function. Hence, +1 is a next PRNG cycle. The tag and the reader share the same seed. Reader has a database with seed values of all the tags. Tag does not send the seed or the original tag ID in the air while communicating. The reader and tag authorize each other by validating the fake ID, secret code and frequency.

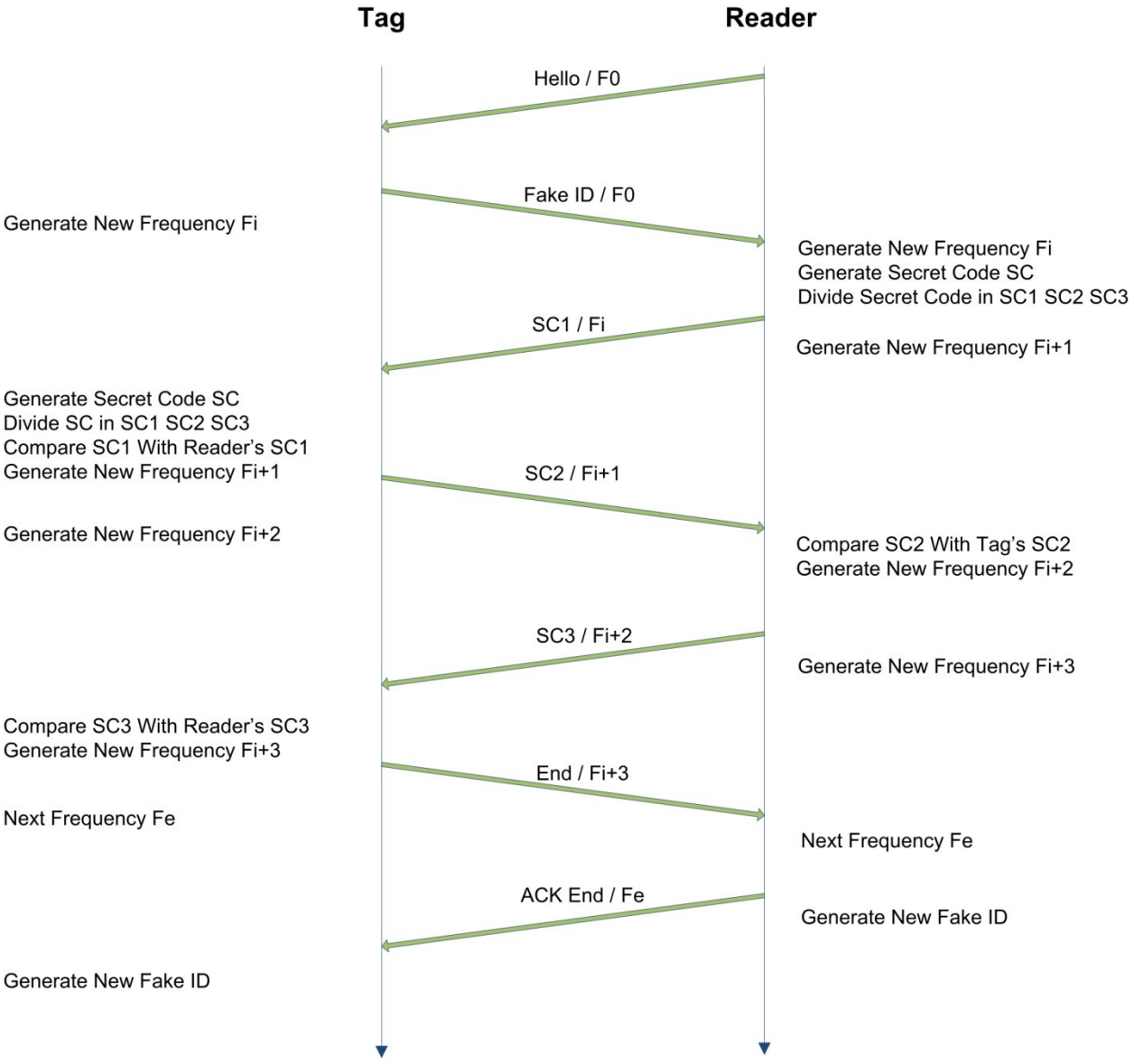

Figure 3.1 Successful Communication Scenario

The reader periodically sends a *hello* message on frequency *F0*. When a tag comes in the vicinity of a reader range, it gets a *hello* signal on the frequency *F0*. Upon receiving the *hello* signal, the tag sends the *fake ID* on frequency *F0* to the reader. After receiving the *fake ID*, the reader checks in the database for the *fake ID* and based on the seed values associated with the *fake ID*, it generates a new frequency *Fi* and a new secret code using PRNG. Secret code is divided into *SC1*, *SC2* and *SC3*. On the tag side it generates frequency *Fi* using PRNG. The tag expects a message on the new frequency *Fi*. When it gets a message on frequency *Fi*, it is part one of the secret code. The tag generates the secret code using PRNG and divides it in three parts *SC1*, *SC2* and *SC3*. The tag compares *SC1* received on *Fi* with the *SC1* generated using PRNG. If it matches then the tag generates the next frequency *Fi+1* and sends the second part of the secret code *SC2* on frequency *Fi+1* to the reader. On the reader side after sending *SC1* on *Fi*, it generates the next frequency *Fi+1* using PRNG. When it receives a message on *Fi+1* from the tag, it compares it with *SC2*. If it matches then the reader validates the tag as a genuine tag and generates the next frequency *Fi+2* using PRNG and sends part three of secret code *SC3* on frequency *Fi+2* to the tag. The tag generates frequency *Fi+2* using PRNG and when it receives the message from the reader on frequency *Fi+2*, it compares it with its *SC3*. If it matches then the 3-way mutual authentication is complete. The tag and the reader know that they are communicating with the valid tag and the valid reader. After the 3-way mutual authentication, the tag and reader generates the next frequency *Fi+3*. The tag sends an *END* signal to the reader on frequency *Fi+3*. Once the reader receives the *END* signal on frequency *Fi+3*, it sends *ACK END* to the tag on frequency *Fe* where *Fe* is a pre-defined frequency. After sending this message, the reader updates the *fake ID* of the tag and updates the database. Once the tag receives the *ACK END* signal, it updates its *fake ID* for the next communication and updates its database. Here data related to the tag is stored in the reader database. Each *fake ID* is a pointer to the reader database. This way the tag does not have to store any data. Data is secure on the reader side as high security can be implemented on the reader side and the tag is without any data so it is less vulnerable to some attacks.

Initialization of communication part:

1. Reader sends *hello* message on frequency *F0* while waiting to talk to any tag
2. Tag responds with its *fake ID* on frequency *F0*

Three-way handshake part:

3. Reader looks up *fake ID* in the database and generates the next frequency *Fi* using PRNG
4. Reader also generates Secret Code (*SC*) using PRNG
5. Reader divides *SC* in three parts *SC1*, *SC2*, *SC3*
6. Reader sends *SC1* on new frequency *Fi*
7. On the tag side, after sending *fake ID* on *F0*, tag generates new frequency *Fi* using PRNG
8. When the tag receives *SC1* on *Fi* from reader, tag generates *SC* using PRNG and divides it in *SC1*, *SC2*, *SC3*
9. Tag compares *SC1* received from reader with its *SC1*
10. If comparison of *SC1* is successful, tag generates new frequency *Fi+1* and sends *SC2* on *Fi+1* to the reader
11. Tag generates next frequency *Fi+2* using PRNG for the next communication
12. On the reader side, after sending *SC1* on *Fi*, reader generates new frequency *Fi+1*
13. When the reader receives *SC2* on *Fi+1*, reader matches *SC2* with its *SC2*
14. If *SC2* comparison is successful, reader generates new frequency *Fi+2* using PRNG and sends *SC3* on *Fi+2*. At this point reader has verified that tag is genuine
15. Reader generates new frequency *Fi+3* for the next communication
16. Tag receives *SC3* on *Fi+2* and tag compares *SC3* with its *SC3*
17. If *SC3* comparison is successful, tag has successfully verified the reader
18. Tag generates new frequency *Fi+3* and sends *END* signal to the reader

19. Reader receives *END* signal and sends *ACK END* signal on *Fe* to the tag to acknowledge that the communication has been successfully over on reader side and tag has been authorized.
20. Tag receives *ACK END* signal on *Fe* and ends the communication.

Preparation for next transaction:

21. Tag and reader generates new *fake ID* for the next communication
22. Tag and reader updates *incomplete* to zero for successful communication

3.3 DETAILED WORKING OF THE PROTOCOL

Figure 3.2 shows detailed working scenario of the protocol. Other than performing core steps mentioned above, the protocol checks for many other things like the status of *incomplete*, *previous frequency*, *fake ID* and *SC*. It performs core steps plus error recovery steps when needed.

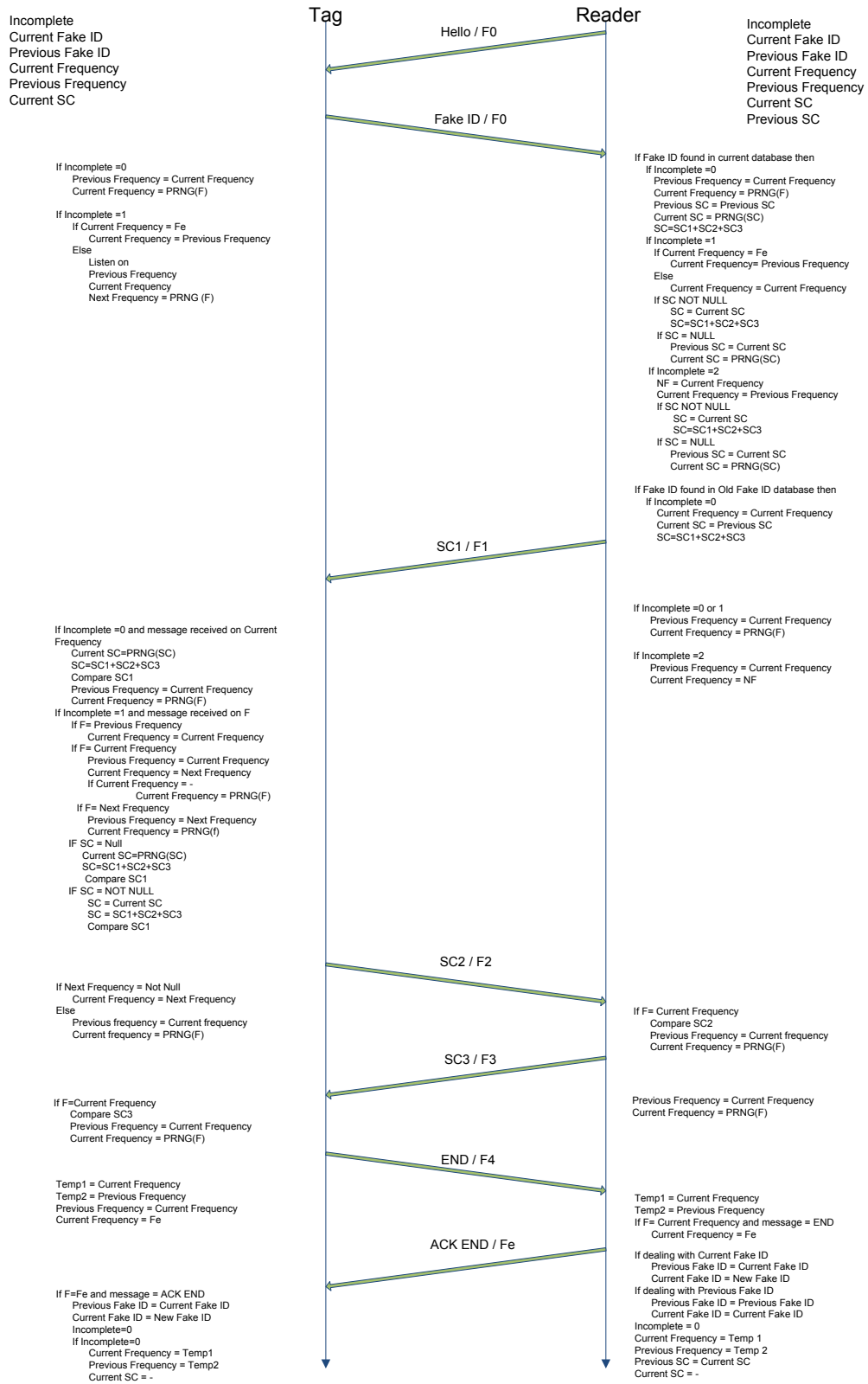

Figure 3.2 Detail Working Protocol

3.3.1 TAG SIDE ALGORITHM

The tag table stores information such as *incomplete*, *current fake ID*, *previous fake ID*, *current frequency*, *previous frequency* and *current SC*.

PART 1 - Receive Hello on F0

When the tag comes in the vicinity of a reader, it gets power from the electromagnetic field generated by the reader. To start the communication, tag sends *fake ID* on frequency *F0* to the reader. Here *F0* is a pre-defined frequency to start the communication.

PART 2 – After Sending Fake ID on F0

```
If Incomplete =0
      Previous Frequency = Current Frequency
      Current Frequency = PRNG(F)
```

The value of *incomplete* tells whether the previous communication was successful or not. A value 0 represents a successful and 1 represents an unsuccessful. If the previous communication was successful, then the protocol runs without an error recovery mechanism. The value of a *current frequency* is assigned to the *previous frequency* and a new *current frequency* is generated using PRNG.

```
If Incomplete =1
      If Current Frequency = Fe
            Current Frequency = Previous Frequency
      Else
            Listen on
            Previous Frequency
            Current Frequency
            Next Frequency = PRNG (F)
```

If the previous communication was unsuccessful then the tag follows an error recovery mechanism. To synchronize the frequency with the reader, the tag checks for the value of *current frequency*. If the value of a *current frequency* stored on the tag is *Fe* then it simply assigns the *previous frequency* to the *current frequency*. If the value of a *current frequency* stored on the tag is not *Fe*, then the last communication was lost between any steps before generating *Fe*. In this case, the tag looks for a response from the reader on the *previous frequency*, *current frequency* and *next frequency*.

PART 3 – After Receiving SC1 on F1

```
If Incomplete =0 and message received on Current Frequency
      Current SC=PRNG(SC)
      SC=SC1+SC2+SC3
      Compare SC1
      Previous Frequency = Current Frequency
      Current Frequency = PRNG(F)
```

When a tag receives the first part of the secret code on the current frequency *F1*, it checks the status for incomplete. If the value of *incomplete* is zero, then the tag generates a secret code using PRNG and divides it into three parts. The tag performs comparison between the reader's *SC1* and its own *SC1*. At this point the tag has verified two things: 1) the frequency is correct and 2) the part one of the secret code is correct. The tag then generates a new frequency using PRNG for the next step.

```
If Incomplete =1 and message received on F
      If F= Previous Frequency
     Current Frequency = Current Frequency
      If F= Current Frequency
            Previous Frequency = Current Frequency
            Current Frequency = Next Frequency
            If Next Frequency = -
         Current Frequency = PRNG(F)
      If F= Next Frequency
```

```
            Previous Frequency = Next Frequency
            Current Frequency = PRNG(f)

      IF SC = Null
            Current SC=PRNG(SC)
            SC=SC1+SC2+SC3
            Compare SC1
      IF SC = NOT NULL
             SC = Current SC
             SC = SC1+SC2+SC3
             Compare SC1
```

If the previous communication with the reader was unsuccessful then the tag follows the error recovery part. Depending on which frequency the tag gets response from the reader, it adjusts its frequencies. If the tag gets response from the reader on a *previous frequency*, then the tag does not generate any new frequency and uses *current frequency* from the tag table as a *current frequency*. If the tag gets response on a *current frequency* then it assigns the *current frequency* to the *previous frequency* and the *current frequency* becomes the *next frequency*. If the tag gets a response from the reader on a *next frequency* then the *previous frequency* becomes a *next frequency* and the *current frequency* becomes a new frequency generated using PRNG. This is needed to synchronize the frequency between the tag and the reader. Once the frequency synchronizes with the reader frequency, the tag compares the part one of the secret code. If the secret code is null then it generates a secret code and then compares.

PART 4 – After sending SC2 on F2

```
If Next Frequency = Not Null
      Current Frequency = Next Frequency
Else
      Previous Frequency = Current Frequency
      Current Frequency = PRNG(F)
```

After sending the second part of the secret code to the reader, the tag assigns a new value to the *current frequency*. If the *next frequency* was generated in the previous steps then the *next frequency* becomes the *current frequency*. Otherwise a new *current frequency* is generated using PRNG.

PART 5 – After Receiving SC3 on F3

```
If F=Current Frequency
      Compare SC3
      Previous Frequency = Current Frequency
      Current Frequency = PRNG(F)
```

Once the tag gets a message on the *current frequency*, it compares with the third part of the secret code. If that matches, then the tag generates a new frequency using PRNG to send the next message. At this point a 3-way mutual authentication is complete and the tag has verified the reader and the reader has verified the tag.

PART 6 – After Sending END on F4

```
Temp1 = Current Frequency
Temp2 = Previous Frequency
Previous Frequency = Current Frequency
Current Frequency = Fe
```

The tag sends *END* signal to the reader and assigns frequency *Fe* to the *current frequency*. *Temp1* and *temp2* variables stores a *current frequency* and a *previous frequency*, which are used later to update the tag table.

PART 7 – After receiving ACK END on Fe

```
If F=Fe and message = ACK END
      Previous Fake ID = Current Fake ID
```

```
Current Fake ID = New Fake ID
Incomplete=0
If Incomplete=0
      Current Frequency = Temp1
      Previous Frequency = Temp2
Current SC = -
```

Once the tag receives an *ACK END* signal on the frequency *Fe* from the reader, communication is over and the tag updates the tag table. It updates values for *incomplete*, *current frequency*, *previous frequency* and *SC*. The *current fake ID* is assigned to the *previous fake ID* and a *new fake ID* is generated for the next communication.

3.3.2 READER SIDE ALGORITHM

The values for *incomplete*, *current fake ID*, *previous fake ID*, *current frequency*, *previous frequency*, *current SC* and *previous SC* are stored in the reader table.

PART 1 - Send Hello on F0

The reader periodically sends a hello signal on the frequency *F0*. This initial frequency *F0* is a predefined frequency which is used to start the communication with the tag. The reader waits for a reply from the tag. The reader does not perform any operation during this phase. If after certain time, the reader does not get response from any tag, it resends a *hello* signal.

PART 2 – After Receiving Fake ID on F0

```
If Fake ID found in current database then
    If Incomplete =0
       Previous Frequency = Current Frequency
       Current Frequency = PRNG(F)
```

```
   Previous SC = Previous SC
   Current SC = PRNG(SC)
   SC=SC1+SC2+SC3

If Incomplete =1
   If Current Frequency = Fe
         Current Frequency= Previous Frequency
   Else
         Current Frequency = Current Frequency
   If SC NOT NULL
         SC = Current SC
         SC=SC1+SC2+SC3
    If SC = NULL
         Previous SC = Current SC
         Current SC = PRNG(SC)

 If Incomplete =2
    NF = Current Frequency
    Current Frequency = Previous Frequency
    If SC NOT NULL
          SC = Current SC
          SC=SC1+SC2+SC3
    If SC = NULL
         Previous SC = Current SC
         Current SC = PRNG(SC)
```

When a reader gets any reply on frequency *F0*, it checks for the tag *fake ID* in the tag database. If the *fake ID* is found in the current database then the reader checks for status of *incomplete*. If the value of *incomplete* is 0, that means previous communication was successful. If *incomplete* is one, then the previous communication was unsuccessful. The value of *incomplete* 2 tells that the last two communications were failed.

If *incomplete* is 0 then the reader generates a next frequency and a secret code using PRNG. If *incomplete* is 1 and *current frequency* is *Fe* then the reader assigns a value of

the *previous frequency* to the *current frequency* else the reader keeps the *current frequency* the same. At this point if the reader table has a secret code value associated with the *fake ID*, it uses that secret code for this communication; otherwise generates a new secret code using PRNG.

If the value of *incomplete* is 2 then the *next frequency* becomes *current frequency* and *current frequency* becomes *previous frequency*. If the reader table has a secret code value associated with the *fake ID*, it uses that secret code for this communication; otherwise generates a new secret code using PRNG. An *incomplete* 2 is designed to overcome the de-synchronization attack and to overcome the scenario in which communication fails more than ones.

```
If Fake ID found in Old Fake ID database then

    If Incomplete =0
        Current Frequency = Current Frequency
        Current SC = Previous SC
        SC=SC1+SC2+SC3
```

When communication is dropped in the last step after the reader sends an *ACK END* signal to the tag, the reader updates *fake ID* of the tag and updates the reader table but on the tag side as an *ACK END* signal is not received, it does not update the *fake ID*. When the tag comes in contact with the reader again, the tag has an *old fake ID*. In this case, the reader always has *incomplete* value as 0. If the reader does not find the *fake ID* in the *current fake ID* database, it looks in a *previous fake ID* database. If the *fake ID* is found in the old database then reader keeps the *current frequency* the same and the *previous secret code* becomes a *current secret code*. This is done to synchronize with the tag.

PART 3 – After Sending SC1 on F1

```
If Incomplete =0 or 1
      Previous Frequency = Current Frequency
```

```
      Current Frequency = PRNG(F)

If Incomplete =2
      Previous Frequency = Current Frequency
      Current Frequency = NF
```

After sending the first part of the secret code to the tag, the reader generates a new frequency using PRNG provided incomplete value is 0 or 1. If incomplete value is 2 then the *previous frequency* becomes *current frequency* and the *current frequency* becomes *next frequency*.

PART 4 – After Receiving SC2 on F2

```
If F= Current Frequency
      Compare SC2
      Previous Frequency = Current Frequency
      Current Frequency = PRNG(F)
```

When the reader receives part two of the secret code, it compares with its *SC2*. If match is successful then generates a new frequency using PRNG.

PART 5 – After Sending SC3 on F3

```
Previous Frequency = Current Frequency
Current Frequency = PRNG(F)
```

After sending the part three of the secret code to the tag, the reader generates a new frequency using PRNG and waits for a reply from the tag on a new frequency.

PART 6 – After Receiving END on F4

```
Temp1 = Current Frequency
```

```
Temp2 = Previous Frequency
If F= Current Frequency and message = END
                Current Frequency = Fe
```

Temp1 and *Temp2* are the variables which are used to store the values for *current frequency* and *previous frequency*, which are used in the next step to update the reader table. If the reader receives a message *END* on the *current frequency*, the *current frequency* becomes *Fe*.

PART 7 – After Sending ACK END on Fe

```
If dealing with Current Fake ID
      Previous Fake ID = Current Fake ID
      Current Fake ID = New Fake ID

If dealing with Previous Fake ID
      Previous Fake ID = Previous Fake ID
     Current Fake ID = Current Fake ID

Incomplete = 0
Current Frequency = Temp 1
Previous Frequency = Temp 2
Previous SC = Current SC
Current SC = -
```

In the last step, if the *fake ID* is *current fake ID* then the reader generates a new *fake ID* for the next communication. If the *fake ID* is a *previous fake ID* then the reader does not generate a new *fake ID* and uses the same *current fake ID* in the next communication. The reader table values of *incomplete*, *current frequency*, *previous frequency*, *previous SC* and *current SC* are also updated at this point.

3.4 SIMULATION

Java based RFID simulation framework allows machines on the same network to communicate with each other as a reader or a tag. The framework contains genericsenderreceiver.java, reader.java, tag.java and taginfo.java files for basic networking functionality. The implementation allows a successful scenario between tag and the reader for anti-cloning protocol. MyReader.java, MyTag.java, PRNG.java, TagConsole.java and ReaderConsole.java are the files required for deploying the protocol along with the framework.

GenericSenderReceiver.java extends thread class and it is extended by reader.java and tag.java. Reader.java and tag.java uses objects from taginfo.java and PRNG.java. These files contain the basic framework on which computers communicate on the given network. Some of the beginning conditions are also included in reader.java and tag.java. Specifics of the protocol are implemented in myreader.java and mytag.java which extends reader.java and tag.java respectively. In our implementation, readers communicate through port number 5555 and tags communicate through port 4444. The machine that simulates reader runs ReaderConsole.java and machine simulating tag runs TagConsole.java. Since the protocol changes frequency for each message, frequency is represented as a number along with sending the secret code.

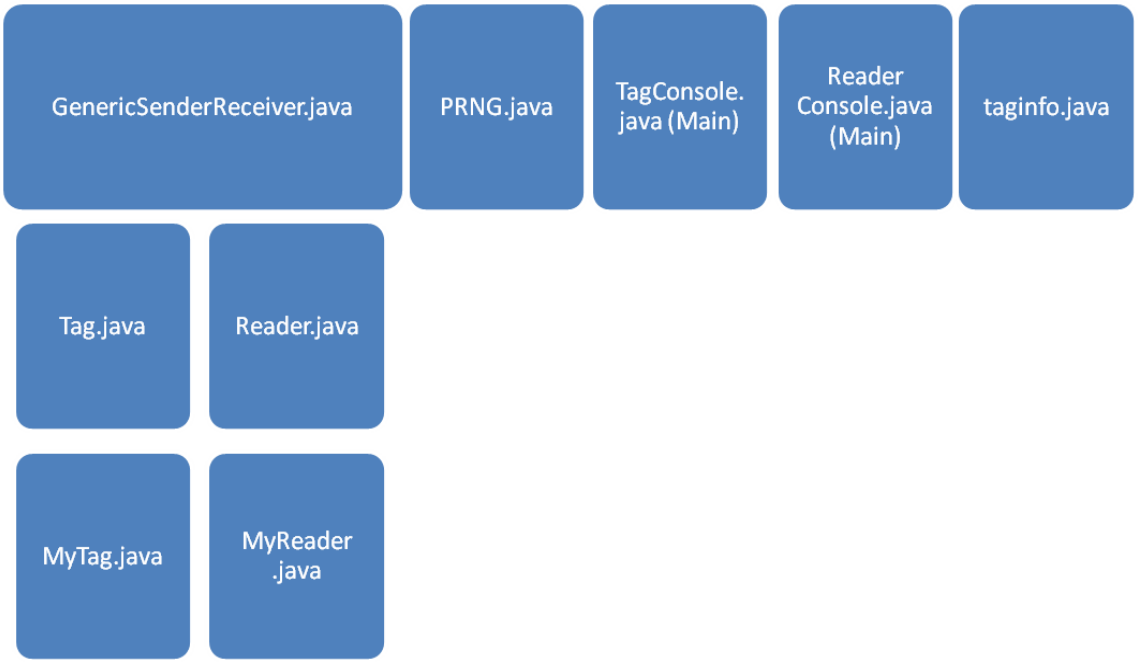

Figure 3.3 Java File Structure

Steps for deployment with description

1. Deploy the java files along with framework on two machines
2. Machine acting as reader runs ReaderConsole.java and other machine runs TagConsole.java
3. It is suggested to start process on the tag machine first and then the reader machine
4. When starting the reader machine it sends hello message to find which tags are active
5. It gives available tag's fake ID on the screen
6. On the reader console the command "auth:###" or "sel:###" (in this case tag ID is 101) starts the process for verifying tag
7. Secret code and frequencies are generated by the applicable PRNG functions from PRNG class and below transactions are done from MyReader.java and MyTag.java
8. Reader sends message number1/secret code part1—frequency number to the tag
9. Tag verifies the secret code and frequency. If correct, then respond with message

number2/secret code part2—frequency number to the reader

10. Reader matches secret code part 2 and frequency and confirms the tag identity
11. Reader sends message number3/secret code part3—frequency number to the tag
12. Tag matches secret code and frequency and verify reader identity
13. Tag sends END signal to the reader and reader replies with ACK END signal
14. Tag and reader updates their databases

3.5 TEST RUN

Environment 1
Reader Machine = Windows Vista (Net Beans IDE for Windows 6.9.1)
Tag Machine = Mac OS X 10.5 (Net Beans IDE for Mac 6.9.1)

Environment 2
Reader Machine = Windows XP (Net Beans IDE for Windows 6.9.1)
Tag Machine = Mac OS X 10.5 (Net Beans IDE for Mac 6.9.1)

Description of the output

In the figure 3.4 and 3.5, the very first line shows the IP address followed by the port number of the respective computers. The second bunch shows the values of all the variables used in the protocol at initial state. Assuming tag is already started, reader executes “taglist” command which sends hello message to all the tag machines on port 4444 and reply is recorded and displayed. In figure 3.4, we can see that the reader is showing the reply from tag 101. On the other side on the tag machine, we can see that “hello-reply: 101” is sent to our reader machine on IP 192.168.0.5 on port 5555. After this process, user or reader operator has to decide that which tag needs to be authenticated and “auth: 101” command is used to authenticate tag with fake ID of 101. On the reader side we can see the message 1/1011—6 is sent to the tag where 1 is authentication message number, 1011 is part one of SC and 6 is the frequency. In figure 3.5, tag shows

that *SC1* on both sides is matching. Similarly, messages 2, 3, 4 and 5 are sent and confirmation of matching is displayed on the other end. It is worth noting that *Fe* is a fixed frequency which is 512 in this case and it is managed with the help of temporary variables to restore back to previous state. End values of all the variables in figure 3.4 are displayed before restoration of temporary variables where as in figure 3.5 we can see the values after restoration. This way *Fe* is not part of the PRNG cycle and replaced in current frequency for temporary purposes.

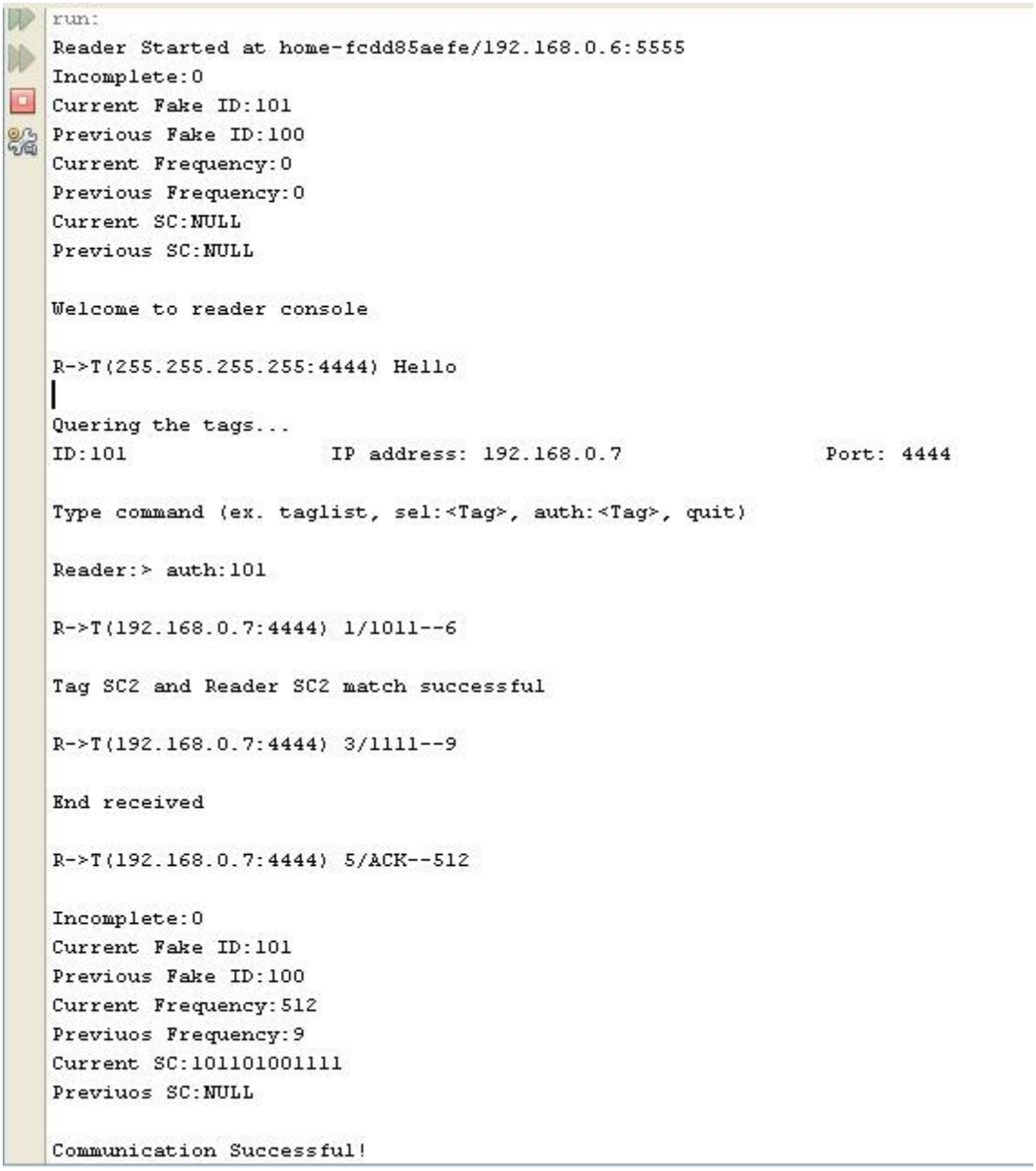

Figure 3.4 Reader Output

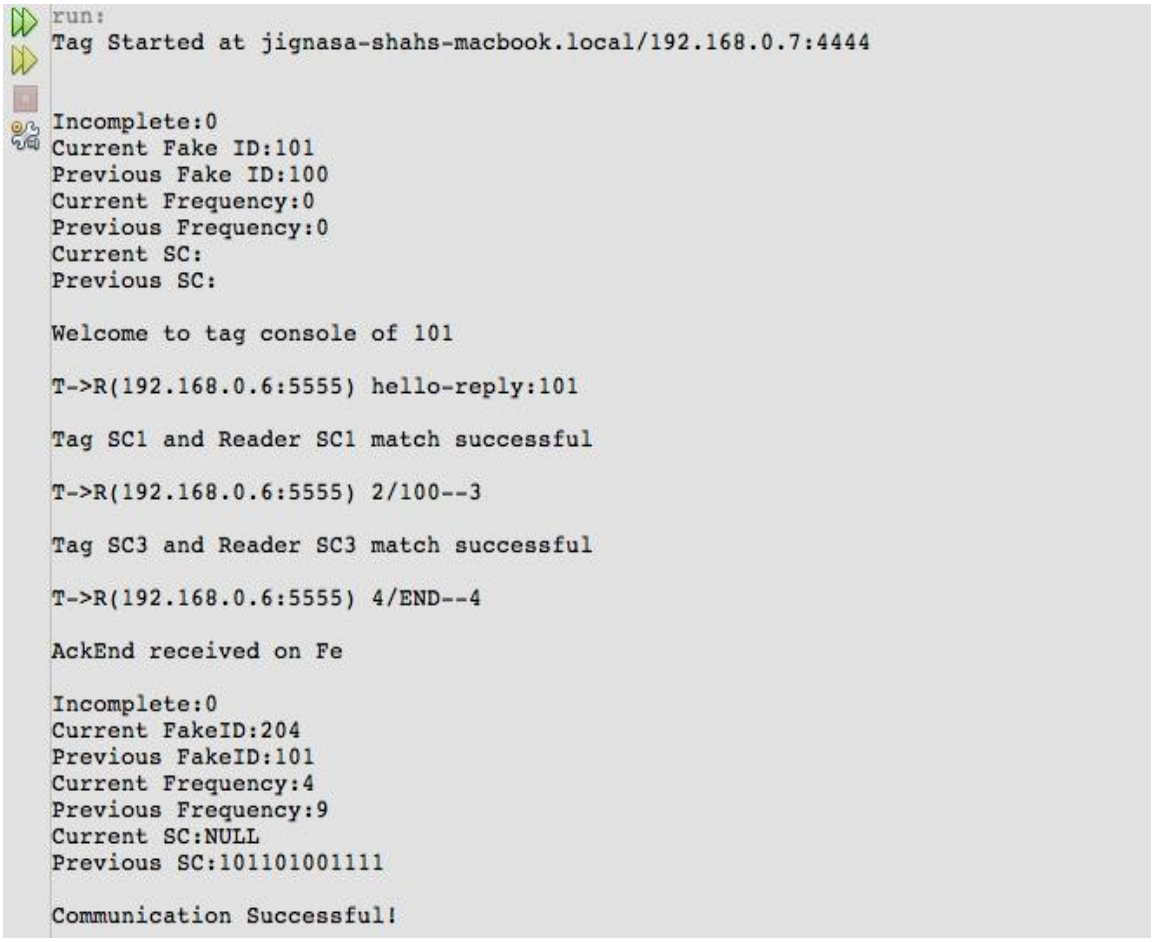

Figure 3.5 Tag Output

CHAPTER 4 ANALYSIS

An unauthorized duplication of a legitimate tag is called cloning attack. To perform cloning attack, an attacker reads a tag data with an unauthorized reader and gets information about the tag, which is further used to make cloned tag. The protocol proposed here is mainly designed to prevent a cloning attack but it also survives other attacks such as an unauthorized reading of the tag and reader, replay attack and de-synchronization attack. As mentioned in the detailed protocol diagram, the reader and tag performs steps for error recovery to recover from the attack already happened. Table 4.1 presents a list of attacks and their description. Later in this section, attack scenarios and error recovery scenarios are explained.

Table 4.1 Overview of all Attack Scenarios

Attack Type	Description
Data lost	Reader to tag *hello* on *F0* fails
	Tag to reader *fake ID* on *F0* fails
	Reader to tag *SC1* on *F1* fails
	Tag to reader *SC2* on *F2* fails
	Reader to tag *SC3* on *F3* fails
	Tag to reader *END* on *F4* fails
	Reader to tag *ACK END* on *Fe* fails
Unauthorized reading	An intruder tag tries to communicate with the reader
	An intruder reader tries to communicate with a genuine tag
Replay	An intruder captures data from a previous communication and launches replay attack with an intruder tag
	An intruder captures data from previous communication and launches replay attack with an intruder reader

Attack Type	Description
De-synchronization	An attacker tag tries to communicate with the reader more than one time to perform de-synchronization attack
	An attacker reader tries to communicate with the tag more than one time to perform de-synchronization attack
Man-in-the-middle	An intruder captures communication between the tag and reader and changes the data transmitted

4.1 DATA LOSS

Due to many reasons, a signal sent from a reader or a tag can be lost or unreachable. The tag goes out of the vicinity of the reader during communication or an obstacle between the tag and reader can also block the signal. Data loss is a very general scenario. Scenarios for data loss show how the protocol recovers from a de-synchronization of frequency and secret code.

Case 1: Reader to Tag Hello on F0 Fails

Type: Data lost while communicating
Detail: As shown in figure 4.1, the tag and reader does not change anything other than sending *hello* message and *fake ID*. Therefore, there is no problem if this signal is lost. The reader resends *hello* signal after timeout.

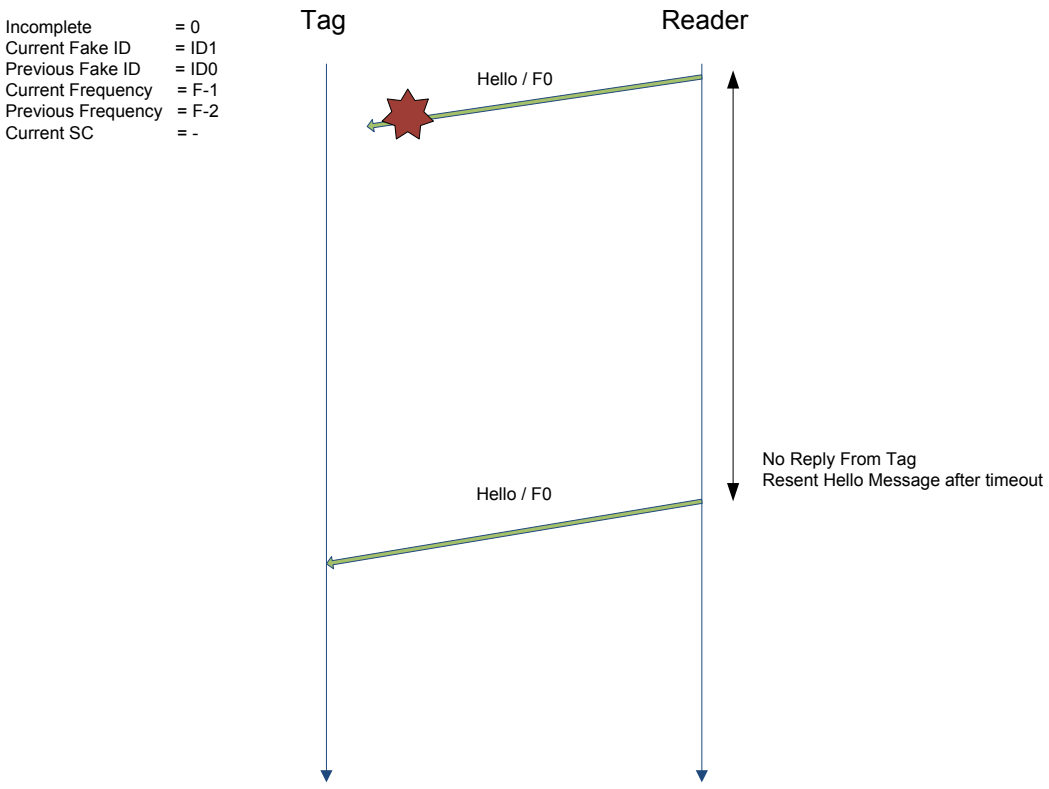

Figure 4.1 Hello Signal Loss Scenario

Case 2: Tag to Reader Fake ID on F0 Fails

Type: Data lost while communicating

Detail: As shown in figure 4.2, the first message from the tag to reader fails. The reader does not have any problem and it resends *hello* message after timeout. But on the tag side, it generates a new frequency *F1*. After time out on tag side, it updates the tag table. It now has *incomplete* as 1 and a *current frequency* as *F1*.

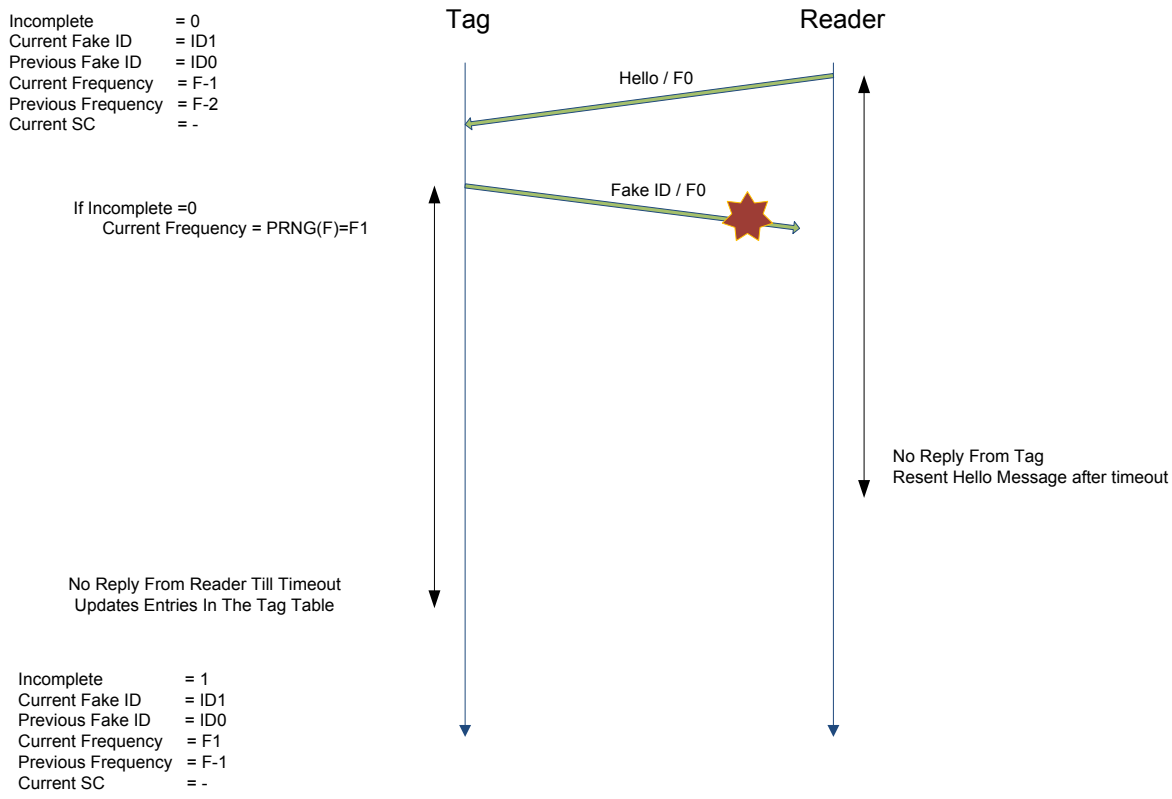

Figure 4.2 Fake ID on F0 Fails

When the tag comes in contact with the reader again, it follows the algorithm for *incomplete* as 1 as shown in figure 4.3. It synchronizes to the frequency of a reader. The tag waits for the response from a reader on three different frequencies; *previous frequency*, *current frequency* and *next frequency*. The tag adjusts its *current frequency* based on the response from the reader.

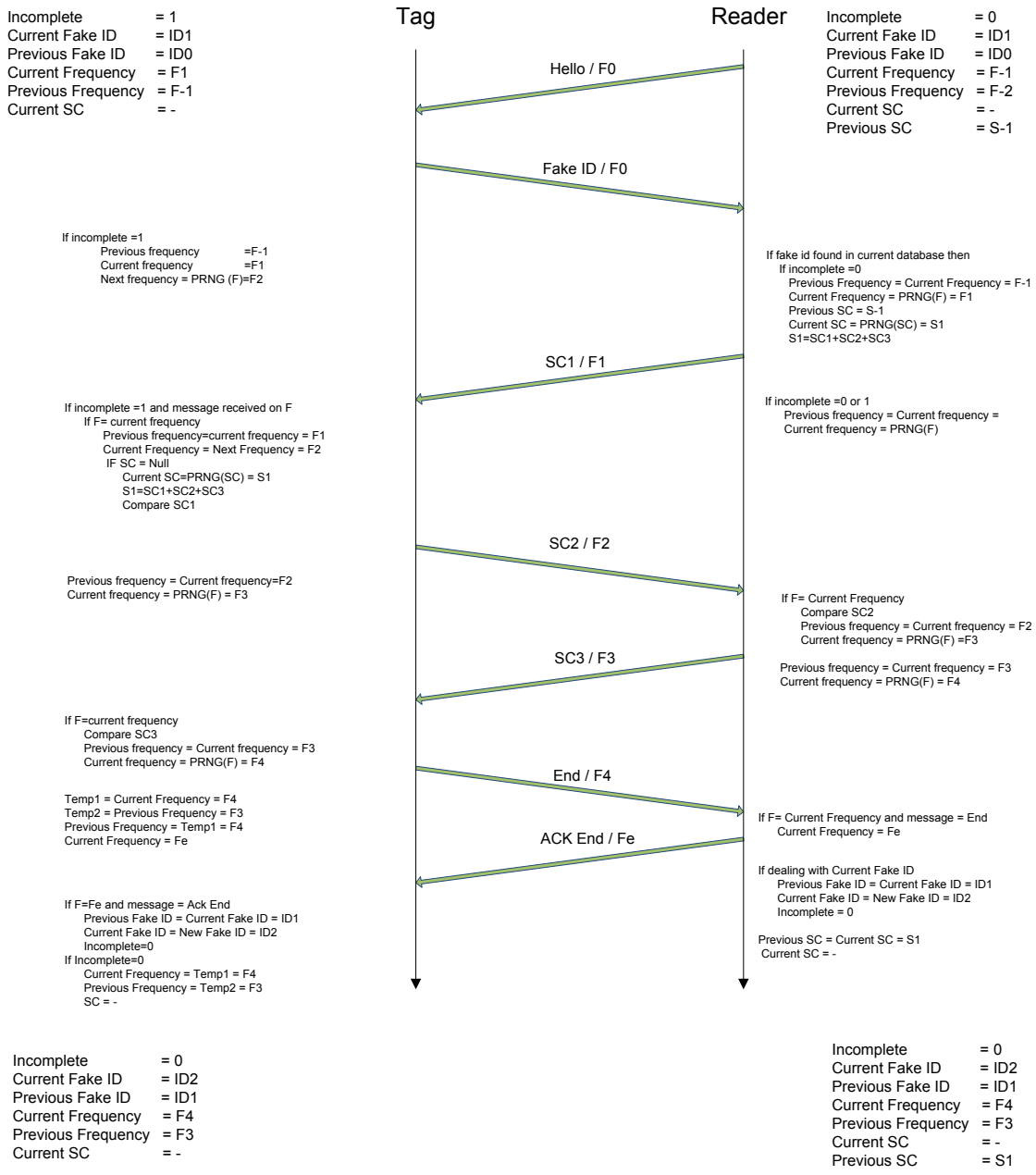

Figure 4.3 Recover Fake ID on F0 Fails

Case 3: Reader to Tag SC1 on F1 Fails

Type: Data lost while communicating

Detail: As shown in figure 4.4, a signal from the reader to a tag containing a secret code part 1 fails. After sending *SC1*, reader generates next frequency. Due to lost signal, reader does not get reply from the tag so it updates the reader table after time out. On the tag side due to lost signal, the tag does not receive a signal from the reader so it updates the tag table after timeout. Here the tag and the reader have different *current frequency*. The reader is ahead in generating next frequency. The reader has also generated a secret code but the tag has not.

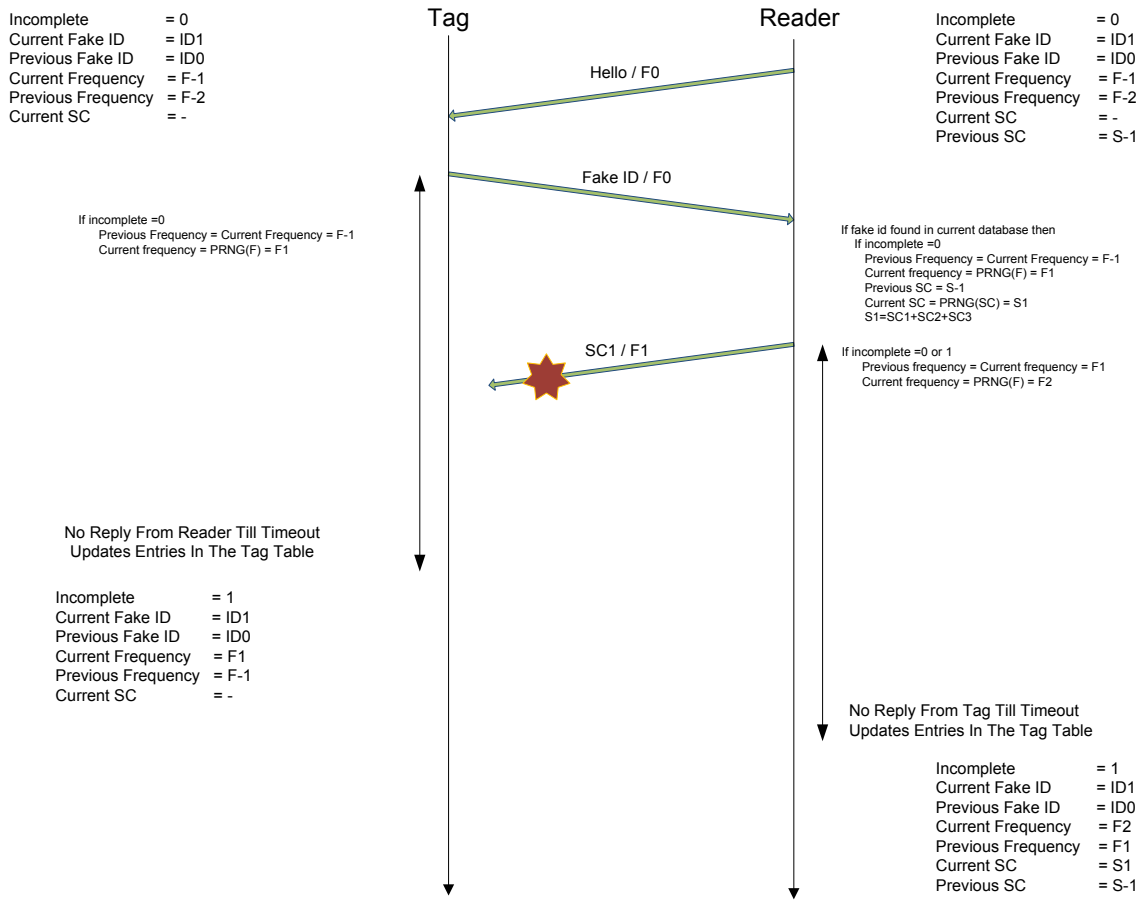

Figure 4.4 SC1 on F1 Fails

When the tag comes in contact with the reader again, the tag replies with the *hello* message with the tag *fake ID*. Here the last communication was incomplete, so the tag and reader has *incomplete* as one. Both follow the algorithm as shown in figure 4.5 for incomplete as one and as the tag is behind generating next frequency, it waits for a response from the reader on a *previous frequency*, *current frequency* and *next frequency*. In this case the tag gets response from the reader on *next frequency* because reader is ahead in generating frequency. The tag adjusts its frequency accordingly.

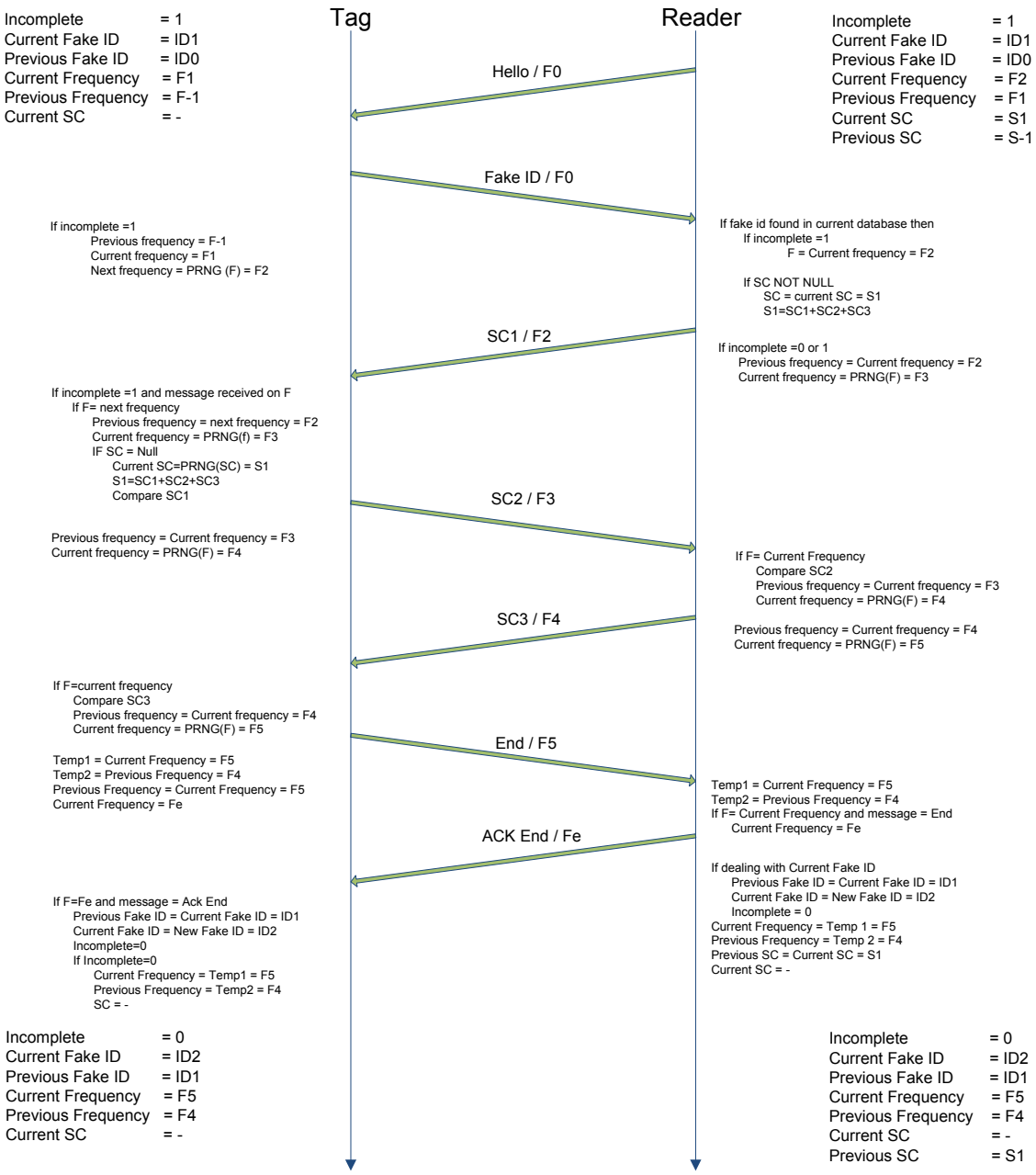

Figure 4.5 Recover SC1 on F1 Fails

Case 4: Tag to Reader SC2 on F2 Fails

Type: Data lost while communicating

Detail: The part one of a secret code is successful. As shown in figure 4.6, after sending *SC1* on frequency *F1*, the reader generates next frequency *F2*. The tag sends part two of the secret code to the reader and generates next frequency *F3*. The signal containing *SC2* from tag to reader fails. After timeout, the reader updates the reader table, which has current frequency as *F2*. When the tag does not receive a response from the reader till time out, the tag updates the tag table. In this case *current frequency* ends up at *F3* in the tag table.

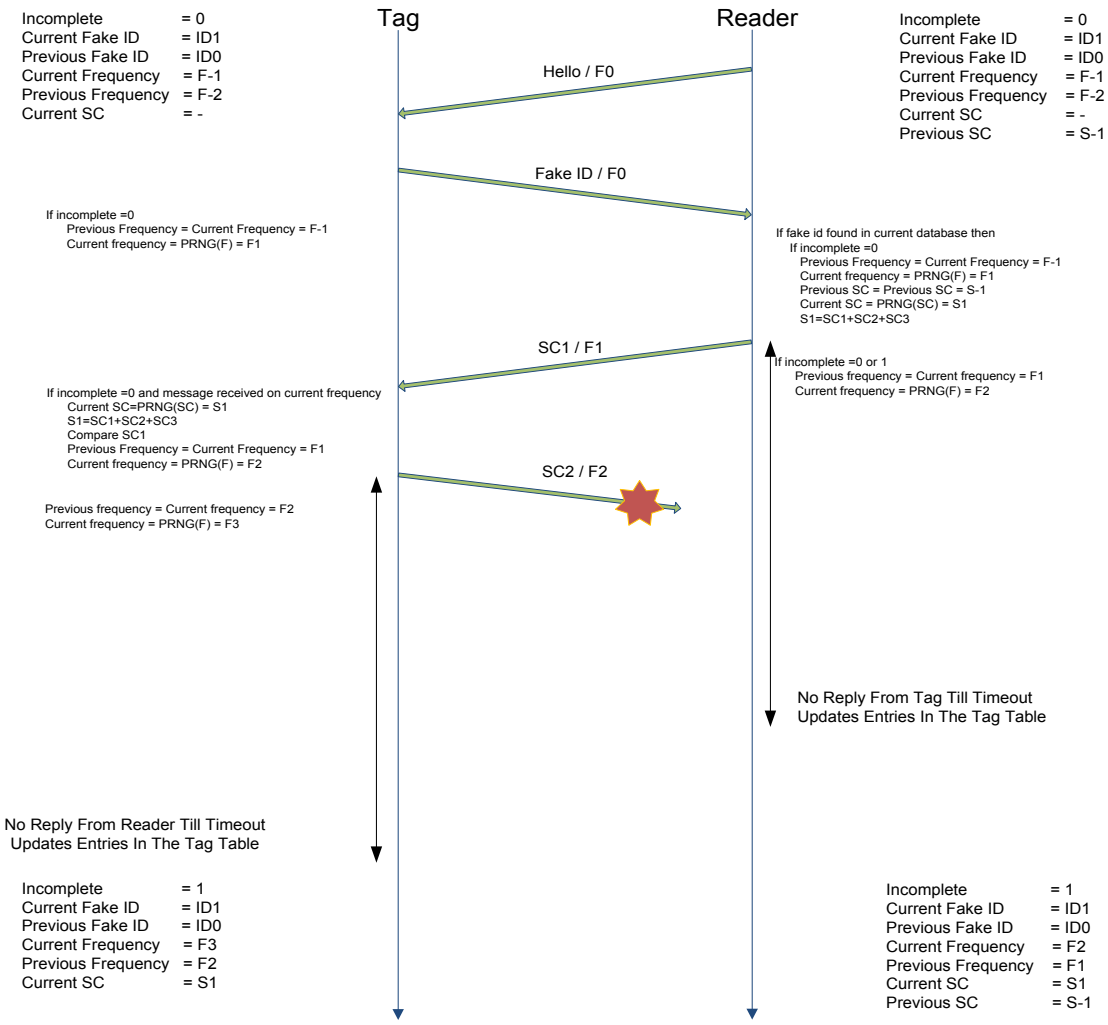

Figure 4.6 SC2 on F2 Fails

Here, the tag is one frequency ahead than the reader and *incomplete* value is 1. As shown in figure 4.7, after sending a *fake ID* to the reader, tag expects a response from *previous frequency*, *current frequency* and *next frequency*. Because the reader is one frequency behind, the tag receives response on a *previous frequency*. In this case the *current frequency* remains the same and the *next frequency* is stored temporarily for the next message.

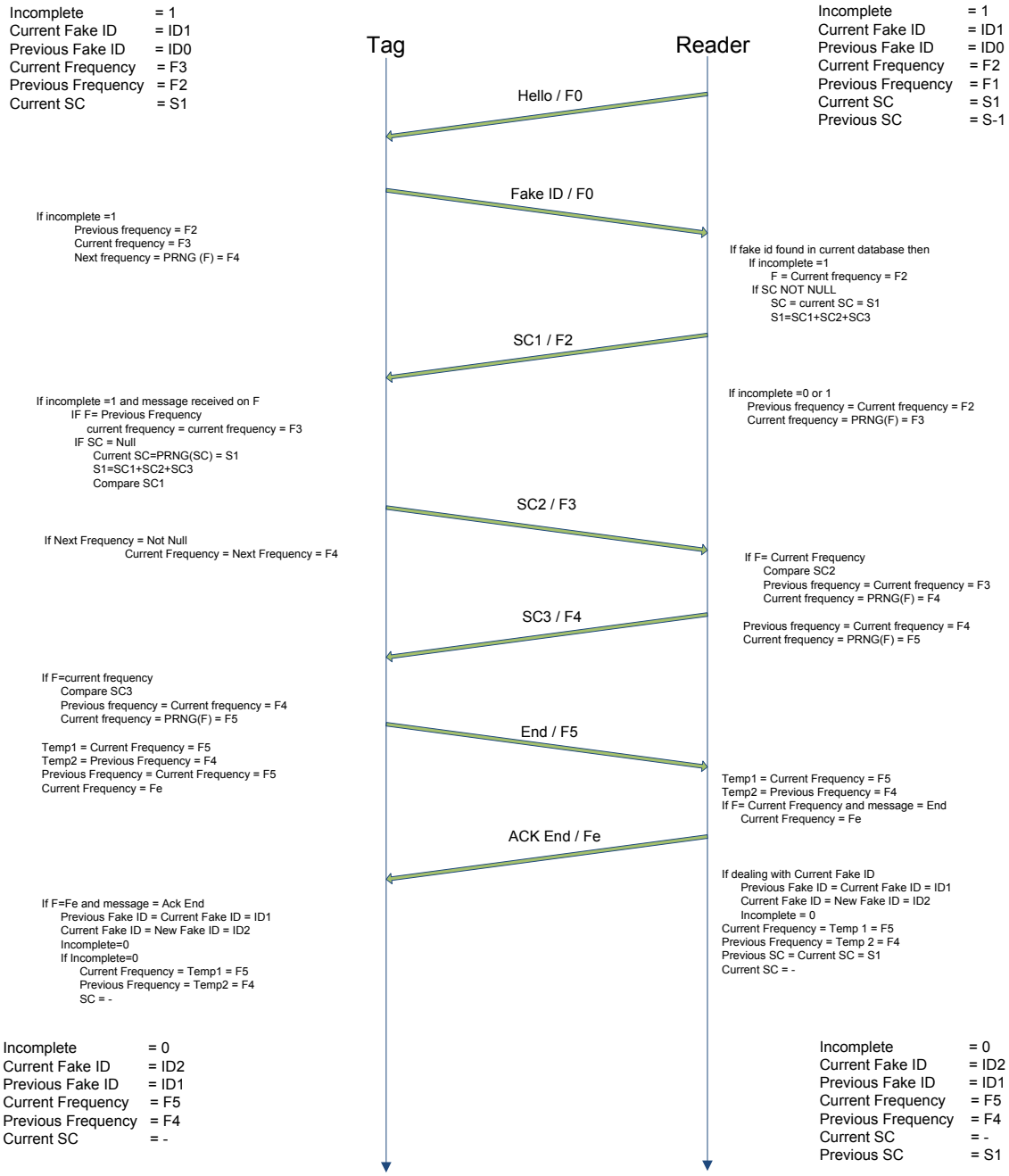

Figure 4.7 Recover SC2 on F2 Fails

Case 5: Reader to Tag SC3 on F3 Fails

Type: Data lost while communicating

In this scenario, after sending the secret code part two, the tag generates a new frequency. In this case, it is *F3* as shown in figure 4.8. On the reader side after sending the secret code part three, it generates a new frequency *F4*. This signal is lost in the communication and after time out the tag and reader update their tables.

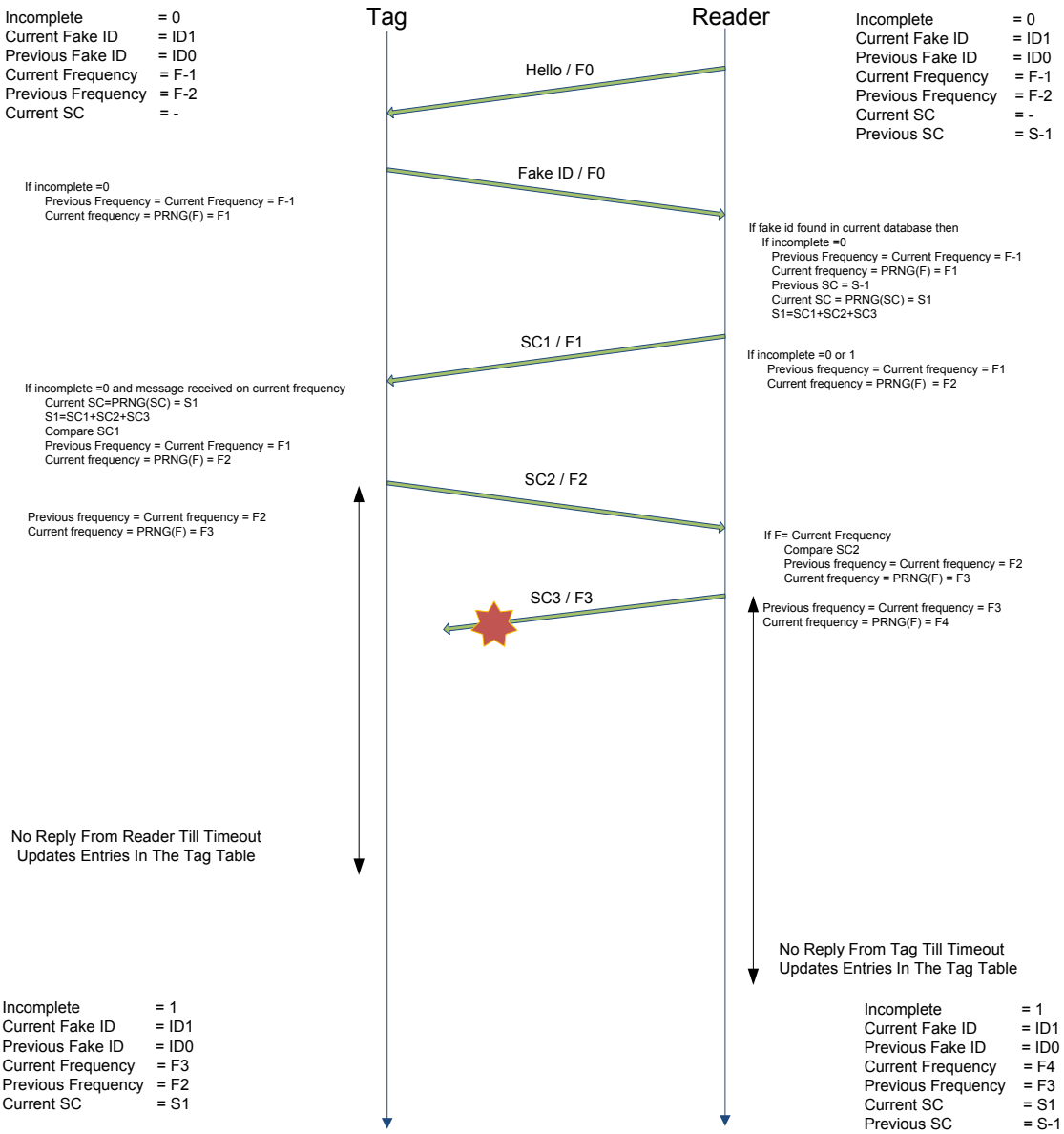

Figure 4.8 SC3 on F3 Fails

When the tag comes in contact with the reader, both have incomplete as one. The tag and the reader follow the protocol as shown in figure 4.9 to synchronize the frequency.

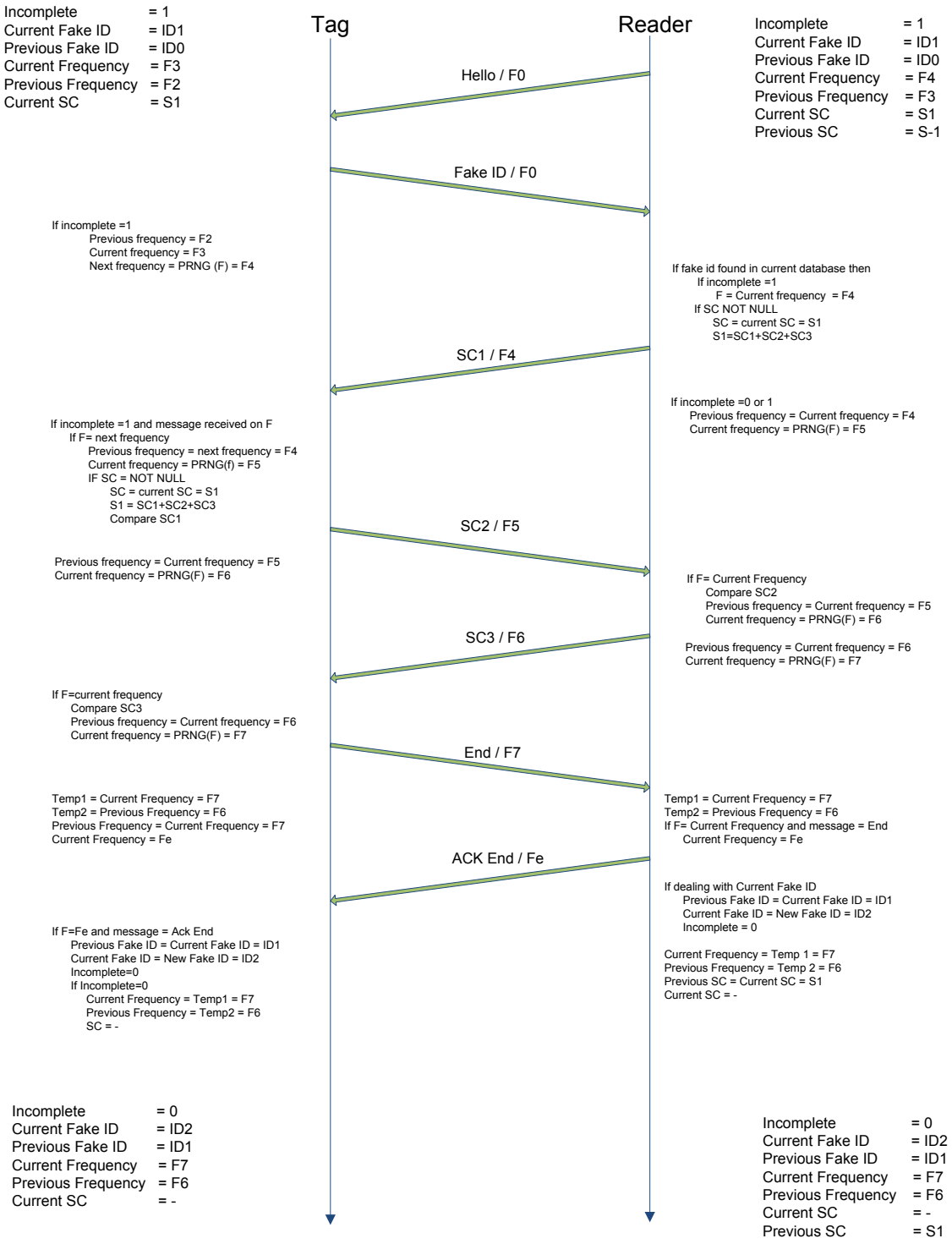

Figure 4.9 Recover SC3 on F3 Fails

Case 6: Tag to Reader END on F4 Fails

Type: Data lost while communicating

As shown in figure 4.10, the reader sends a *SC3* to the tag and generates next frequency *F4*. The tag gets secret code part three and compares it. If there is a match, it generates a next frequency *F4*. The tag now sends *END* signal to the reader and assigns next frequency as *Fe*. On the reader side, the reader is waiting for the *END* signal which is lost during communication. After timeout, both the reader and tag update their table.

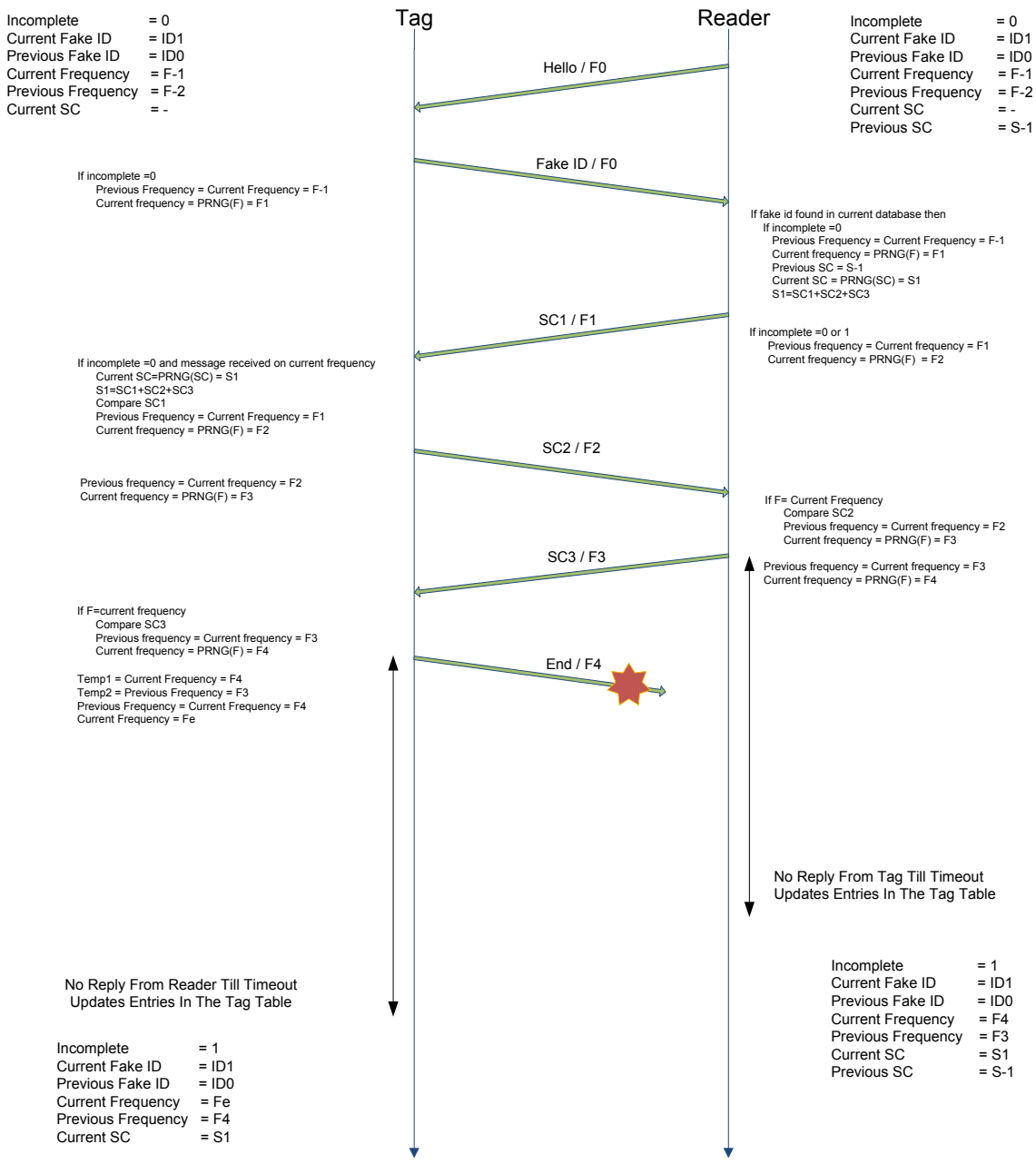

Figure 4.10 END on F4 Fails

When the tag comes in the vicinity of the reader again, both have incomplete value as one. As shown in figure 4.11, on the tag side, *current frequency* is *Fe* and on the reader side it is *F4*. The tag synchronizes the frequency to the reader by assigning a *previous frequency* to the *current frequency*.

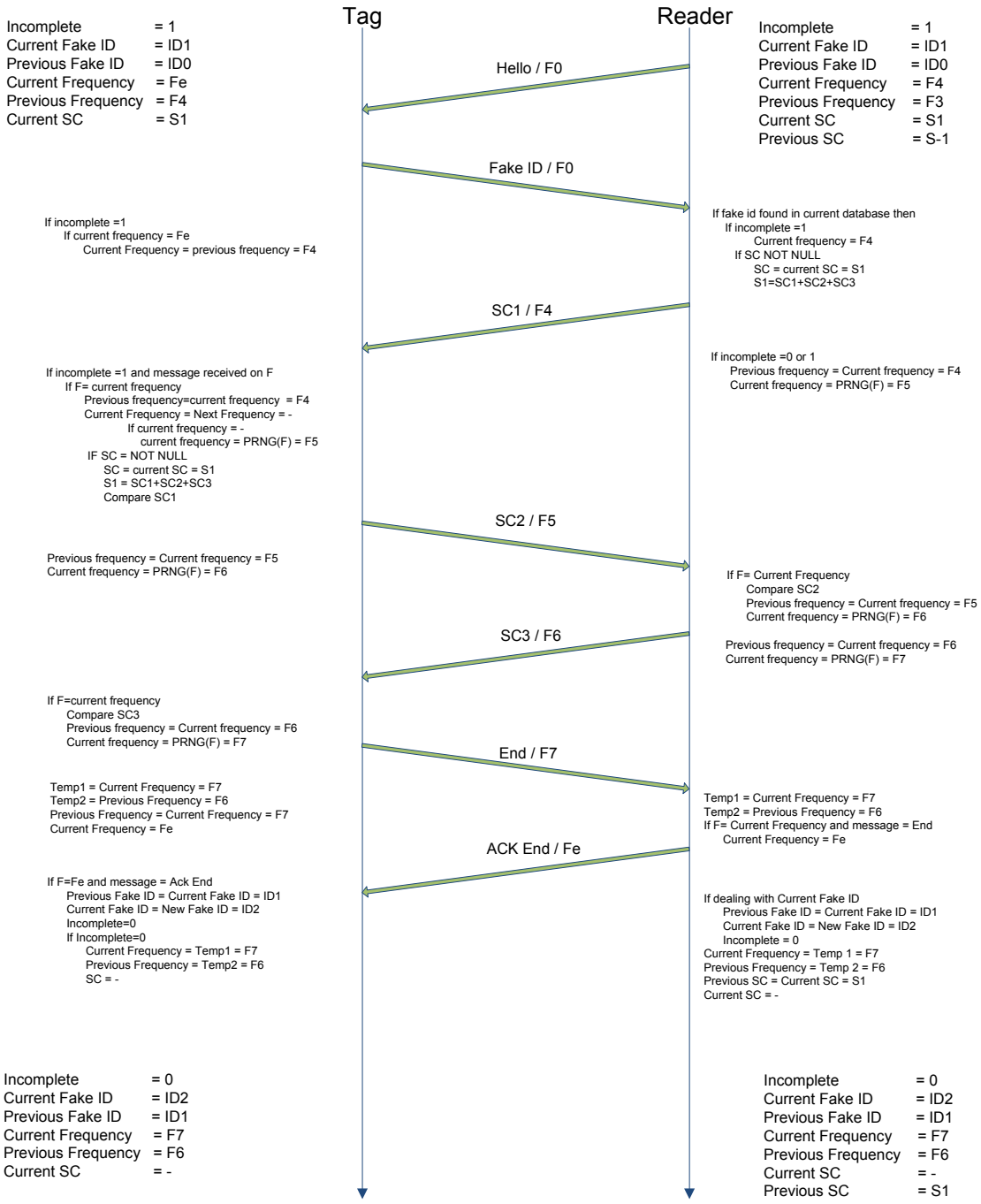

Figure 4.11 Recover END on F4 Fails

Case 7: Reader to Tag ACK END on Fe Fails

Type: Data lost while communicating

In this scenario, the last communication signal is lost. As shown in figure 4.12, after sending *ACK END* to the tag, the reader generates a new *fake ID*. Reader updates values for *current frequency*, *previous frequency*, *fake ID*, *Current SC* and *previous SC* in the reader table. On the reader side, the communication is successfully completed and *incomplete* is set to zero. On the tag side, the tag waits for the *ACK END* signal and after timeout the tag updates its table entries. The tag has *incomplete* as one and *current frequency* as *Fe*.

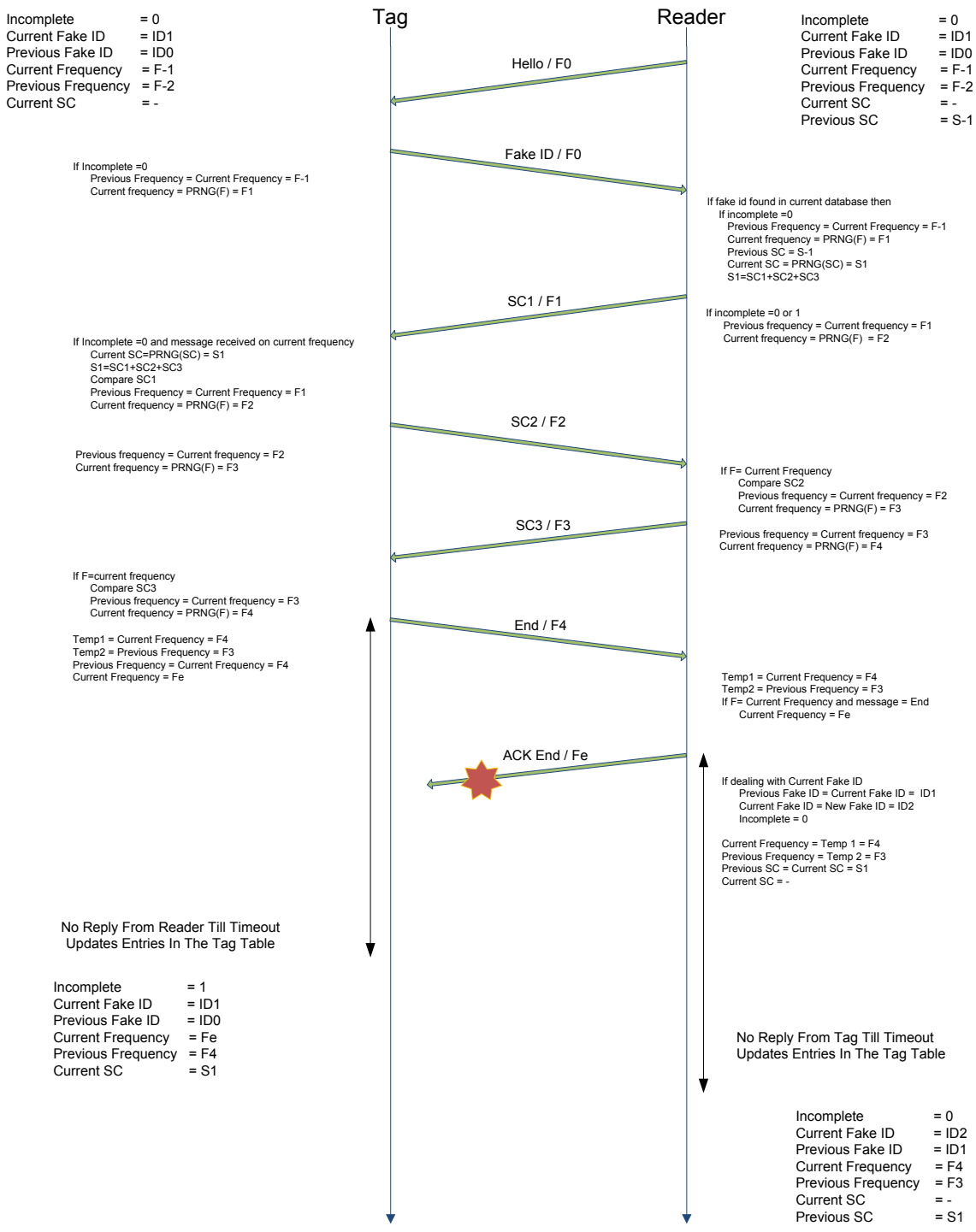

Figure 4.12 ACK END on Fe Fails

Figure 4.13 shows when the tag comes in contact with the reader again, the reader has a *current fake ID* and the tag has a *previous fake ID*. When the reader receives a *previous fake ID* from the tag, it first searches in the current *fake ID* database. When the *fake ID* is not found in the current fake ID database, it searches in the previous fake ID database and finds the tag. To synchronize the frequency, the tag uses the *previous frequency* as a *current frequency*. The reader will not generate a new frequency but uses the *current frequency* to communicate. Thus the reader and tag synchronizes the *current frequency* and *fake ID*.

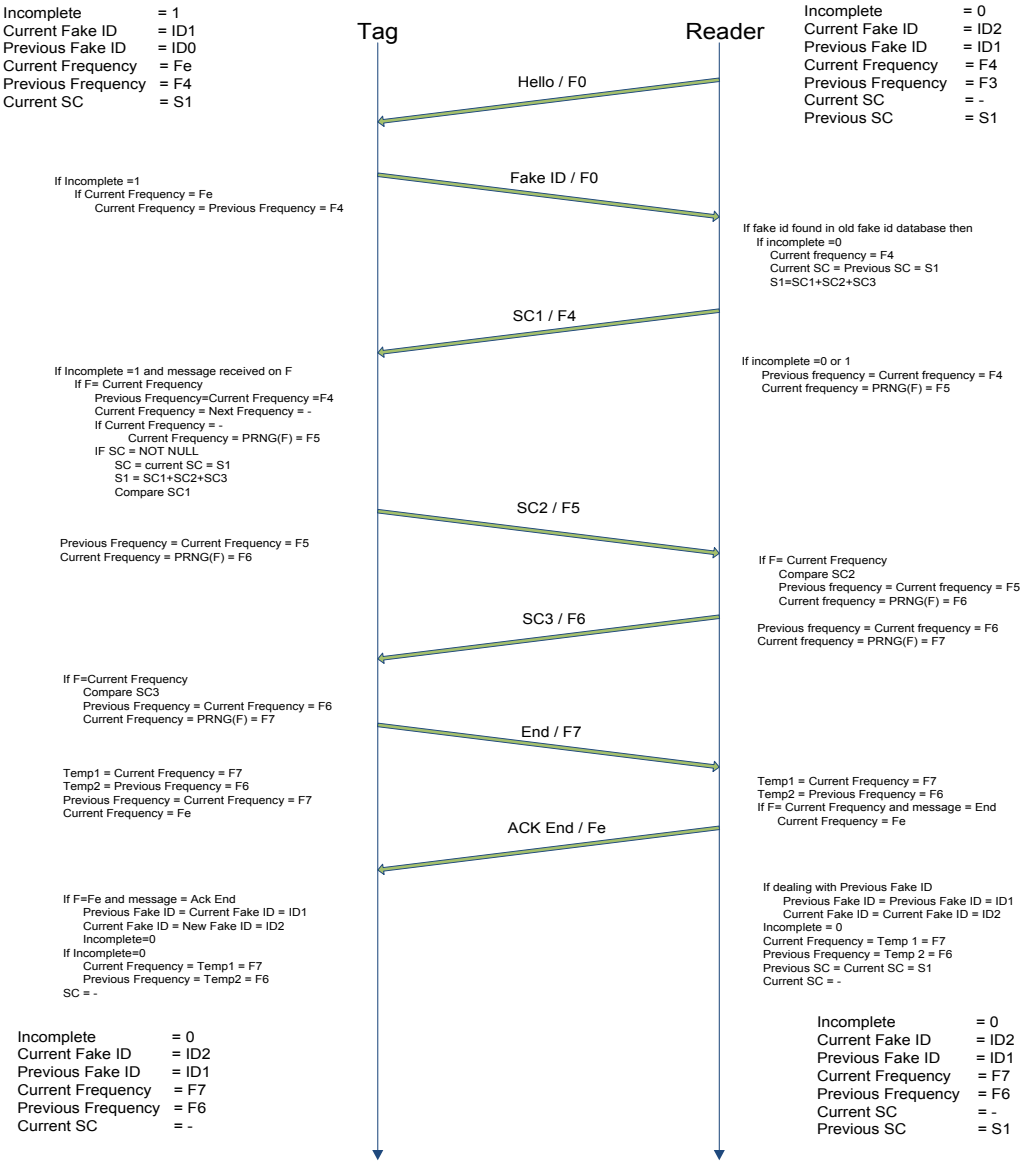

Figure 4.13 Recover ACK END on Fe Fails

4.2 UNAUTHORIZED READING

Unauthorized reading of a tag or a reader is performed to gain more information about the tag or the reader. To perform a cloning attack, an intruder reads the tag data from an unauthorized reader and generates a cloned tag from this information.

Case 8: Intruder Tag Tries to Communicate with the Reader

Type: Unauthorized reading

In this scenario, an intruder tag tries to communicate with a genuine reader. As shown in figure 4.14, when an intruder tag gets a *hello* message, it replies with the *fake ID*. Just to show this scenario, assume that the *fake ID* is correct *fake ID* of some genuine tag. So the reader generates next frequency and next secret code. Reader sends part one of the secret code to the tag and generates next frequency, in this case it is *F2*. An intruder tag does not have seed values to generate a frequency or secret code. So whatever the tag sends there is a rare chance that an intruder tag sends correct part one of secret code on the correct frequency. So after time out the reader updates its table entries.

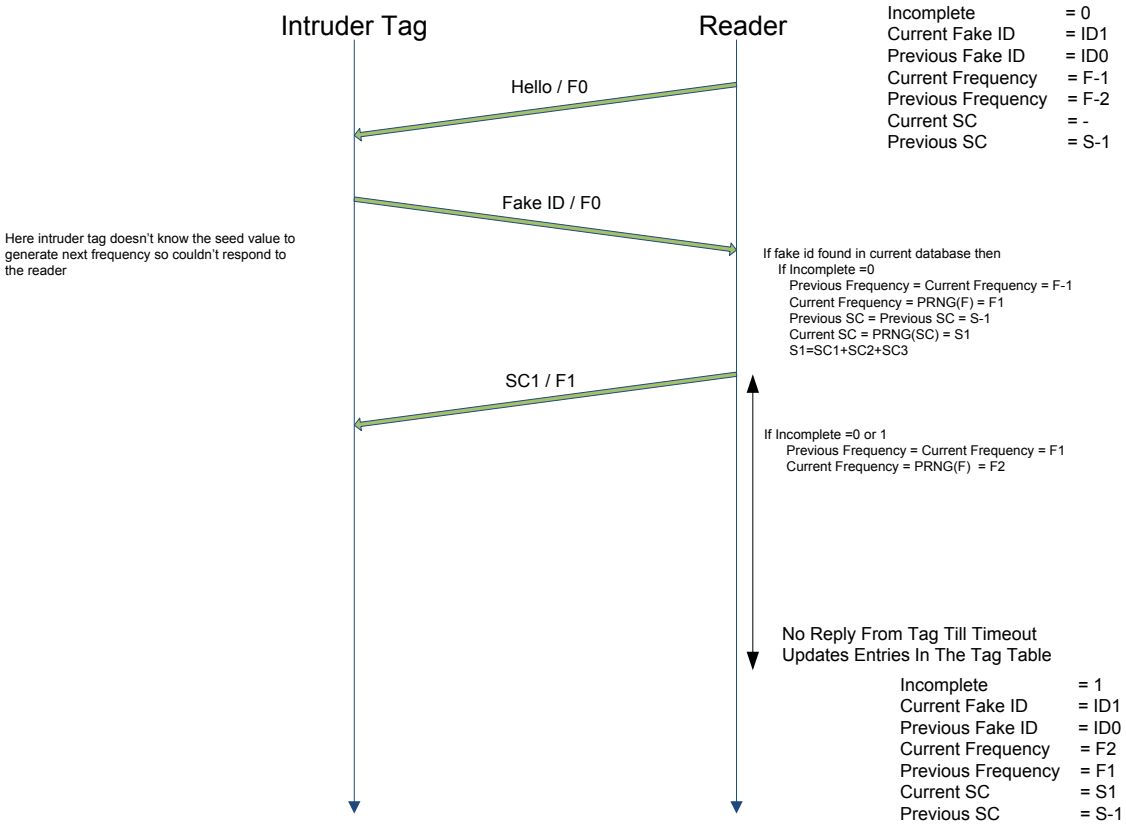

Figure 4.14 Unauthorized Reading from an Intruder Tag

In this scenario, the reader is ahead in generating frequency by two frequencies, so it takes two rounds to recover from this error. As shown in figure 4.15 when a genuine tag comes in contact with the reader, the tag has *incomplete* as zero but the reader has *incomplete* as one. In the first round the tag generates a new frequency as *F1*. The reader does not generate any frequency and keeps the *current frequency* as *F2*. Now the tag and reader has one level of frequency difference which is recovered in the second round.

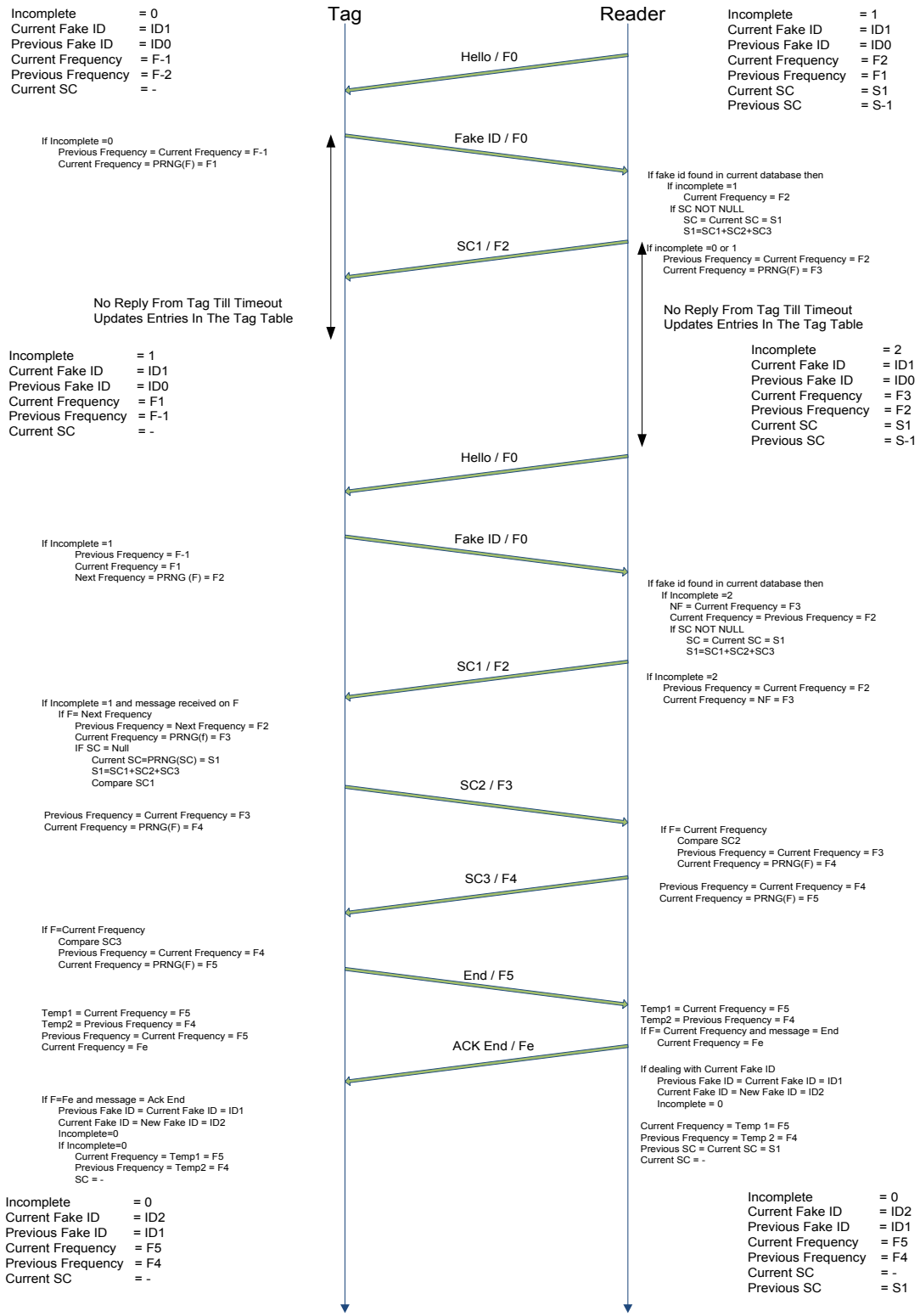

Figure 4.15 Recover Unauthorized Reading from an Intruder Tag

Case 9: Intruder Reader Tries to Communicate with Genuine Tag

Type: Unauthorized reading

In this scenario an intruder reader tries to read a tag. As shown in figure 4.16, an intruder reader sends *hello* message to the tag and tag replies with its *fake ID*. The tag generates next frequency. The intruder reader does not have seed values to generate a next frequency or secret code. Hence, the tag does not get reply from the reader and after timeout; the tag will update table entries.

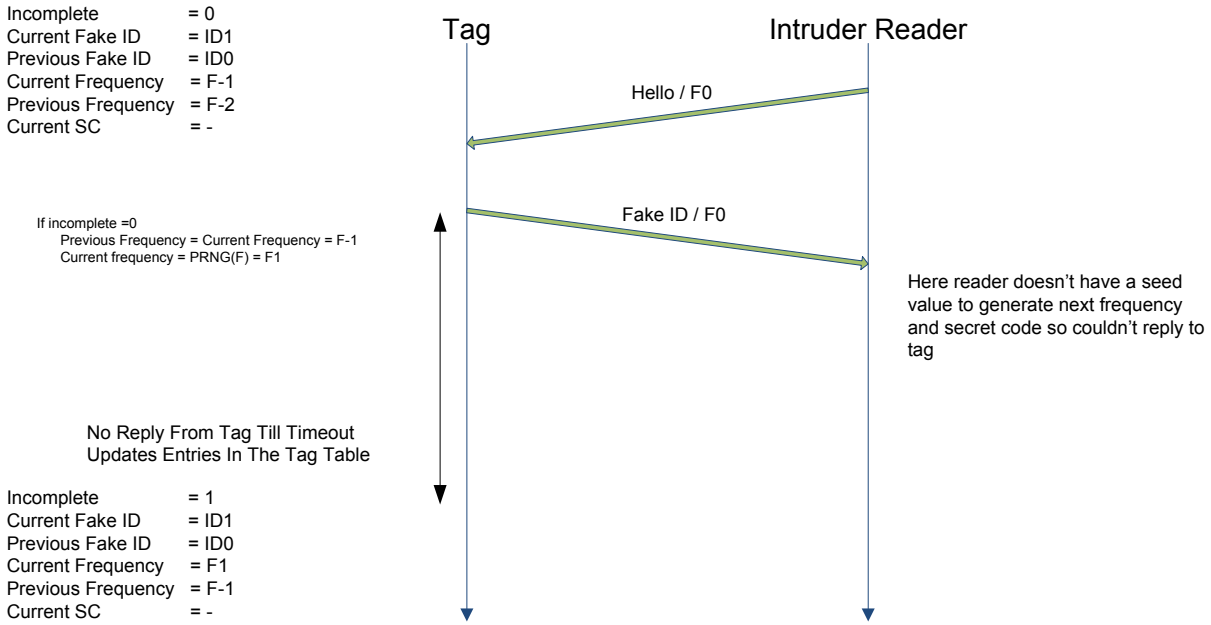

Figure 4.16 Unauthorized Reading from an Intruder Reader

As shown in figure 4.17, when a tag comes in contact with the genuine reader, it synchronizes frequency with the reader by not generating a new frequency in the second step.

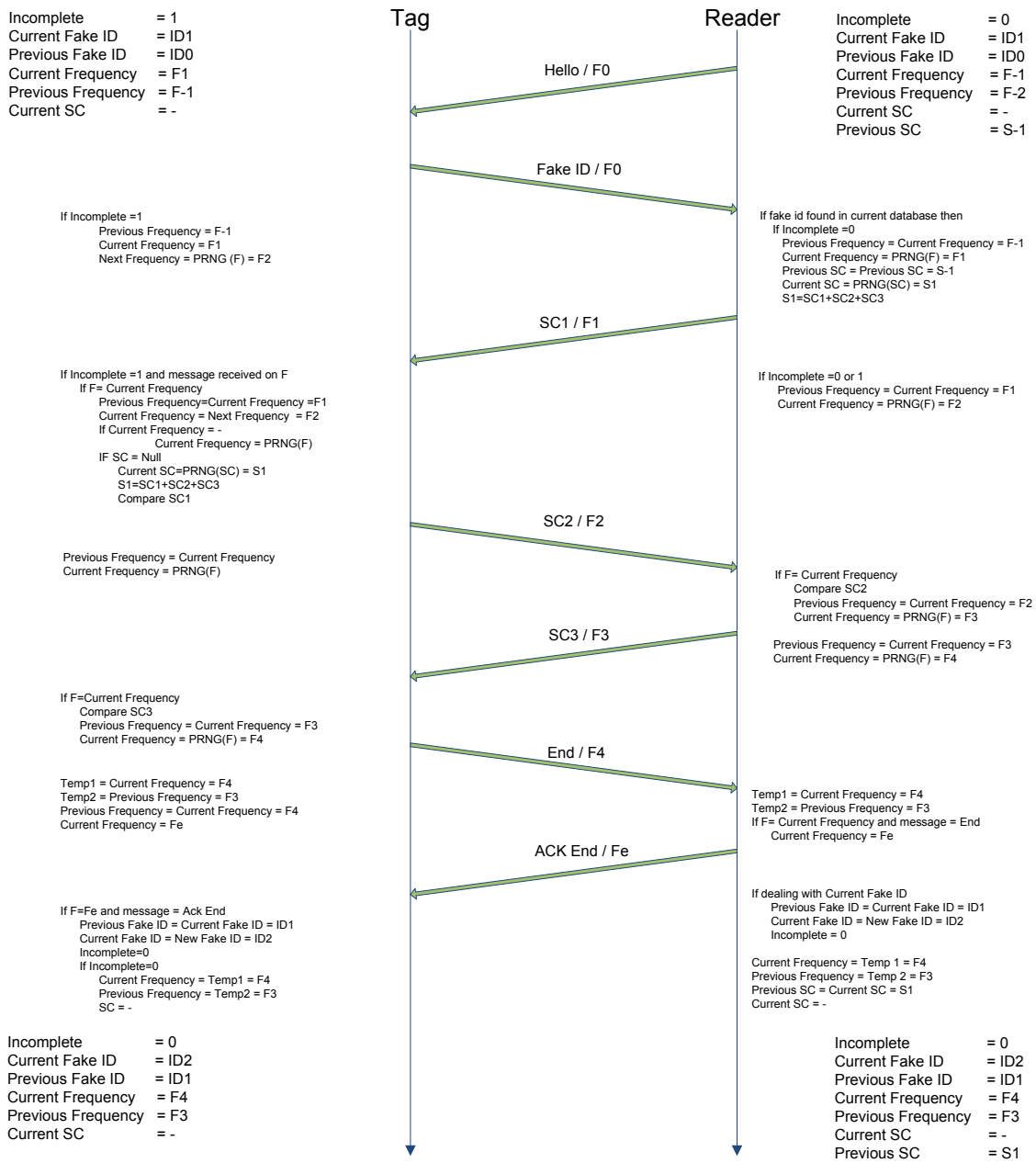

Figure 4.17 Recover Unauthorized Reading from an Intruder Reader

4.3 REPLAY ATTACK

An intruder captures communication between a tag and a reader, and then replays that information to gain access. An intruder tag or reader tries to impersonate the identity of a genuine reader or tag.

Case 10: Intruder Captures Data from Previous Communication and Launch Replay Attack with an Intruder Tag

Type: Replay Attack

An intruder retransmits captured data from the previous communication is known as replay attack. In this scenario, an intruder tag performs a replay attack. As shown in figure 4.18, the reader sends a *hello* message. The intruder tag sends the tag *fake ID* to the reader. The intruder tag has information captured from the previous communication, and it sends a *previous fake ID* to the reader. The reader finds *fake ID* in the old fake ID database, and the reader does not generate a new frequency. It sends part one of the secret code on the last frequency which is in this case is *F4*. Now, if this tag was a genuine tag and provided with the *previous fake ID*, it is only possible in the case 7. In that case, the tag should reply back with the part two of a previous secret key on a new frequency which is *F5*. But in this scenario, an intruder is trying to perform a replay attack; it sends part two of the secret code on frequency *F2*. Reader is expecting reply on frequency *F5* and it updates the reader table after timeout.

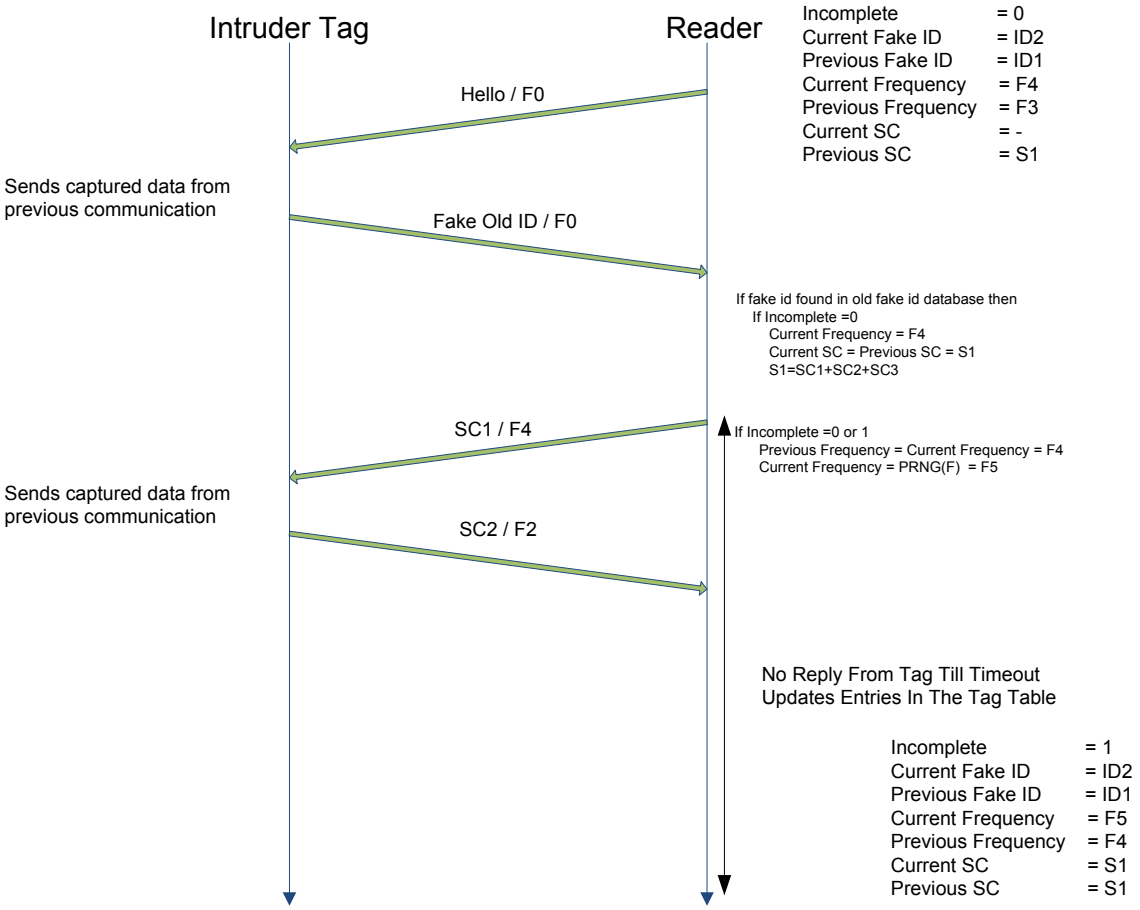

Figure 4.18 Replay Attack on Tag

When the reader comes in contact with a genuine tag, it uses *current frequency* of the last transaction as *current frequency*. Figure 4.19 shows how the reader and the tag synchronizes.

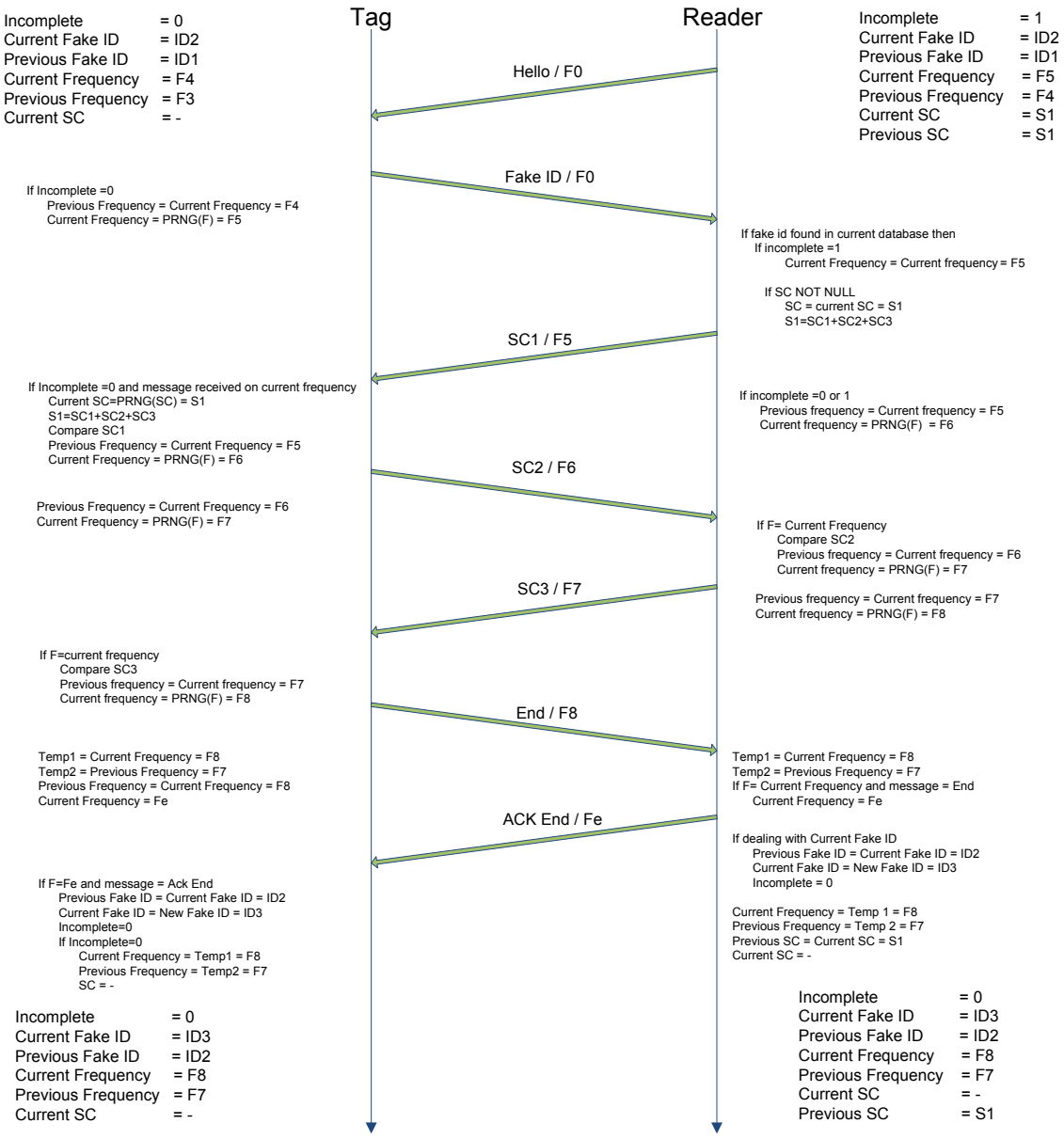

Figure 4.19 Recover Replay Attack on Tag

Case 11: Intruder Captures Data from Previous Communication and Launch Replay Attack with Intruder Reader

Type: Replay Attack

In this scenario, an intruder reader transmits the data captured from the previous communication. As shown in figure 4.20, an intruder reader sends *hello* message to the tag and the tag sends *fake ID* to the reader. An intruder reader retransmits part one of the secret code which is captured in previous transaction on *F1*. On the tag side it has generated new frequency *F5* and expecting reply from the reader on this frequency. After time out, the tag updates its table entries.

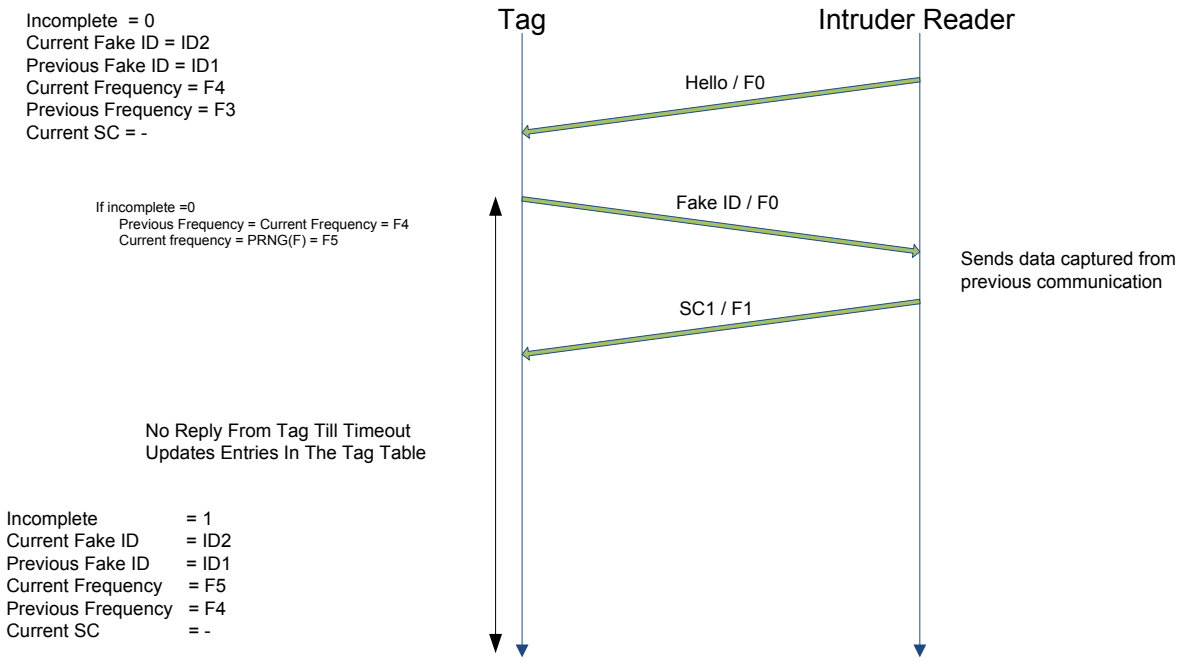

Figure 4.20 Replay Attack on Reader

When the tag comes in the vicinity of a genuine reader, it is one frequency ahead than the reader. Figure 4.21 shows how the tag synchronizes with the reader by not generating a next frequency in one step.

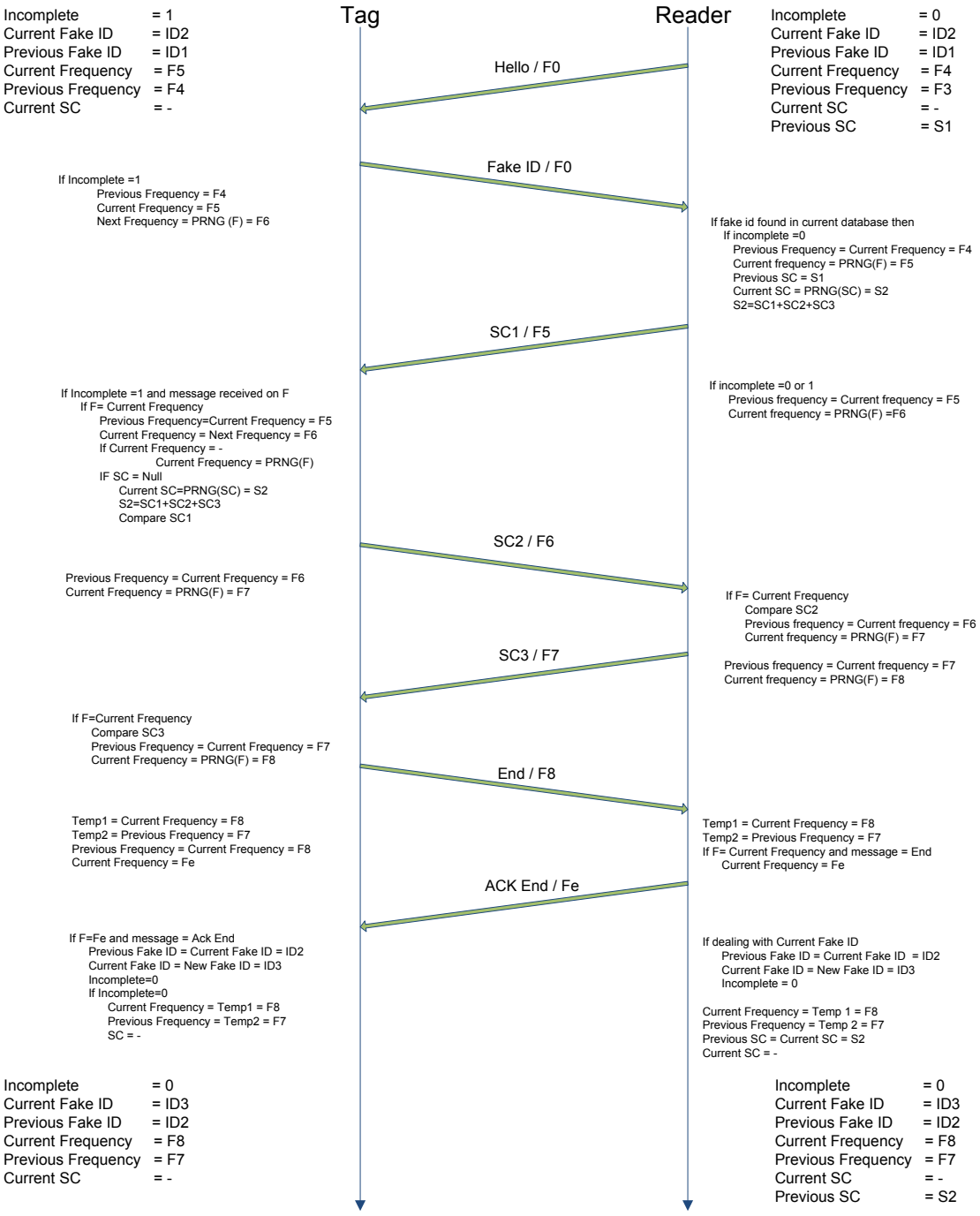

Figure 4.21 Recover Replay Attack on Reader

4.4 DE-SYNCHRONIZATION ATTACK

An intruder tag or reader sends information to a genuine reader or a tag and break the synchronization.

Case 12: Attacker Tag Tries to Communicate with Reader More Than One Time to Perform De-synchronization Attack

Type: De-Synchronization

As shown in figure 4.22, when an attacker tag receives a *hello* message, it sends *fake ID* to the reader. The reader generates a new frequency and a secret code. The reader sends part one of the secret code to the tag and generates next frequency *F2* for the next message. The reader does not get reply from the tag and after time out it updates its table. An attacker tag again performs de-synchronization. On the reader side, incomplete is one therefore the reader does not generate new frequency and uses *F2* as a *current frequency*. Again, after timeout reader updates the table entries. This time *incomplete* becomes 2 and *current frequency* is *F3*. Reader will not change value the frequencies if any more attacks are made from the tag side.

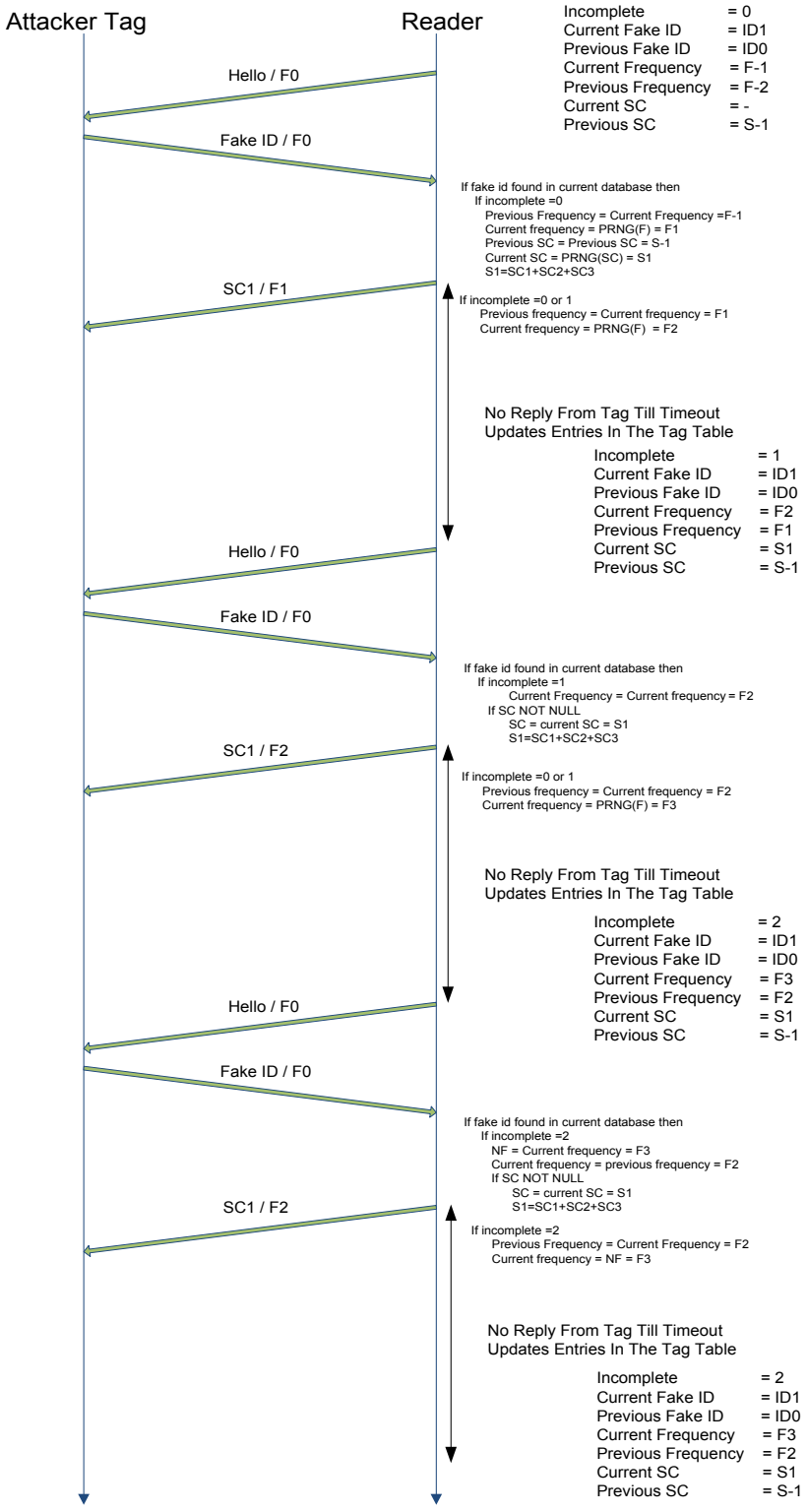

Figure 4.22 De-synchronization Attack by Tag

When a genuine tag comes in contact with the reader, the reader is two frequencies ahead than the tag so it takes two rounds to recover from the error. As shown in figure 4.23, during the first round on the tag side, as *incomplete* value is 2, the *current frequency* remains *F3*. On the tag side, the *current frequency* is *F2*. Both will update the table entries after timeout. In the second round, the reader does not generate a new frequency and uses *F3* as *current frequency*. On the tag side, it generates next frequency so now *current frequency* is *F3*. Thus the tag and the reader are synchronized to each other.

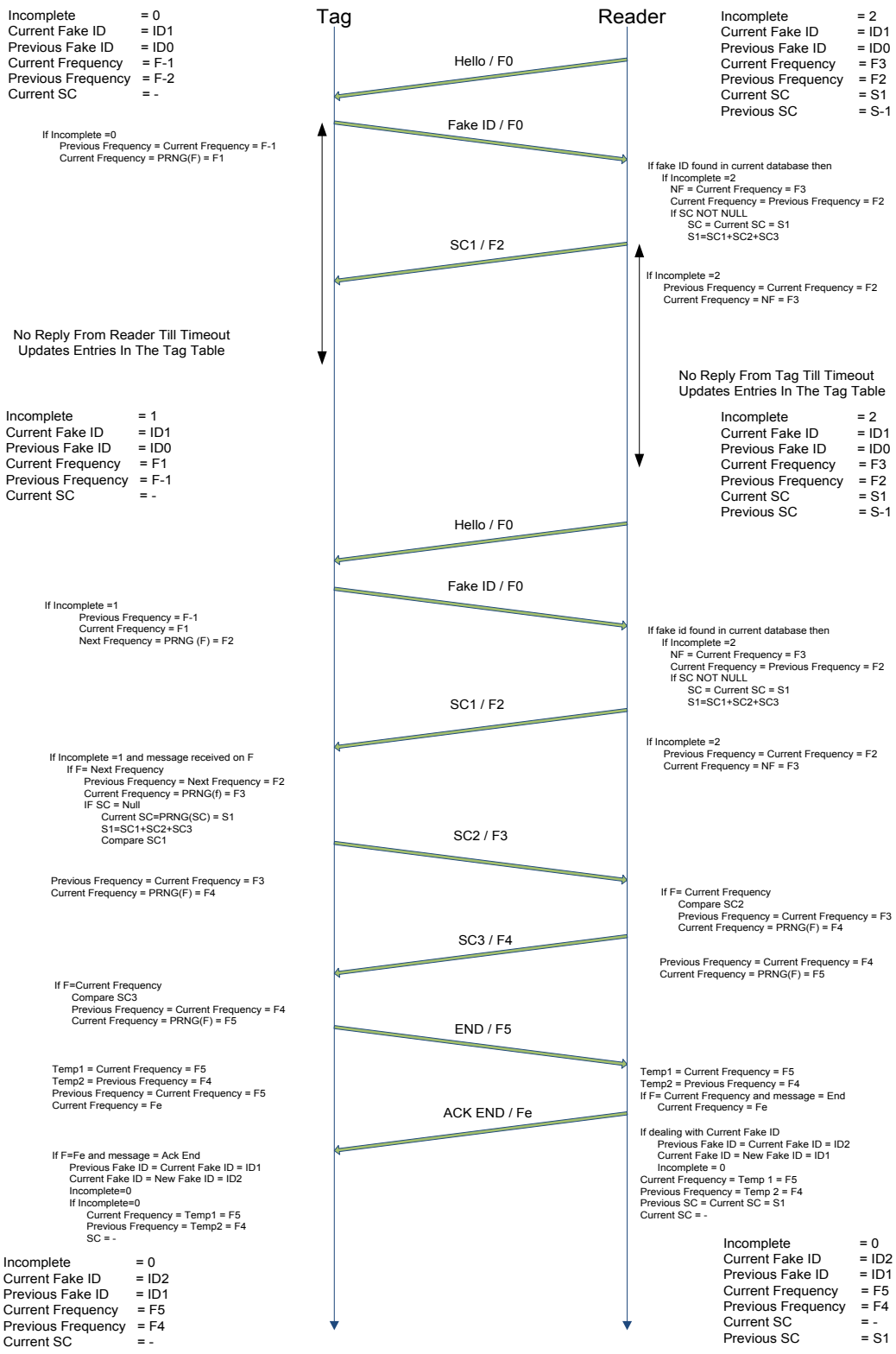

Figure 4.23 Recover de-synchronization Attack by Tag

4.5 MAN-IN-THE-MIDDLE ATTACK

An intruder captures the data between a tag and a reader then alters the information to stop the communication.

Case 13: An Intruder Captures Communication between the Tag and Reader and Changes the Data Transmitted

Type: Man-in-the-middle

In this attack, an attacker captures data from a communication, alters it and transmits it. Any of the altered messages are not received by the tag or the reader. Hence, all scenarios related to this are similar to the data loss scenarios and can recover by the same methods.

As per the classification [60], security protocols are classified in four categories.

Ultra Light Weight: simple bitwise operations like XOR, AND, OR on tags

Light Weight: random number generator, CRC but not hash functions

Simple: random number generator, one-way hashing function on tags

Full Fledged: symmetric encryption, cryptographic one-way function, public key algorithms

The protocol proposed here uses random number generator and compare functions (which are made from basic bitwise operations), therefore it falls under the light weight RFID security protocol. The analysis above shows that the combination of changing frequency, secret code and use of fake ID makes it difficult for an intruder to communicate with a tag or a reader. It not only helps preventing unauthorized reading but also prevents cloning attack, man-in-the-middle attack, de-synchronization attack and replay attack.

CHAPTER 5 CONCLUSION

The field of ubiquitous computing is growing due to the addition of computational power into mobile devices. Tags are having more memory than ever before. Some new RFID tags allow security protocols to be built into the hardware circuitry. Now, it is also possible to incorporate an RFID reader into a handheld device such as a cellular phone or a smartphone. Having a smart tag will promise enhanced security measures and prevent tag cloning.

From the results, it is clear that Radio Frequency Identification (RFID) security requires lightweight and strong protocol to defend against many forms of attacks. Having a synchronized protocol has the benefits of tracking the number of reads on both sides. With more and more computational power added to RFID tags, it may be possible to log the activities on both sides and analyze them when the RFID tag is back to the system creators.

The cloning prevention protocol presented here uses lightweight functions like PRNG and compare functions, which are possible to deploy in current tags. The protocol not only helps preventing cloning but also recovers from man-in-the-middle attacks, de-synchronization attacks, unauthorized reading attacks and replay attacks. The Java based simulation helped us to verify that the prptocol works in simulated environment and it is helpful in finding any flaws before physical deployment of the tags. The simulation also helped us in understanding how the protocol performs in the attack scenarios. Further extension can be made to develop an intrusion detection system, which monitors the number of reads on both sides to analyze any type of attack.

BIBLIOGRAPHY

[1] C. A.Walton. "Portable radio frequency emitting identifier." US Patent 4384288, May 1983.

[2] “ISO committee released cargo container security.” Internet: http://www.4engr.com/press/catalog/2390/index.html.

[3] “EPCglobal home page.” Internet: http://www.epcglobalinc.org/home.

[4] S. Sampalli et al. “Test Suite for Vulnerability Analysis of RFID Systems,” International Conference of Wireless Networks ICWN *2010*.

[5] “RFID standards.” Internet: http://nje-rfid.com/RFIDStandards.htm

[6] H. Lehpamer. *RFID design principles,* Boston: Artech House, 2008, ISBN: 9781596931947.

[7] “How to Write an RFID Virus.” Internet: http://www.rfidvirus.org/virus.html

[8] Industry Canada Deliverable, Phase 2, Deliverable 1.1.

[9] B.Jamali, P. H Cole, and D. Engel. “RFID tag vulnerabilities in RFID systems,” *Networked RFID Systems and Lightweight Cryptography 2008*, DOI: 10.1007/978-3-540-71641-9_7, pp 147–155.

[10] Y. Oren and A. Shamir. “Power analysis of RFID tags.” Internet: http://www.wisdom.weizmann.ac.il/~yossio/rfid/.

[11] N. T. Courtois. “Card-Only Attacks on MiFare Classic or How to Steal Your Oyster Card and Break into Buildings worldwide.” Internet: https://www.cosic.esat.kuleuven.be/rfidsec09/Papers/mifare_courtois_rfidsec09.pdf.

[12] M. Mohan, V. Potdar and E. Chang. “Recovering and Restoring Tampered RFID Data using Steganographic Principles,” *IEEE International Conference on Industrial Technology*, December 2006, ISBN: 1-4244-0726-5, pp 2853-2859.

[13] “The Australian ePassport, Australian Government, Department of Foreign Affairs and Trades.” Internet: http://www.dfat.gov.au/dept/passports/.

[14] “Sweden Switches to E-Passports.” *RFID Journal,* Internet: http://www.rfidjournal.com/article/print/1942.

[15] “Security expert cracks RFID in UK.” *Computer World,* Internet: passporthttp://computerworld.co.nz/news.nsf/news/7E6B10AC2689F81ECC2572970019E2AD.

[16] “British RFID passports cracked.” *Canadian Privacy Law Blog,* Internet: http://www.privacylawyer.ca/blog/2006/11/british-rfid-passports-cracked.html.

[17] “Cracked it.” *The guardian*, Internet: http://www.guardian.co.uk/technology/2006/nov/17/news.homeaffairs.

[18] K. Koscher, A. Juels, T. Kohno and V. Brajkovic. “EPC RFID Tags in Security Applications: Passport Cards, Enhanced Drivers Licenses, and Beyond.” Internet: http://www.rsa.com/rsalabs/staff/bios/ajuels/publications/EPC_RFID/Gen2authentication --22Oct08a.pdf.

[19] “Help for Boeing, Airbus Suppliers.” Internet: http://www.rfidjournal.com/article/articleview/1226/1/1

[20] **“**Smart Border Alliance RFID Feasibility Study Final Report.” RFID Security and Privacy White Paper, USVISIT-APMO-CONTHSSCHQ04D0096T006-RPT050010-F, Internet: http://www.dhs.gov/xlibrary/assets/foia/US-VISIT_RFIDattachE.pdf.

[21] P. H. Cole and D. C. Ranasinghe. *Networked RFID Systems and Lightweight Cryptography: Raising Barriers to Product Counterfeiting.* (1st edition) ISBN-10:3540716408, Springer.

[22] L.T. Mirowski and J.S. Hartnett. “Deckard: A System to Detect Change of RFID Tag Ownership,” *International Journal of Computer Science and Network Security*, ISSN 1738-7906, vol. 7, pp 89-99. Jul 2007.

[23] J. Kim and H. Kim. “Anti-Counterfeiting Solution Employing Mobile RFID Environment,” *in Proc. World Academy of Science, Engineering and Technology,* 2005, vol. 8, pp 141-144.

[24] M. Lehtonen, F. Michahelles and E. Fleisch. “How to Detect Cloned Tags in a Reliable Way from Incomplete RFID Traces,*” in 2009 IEEE International Conference on RFID*, Orlando, Florida, pp 257 – 264, Apr 27-28, 2009.

[25] M. Lehtonen, F. Michahelles and E. Fleisch. “Probabilistic Approach for Location-Based Authentication,” *in 1st International Workshop on Security for Spontaneous Interaction (IWSSI) at UbiComp’07*, Austria, Sep 2007.

[26] A. Ilic, M. Lehtonen, F. Michahelles and E. Fleisch. “Synchronized Secrets Approach for RFID-enabled Anti-Counterfeiting,” *Demo at Internet of Things Conference*, Zurich, Switzerland, 2008.

[27] B. Danev, T. S. Heydt-Benjamin and S. Capkun. “Physical-layer Identification of RFID Devices,” *Proceedings of the 18th conference on USENIX Security Symposium*, pp 199-214, 2009.

[28] L. Bolotny and G. Robins. "Physically Unclonable Function -Based Security and Privacy in RFID Systems," *Proceedings of the 5th Annual IEEE International Conference on Pervasive Computing and Communications*, pp 211-220*,* Mar 2007.

[29] A. Yamamoto, S. Suzuki, H. Hada, J. Mitsugi, F. Teraoka, O. Nakamura. "A Tamper Detection Method for RFID Tag Data," *2008 IEEE International Conference on RFID*, pp 51-57*,* 16-17 Apr 2008.

[30] S. Han, C. Chao-Hsien. "Tamper Detection in RFID-Enabled Supply Chains Using Fragile Watermarking," *2008 IEEE International Conference on RFID*, pp. 111-117*,* 16-17 Apr 2008.

[31] M. Lehtonen, D. Ostojic, A. Ilic and F. Michahelles. "Securing RFID systems by detecting tag cloning," *In the 7th International Conference on Pervasive Computing, Pervasive'09,* Nara, Japan, pp 291-308, May 2009.

[32] G. Thamilarasu and R. Sridhar. "Intrusion Detection in RFID Systems," *IEEE MILCOM,* 2008, pp. 1-7.

[33] R. A. Wasniowski. "Database Support for Discovering Patterns in Large Datasets Collected From Multiple Sensors," *Proceedings of the 5thInternational Conference on Information Technology: New Generations*, pp. 1279-1280, 2008.

[34] M. Aigner, T. Burbridge, A. Ilic, D. Lyon, A. Soppera, M. Lehtonen. "RFID Tag Security," Building Radio frequency Identification for the Global Environment, Internet: http://www.bridge-project.eu/data/File/BridgesecuritypaperDL_9.pdf.

[35] K. Koscher, A. Juels, T. Kohno, and V. Brajkovic. "EPC RFID Tags in Security Applications: Passport Cards, Enhanced Drivers Licenses, and Beyond," *Draft manuscript, in submission, RSA Laboratories,* 2008.

[36] M. Lehtonen, T. Staake, F. Michahelles, and E. Fleisch. "From Identification to Authentication – A Review of RFID Product Authentication Techniques", *In Networked RFID Systems and Lightweight Cryptography II*, Springer, pp. 169-187, 2008.

[37] R. Anderson. "RFID and the Middleman," *USEC*, vol. LNCS (4886), pp. 46-49, 2007.

[38] "RFID Feasibility Study Final Report." RFID Security and Privacy White Paper, USVISIT-APMO-CONTHSSCHQ04D0096T006-RPT050010-F, Internet: http://www.dhs.gov/xlibrary/assets/foia/US-VISIT_RFIDattachE.pdf

[39] "The Consequences to Citizen Privacy and National Security in Adopting RFID Technology for, Border Crossing Identity Documents." Smart Card Alliance Identity Council, Oct 2007, Internet:

http://www.smartcardalliance.org/resources/pdf/Border_Identity_Technology.pdf

[40] T. Dimitriou. "A Lightweight RFID Protocol to protect against Traceability and Cloning attacks," *Proceedings of the 5th International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pp. 59-66, Sep 2005.

[41] T. C. Chiu, Q. Li. "A robust and secure RFID-based pedigree system (short paper)," *International Conference on Information and Communications Security*, Raleigh NC, vol. 4307, no. 8, pp. 21-29, 2006.

[42] D. M. Konidala, Z. Kim and K. Kim. "A simple and cost-effective RFID tag-reader mutual authentication scheme," *Proceedings of International Conference on RFID Security (RFIDSec)'07*, pp. 141-152, Jul 2007.

[43] M. Mostafa, El-Said, I. Woodring. "An Empirical Study for Protecting Passive RFID Systems against Cloning," *Proceedings of the 2009 6th International Conference on Information Technology: New Generations*, vol. 00, pp. 558-563, 2009.

[44] E. Y. Choi, D. H. Lee, J. I. Lim. "Anti-cloning protocol suitable to EPCglobal Class-1 Generation-2 RFID systems," *Computer Standards & Interfaces*, vol. 31, issue 6, pp. 1124-1130, Nov 2009.

[45] M. Kwak, J. Kim and K. Kim. "Desynchronization and Cloning resistant light-weight RFID authentication protocol using integer arithmetic for low-cost tags," *Symposium on Cryptography and Information Security (SCIS 2009)*, Otsu, Japan, 2009.

[46] J. Abawajy. "Enhancing RFID Tag Resistance against Cloning Attack," 3rd *International Conference on Network and System Security*, pp.18-23, 2009.

[47] D. N. Duc, J. Park, H. Lee, and K. Kim. "Enhancing security of EPC global gen-2 RFID tag against traceability and cloning," *Proceedings of SCIS*, Hiroshima, Japan, pp. 97, 2006.

[48] D. N. Duc, H. Lee, K. Kim. "Enhancing Security of Class I Generation 2 RFID against Traceability and Cloning," *Proceedings of Networked RFID Systems and Lightweight Cryptography*, pp. 269-277, 2008.

[49]M. Mitra. "Privacy for RFID Systems to Prevent Tracking and Cloning," *in Proceedings of International Journal of Computer Science and Network Security (IJCSNS)*, vol. 8 (no. 1), pp. 1-5, Jan 2008.

[50] P. Tuyls and L. Batina. "Rfid-tags for anti-counterfeiting," *in Topics in Cryptology (CT-RSA 2006)*, Ser. LNCS 3860, pp. 115-131, Feb 2006.

[51] Y. Liang, C. Rong. “Strengthen RFID Tags Security Using New Data Structure,” *International Journal of Control and Automation*, ISSN 2005-4297, vol.1, pp.51-58, 2008.

[52] A. Juels. “Strengthening EPC tags against cloning,” *Proceedings of the 4th ACM workshop on Wireless security*, Cologne, Germany, pp. 67-76, Sep 2002.

[53] M. Lehtonen, D. Ostojic, A. Ilic and F. Michahelles. “Securing RFID systems by detecting tag cloning,” *Proceedings of the 7th International Conference on Pervasive Computing*, Nara, Japan, vol. 5538, pp. 291-308, May 2009.

[54] “Frequency allocation and radio treaty matters; general rules and regulations.” Internet:http://ecfr.gpoaccess.gov/cgi/t/text/text-idx?type=simple;c=ecfr;cc=ecfr;sid= 1bc 9d9b4b0021e4d5e786fe12abb0170;region=DIV1;q1=fcc%20rule;rgn=div5;view=text;id no=47;node=47%3A1.0.1.1.3#47:1.0.1.1.3.7.218.1

[55] C. Tankink. “Misplaced trust - RFID tags as malware carriers”, Internet: http://www.win.tue.nl/~aserebre/2IF03/2008/papers/2/Carst.pdf.

[56] P. López. “Lightweight Cryptography in Radio Frequency Identification (RFID) Systems,” Ph.D. Thesis report, Internet:
http://pperis.host22.com/pperis/thesis/peris_thesis.pdf.

[57] Y. Oren and A. Shamir. “Power analysis of RFID tags,” Internet: http://www.wisdom.weizmann.ac.il/~yossio/rfid/ [Jan 13, 2009].

[58] “Are there any standards for RFID?.” RFID Journal, Internet:

http://www.rfidjournal.com/faq/21/90

[59] W. Hansen and F. Gillert. “RFID for the Optimization of Business Processes,” *Edition illustrated,* John Wiley and Sons, 2008, ISBN 0470724226, 9780470724224.

[60] H. Chien. “SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity,” *IEEE transactions on dependable and secure computing*, vol. 4, no. 4, Oct-Dec 2007.

[61] “Boeing and Airbus Team Up for RFID Standards.” Internet: http://www.dailywireless.com/news/boeing-and-airbus/

[62] M. W. Cardullo, W. L. Parks III. “Transponder Apparatus and System.” US Patent 3713148, Jan 1973.