Security Analysis of the ISO 14443 RFID Standard Regarding Relay Attacks

Wolfgang Issovits
wolfgang.issovits@student.tugraz.at

Institute for Applied Information
Processing and Communications (IAIK)
Graz University of Technology
Inffeldgasse 16a
8010 Graz, Austria

Master Thesis

Supervisor: Dipl.-Ing. Dr.techn. Michael Hutter
Assessor: Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Karl-Christian Posch

May, 2011

Acknowledgements

I would like to thank my supervisor Michael Hutter for his support and advice during the work on this master thesis. I also would like to thank Stefan Lemsitzer, Christopher Hubmann, and Reinhard Meindl for their technical support and valuable inputs. Most importantly I would like to thank my family and my girlfriend Andrea for their support and encouragement.

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

date	(Wolfgang Issovits)

Abstract

Radio-Frequency Identification (RFID) and Near-Field Communication (NFC) are widely spread contactless communication systems and are commonly used in security-critical applications such as electronic payment and keyless-entry systems. Relay attacks pose a serious threat in this context and are not addressed by most RFID applications in use today. Those attacks circumvent application-layer security and cannot be prevented by the usual cryptographic primitives. In this thesis, we present a practical implementation of a relay attack on systems implementing the ISO/IEC 14443 RFID standard, which was developed by the International Organization for Standardization and the International Electrotechnical Commission. We use an off-the-shelf mobile phone with NFC capabilities and a self-developed RFID-tag emulator that can forward RFID communication over a Bluetooth channel. We show that the attack succeeded and discuss various methods to exploit certain mechanisms of the ISO/IEC 14443 protocol in order to increase the chance for a successful attack. We also give recommendations on how to protect against relay attacks in practice while still complying to the ISO/IEC 14443 standard, which is not considered by most of the proposed countermeasures given in literature.

Keywords: RFID, NFC, Relay Attacks, ISO/IEC 14443.

Kurzfassung

Radio-Frequenz-Identifikation (RFID) und Nah-Feld-Kommunikation (NFC) sind weit verbreitete kontaktlose Kommunikationssysteme und werden häufig für sicherheitskritische Anwendungen, wie elektronische Zahlungs- oder Zugangssysteme, verwendet. Relay-Attacken sind in diesem Zusammenhang eine große Gefahr und werden von aktuell verwendeten RFID-Systemen nur selten berücksichtigt. Diese Attacken umgehen Sicherheitsmechanismen in der Anwendungsschicht und können nicht durch herkömmliche kryptographische Vorgehensweisen verhindert werden. In dieser Arbeit wird eine praktische Relay-Attacke auf Systeme vorgestellt, die den weit verbreiteten ISO/IEC 14443 RFID-Standard verwenden, welcher von der Internationalen Organisation für Normung (ISO) und der Internationalen Elektrotechnischen Kommission (IEC) entwickelt wurde. Dazu werden ein NFC-kompatibles Mobiltelefon und ein selbst entwickelter RFID-Kartenemulator verwendet, welche RFID-Kommunikation über eine Bluetooth-Verbindung weiterleiten können. Die Relay-Attacke auf ein ISO/IEC 14443 Testsystem wurde erfolgreich durchgeführt. Zusätzlich wurden Eigenschaften des ISO/IEC 14443 Standards ausgenutzt, um die Attacke noch effektiver zu gestallten. Des Weiteren werden Möglichkeiten vorgestellt, wie man sich in der Praxis gegen Relay-Attacken schützen kann. Diese wurden konform mit dem ISO/IEC 14443 Standard entwickelt, was bei den meisten in der Literatur existierenden Gegenmaßnahmen nicht der Fall ist.

Stichwörter: RFID, NFC, Relay-Attacken, ISO/IEC 14443.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Radio-Frequency Identification (RFID) is a contactless communication technology that allows a wide range of different applications. RFID systems usually consist of a reader and a transponder. The reader typically consists of a radio-frequency module (to transmit and to receive data), a control unit, and a coupling element to the transponder. The transponder is the data-carrying device of an RFID system and typically contains a coupling element and an electronic microchip. There exist different RFID technologies, which differ in various characteristics such as physical appearance, operating frequency, transmission range, operation mode, coupling method, and power supply of the transponder. Near-Field Communication (NFC) is based on RFID technology and is used for similar applications [9].

There exists a variety of different standards for RFID systems. One of the most common standards for proximity cards (a special class of RFID transponders) is defined in the International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) 14443 specification. It is used by major RFID systems like MIFARE [36], Calypso [4], or electronic passports. NXPs MIFARE system, for example, is used for transit systems in over 650 cities worldwide and had an estimated market share of 75 % in 2010 [32]. Therefore, this standard is a very interesting target for attackers.

With the wide distribution of RFID systems, attacks on those systems evolved. Among others there exist so called relay attacks. During a relay attack, an attacker establishes a channel between two legitimate parties without them being aware of the relay. Relay attacks on RFID systems involve two devices. One device impersonates the reader to the transponder and is called *mole*. The other device impersonates the transponder to the reader and is called *proxy*. Those two devices are connected by a relay channel. One of the crucial factors for the success of a relay attack is the additional delay introduced by the relay [12].

The ISO/IEC 14443 standard does not introduce any security measures against relay attacks. Moreover, it defines a number of mechanisms that can be exploited by such attacks. One of those is the Frame Waiting Time (FWT, the maximum response time for the RFID transponder), which can be chosen up to 4.95 seconds. It also defines recovery functionality (Negative Acknowledges, NAKs), which give an attacker additional time for a relay. It further allows the transponder to request additional computation time using Waiting Time Extensions (WTXs).

In this thesis, we show that the exploitation of those mechanisms is possible and allows relay attacks to be performed on RFID systems. We implemented a relay attack using an off-the-shelf mobile phone with NFC capabilities as mole device and a self developed

RFID-tag emulator as proxy device. Furthermore, we propose countermeasures that are compliant with the ISO/IEC 14443 standard.

First, we introduce a protocol that prevents the exploitation of Waiting Time Extensions. Second, we propose a transmission-parameter check in order to ensure that reader and transponder use the same data rate, block size, and frame waiting time. This ensures that an attacker cannot use different transmission parameters at both ends of the attack to gain additional time for the relay. Finally, we implement an ISO/IEC 14443 compliant distance-bounding protocol. Distance bounding tries to estimate the distance between the reader and the transponder. Therefore, the reader sends a challenge that only the transponder can solve. The transponder should be able to respond in a fixed amount of time so the reader can estimate the distance based on the Round-Trip Time (RTT) during the exchange.

In this thesis, we successfully perform relay attacks on an ISO/IEC 14443 compliant system. Our experiments show that certain mechanisms of the ISO/IEC standard can be exploited for relay attacks. The proposed countermeasures help to prevent the exploitation of those mechanisms while complying to the standard.

This thesis is organized as follows: In **Chapter 2**, we give a brief introduction on RFID. First, we describe the reader and the transponder. Second, we categorize RFID systems with regards to their properties. Finally, we give an overview of existing RFID standards.

Relay attacks are analyzed in detail in **Chapter 3**. We start by presenting the history of relay attacks and their application in different fields. Then we analyze the components of a relay attack, namely the proxy device, the mole device, and the relay channel. We continue this chapter with a close look at the introduced delay during a relay attack. Finally, we conclude the chapter with relay attacks and countermeasures that exist in literature.

Chapter 4 presents the four parts of the ISO/IEC 14443 standard. Part 1 of the standard defines the physical characteristics. The characteristics of the fields to be provided for power and bi-directional communication are defined in Part 2. Initialization and anticollision (handling of multiple transponders) is explained in Part 3. Part 4 defines a half-duplex block-transmission protocol, which is especially designed for contactless environments.

A security analysis of the ISO/IEC 14443 standard regarding relay attacks is performed in **Chapter 5**. First, possible attack scenarios on the physical layer are described. Moreover, we propose the exploitation of mechanisms on the protocol layer.

Chapter 6 presents a number of countermeasures against relay attacks. All countermeasures are designed to be compliant with the ISO/IEC 14443 standard so an integration into existing systems is possible.

Chapter 7 describes the hardware used for our experiments. We present the proxy device (an RFID-tag emulator) and the mole device (an off-the-shelf mobile phone). The test system, consisting of a reader and a transponder, is presented as well.

Details of our relay-attack implementation are presented in **Chapter 8**. We describe how the relay attack was implemented on the proxy device and the mole device as well as how the communication between the two was realized. The implementation of the countermeasures is also outlined in this chapter.

In **Chapter 9**, the results of our experiments are covered. First, we present the outcomes of our relay attack on the ISO/IEC 14443 test system. We also look at the introduced delay and the behavior of the test system regarding the exploitation of protocol mechanisms. Finally, we look at the performance of the implemented countermeasures and how they

can be used in a real system.

Chapter 10 concludes this thesis with the most important outcomes of our experiments. Finally, we suggest possible research topics regarding relay attacks on RFID systems.

Chapter 2

Radio-Frequency Identification

Radio-Frequency Identification (RFID) is a contactless communication technology that allows a wide range of different applications. Those applications differ in physical appearance, operating frequency, transmission range, operation mode, coupling method and power supply of the transponder.

The basic concept of RFID is shown in Figure 2.1. Every RFID system consists of a reader and a transponder. The transponder is the data-carrying device of the system. The reader generates an electric, a magnetic, or an electromagnetic field that is detected by the transponder. This field supplies power and clock frequency for the transponder, which usually does not have an additional power supply. This operation field is also used for data exchange between reader and transponder. They both use different modulation techniques to send data to each other [9].

The following chapter classifies RFID systems with regards to their properties. First, we describe the reader and the transponder. We also describe different classifications of RFID sytems and focus on the most important aspects for our experiments. The information in the following sections is based on Finkenzellers RFID Handbook [9], unless stated otherwise.

2.1 The Reader

The reader, sometimes also called interrogator, is the active device during communication. Although we call it a reader in accordance to literature, it is also capable of writing data.

A typical RFID reader has a radio-frequency module that consists of a transmitter and a receiver. Furthermore, it is composed of a control unit and a coupling element to communicate with the transponder. Usually a reader has one or more additional

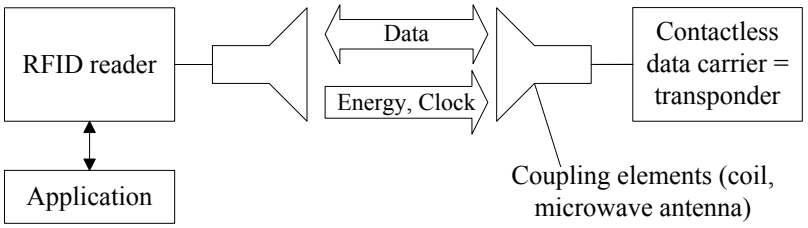


Figure 2.1: Basic concept of RFID systems [9].

interfaces (*e.g.* RS-232, USB) to communicate with other systems. This system could be a PC or a backend system, which runs the application that controls the reader.

2.2 The Transponder

A transponder is a contactless data-carrying device, which is sometimes also called an RFID tag or target. It normally consists of a coupling element and an electronic microchip. The simplest version is a 1-bit transponder, which only allows two states, namely *transponder in interrogation zone* and *no transponder in interrogation zone*. Such devices are often used as electronic anti-theft devices, like for Electronic Article Surveillance (EAS).

The electronic microchip on the transponder allows more complex operations as well, as it is able to store and process data. Those chips exist in a large variety and differ in memory size and supported commands. A typical memory size for an RFID chip is between 16 kB and 64 kB. The commands can extend from basic arithmetic operations to complex cryptographic algorithms.

We farther distinguish between active and passive transponders. Passive transponders are powered through the operating field of the reader and have no dedicated power supply. Active transponders have an additional power source, usually a battery. This battery is only used to power the chip on the transponder. Hence, the field of the reader is always used for transmitting data.

2.3 Operating Frequency, Transmission Range, and Coupling Method

The most important classification criteria of RFID systems are the operating frequency of the reader, the physical coupling method, and the transmission range for the system. Those three properties determine which RFID systems are suitable for different applications.

Operating Frequency. Frequencies are generally split into four ranges: Low Frequency (LF) at 30 kHz - 300 kHz, High Frequency (HF) at 3 MHz - 30 MHz, Ultra-High Frequency (UHF) at 300 MHz - 3 GHz, and microwave frequency at over 3 GHz. The operating frequency of RFID readers typically ranges from 135 kHz longwave to 5.8 GHz microwave.

Transmission Range. The transmission range of RFID systems is divided into three categories. *Close-coupling systems* operate with a very small range of less than 1 cm and are currently exclusively implemented on ID-1 smart cards. *Remote-coupling systems* operate on a distance up to 1.5 m. LF and HF are common operating frequencies for those systems. The last group are *long-range systems*, which operate on UHF or microwave frequency and typically work for ranges of up to 3 m. With active transponders even 15 m and above are possible.

Coupling Methods. The coupling between the reader and the transponder is achieved through an electric, a magnetic, or an electromagnetic field. Close-coupling systems use electric and magnetic fields, remote-coupling devices mainly use magnetic (inductive)

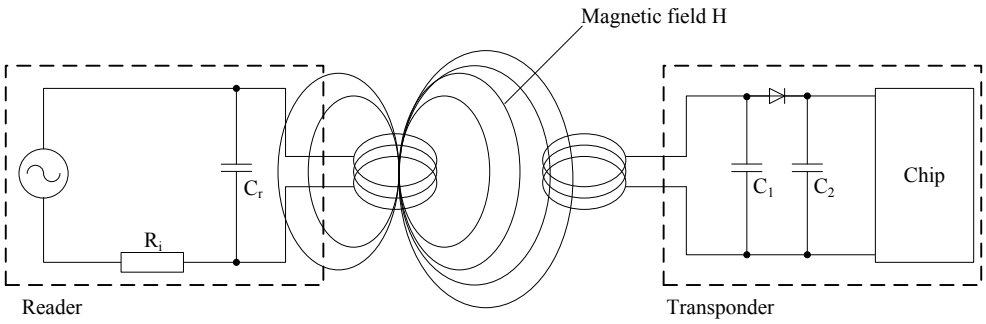


Figure 2.2: Power supply to an inductive coupled transponder [9].

coupling, and long-range coupling systems solely use electromagnetic waves. Note that coupling is responsible for power supply and data transmission.

An example for an inductive coupling is shown in Figure 2.2. The antenna coil of the reader generates a strong, high-frequency electromagnetic field. A small part of this electromagnetic field reaches the antenna coil of the transponder and generates a voltage by inductance. This voltage serves as power supply for the transponder. The capacitor C_r and the inductance of the reader antenna form a parallel resonance circuit that generates high currents and therefore reaches the required field strengths. Also the inductance of the transponder coil and C_1 form a parallel resonance circuit, which is tuned to the operating frequency, resulting in a maximal energy transfer.

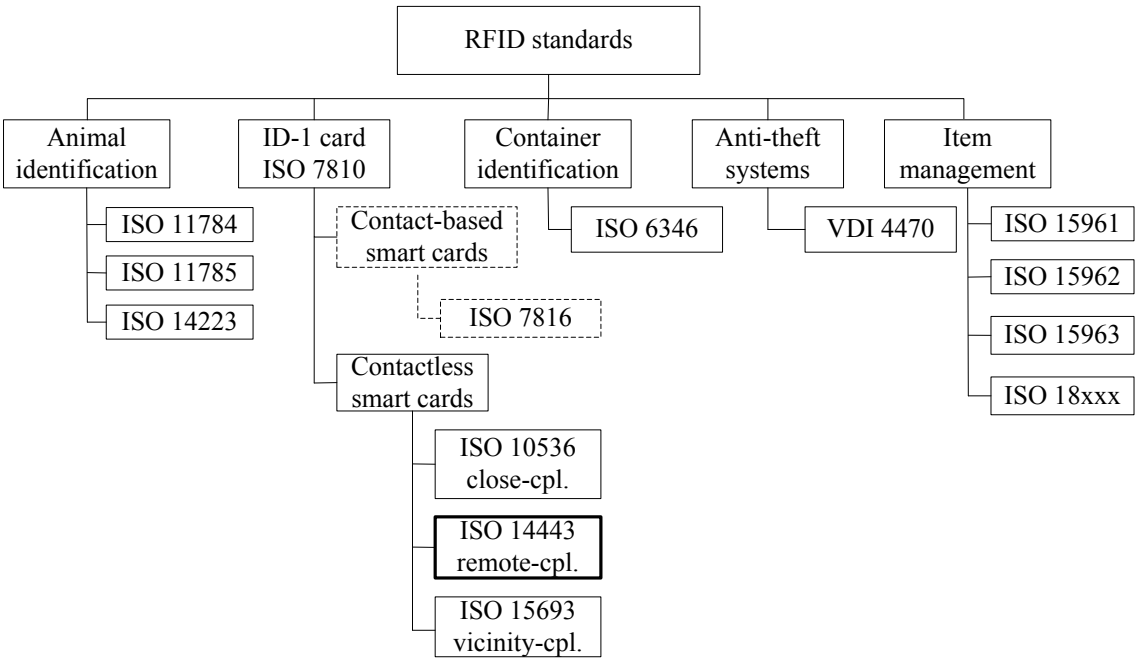


Figure 2.3: An overview of the most important standards for RFID systems.

2.4 RFID Standardization

Responsible for the development of RFID standards is the International Electrotechnical Commission (IEC) [16] that works together with the International Organization for Standardization (ISO) [20]. They release standards that are widely accepted and implemented by most manufacturers. We only refer to them as ISO standards as is customary in literature.

There exists a variety of different ISO standards published that address many different fields. An overview of the most important standards on RFID systems is given in Figure 2.3. Note that ISO 7816 [22] is in this figure for the sake of completeness, even though it is not an RFID standard.

One area of standardization is animal identification, which is specified by the ISO 11784, ISO 11785 and ISO 14223 standards. Other areas covered by ISO standards are container identification (ISO 6346) and item management (ISO 15961, ISO 15962, ISO 15963 and ISO 18xxx). There also exists a guideline developed by The Association of German Engineers (VDI) for anti-theft systems (VDI 4470) [46].

An important area is RFID systems for ID-1 smart cards, which are defined in the ISO 7810 specification [21]. There exists a variety of RFID standards for those cards that can be categorized in conactless smart cards and contact-based smart cards. Our work focuses on the ISO 14443 standard [23] for contactless smart cards, which is explained in detail in Chapter 4.

Chapter 3

Relay Attacks

This chapter provides background information about relay attacks. It first explains the basic concept and presents a number of different scenarios where the attacks are applicable. Afterwards, we analyze the components needed for a relay attack, namely a relay channel, a mole, and a proxy. Next, we estimate the total delay introduced during a relay attack. We finish this chapter with a discussion of relay attacks existing in the literature and possible countermeasures.

3.1 Basic Concept of Relay Attacks

A relay attack is any attack where the attacker relays information between two legitimate parties, without them being aware of the relay. The two parties are made believe that the information is exchanged on a legitimate channel, but it is actually going through a relay channel, created by the attacker.

Relay attacks are not limited to RFID systems. In fact, they have been around for a long time and are known to be effective for many applications. We first present some examples for relay attacks in other fields before explaining their application for RFID systems.

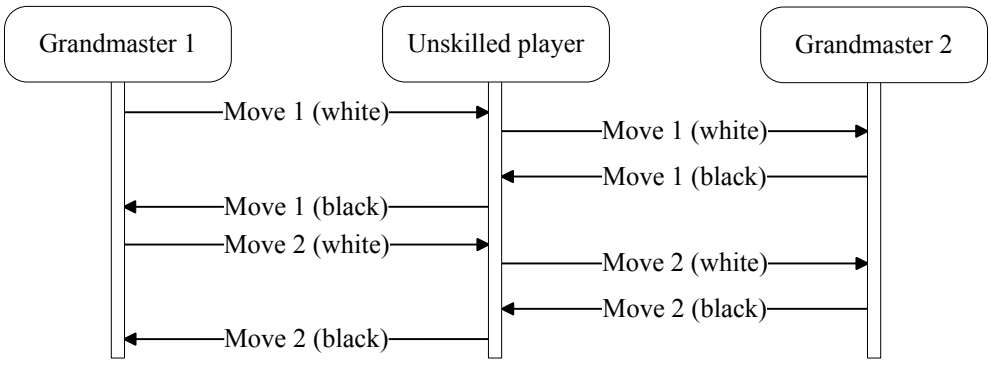


Figure 3.1: Basic concept of the grandmaster chess problem.

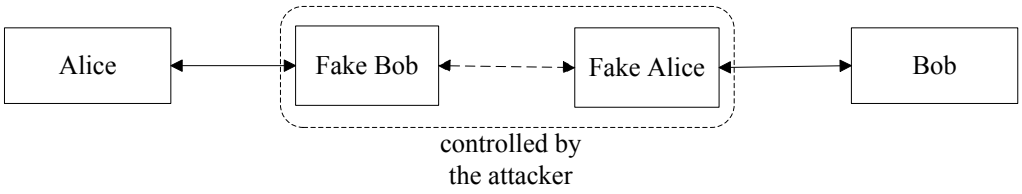


Figure 3.2: Basic concept of a man-in-the-middle attack.

3.1.1 Relay-Attack Scenarios

The concept of relay attacks already exists for several decades and can be performed in many different environments. One of the first relay attacks was introduced by Conway [6] as the grandmaster chess problem. An unskilled chess player could play a game of correspondence chess against two grandmasters simultaneously. If he plays white in one, and black in the other game, he just copies every move of the grandmasters and let them play against each other. He would then either win one game or draw both (see Figure 3.1).

A modern version of a relay attack is the Man-In-The-Middle (MITM) attack that is performed on cryptographic systems. The attacker establishes two connections, one to node *A* (Alice) and one to node *B* (Bob). He then relays every message that Alice sends to Bob and thereby impersonates Alice and vice versa. Alice and Bob think they are talking to each other, but the whole communication goes through the attacker. The basic concept of this attack is shown in Figure 3.2 [38].

Another scenario of a relay attack is a wormhole (or insider) attack on a wireless ad-hoc network (see Figure 3.3). Two or more malicious nodes work together to introduce false routing information. *Attacking node 1* makes node *A* assume that node *B* is closer than it actually is. Therefore, all communication between node *A* and node *B* goes through the attacking nodes. The attacker can then drop or alter packages at will and so manipulate the communication [7].

Note that both, the man-in-the-middle attack and the wormhole attack, can be performed in two different ways: active or passive. During a passive attack, the attacker does only relay the data without changing it, whereas during an active attack, the data might be changed while being relayed.

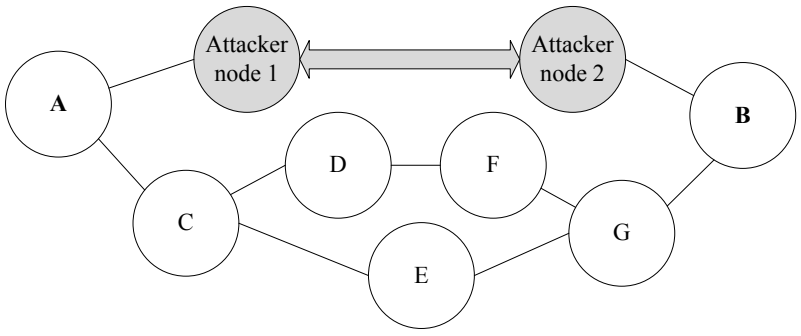


Figure 3.3: Basic concept of a wormhole attack.

3.1.2 Relay Attacks on RFID Systems

The concept of relay attacks can be adapted for RFID systems. One property of RFID systems is that they typically follow a request-answer (challenge-response) scheme. This implies that the reader initiates a data exchange and the transponder only answers to those requests. This is different compared to the attacks described so far and leads to slightly altered requirements for a successful attack on RFID systems.

Relay attacks on RFID systems involve two devices. One device impersonates the reader to the transponder and is called *mole*. The other device impersonates the transponder to the reader and is called *proxy*. Those two devices are connected by a relay channel.

A possible scenario of such an attack is presented in Figure 3.4. Transponder and mole as well as reader and proxy, are connected through a conventional RFID link. Mole and proxy communicate through a relay channel. The reader sends a challenge to the proxy, which only the transponder can solve. The proxy immediately forwards the challenge to the mole, which then sends it to the transponder. The transponder computes a correct response and sends it to the mole. Note that the transponder assumes that it is communicating directly with the reader. The mole then forwards the response to the proxy, which sends it to the reader. The reader assumes that a legitimate transponder in its reading range solved the challenge.

The scenario described above is a passive relay attack because no data is changed by the attacker. Especially for identification purposes, there is usually no need to change any of the data. An attacker could make the reader presume that the genuine transponder is right in front of it, although it is actually in another room or even further away. This attack is very powerful because there is no need to deal with the cryptographic primitive itself. A protocol can be secure in a mathematical sense, while a practical relay attack could still be performed, which makes such attacks powerful and hard to detect [14].

During an active relay attack, the attacker not only relays the data but also performs modifications. Therefore, the attacker needs to understand the relayed information and must be able to make useful changes. This is especially difficult in scenarios where the communication is encrypted. Moreover, altering the data might introduce additional delays, which makes the attack easier to detect [14].

So far we assumed that both parties, the reader and the transponder, are legitimate, which is called a *mafia fraud attack*. However, there are scenarios where the transponder is under the control of an attacker, which is called a *terrorist fraud attack*. This attack could be used to fake the location of a person, for example. The advantage of this attack is that the genuine transponder does not need to operate according to the defined protocol. It might reveal secrets before time, so the attacker can already relay them. This makes it easier for the attacker and harder to find countermeasures. Note that the terrorist fraud

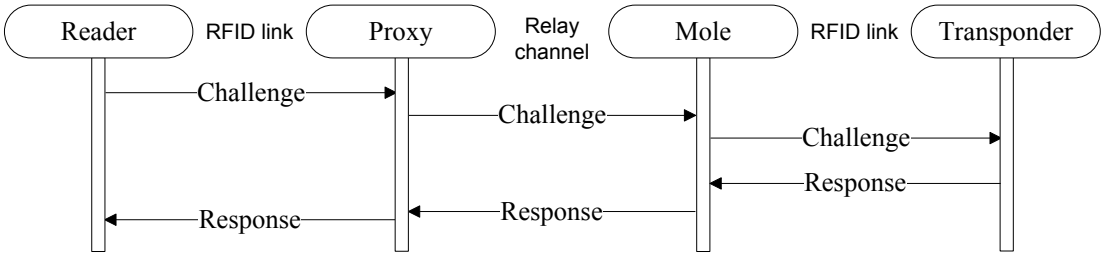


Figure 3.4: Possible scenario for a relay attack on an RFID system.

attack assumes that the genuine transponder does not share its private (secret) key, which would make the attack trivial [30].

3.2 Relay Channel

The relay channel is a very important part of a successful attack. It must be able to relay data fast enough to meet the timing constraints of the reader. Therefore, it must have a low delay and a sufficient data rate.

3.2.1 Physical Boundaries

The maximum speed that data can move is the speed of light in a vacuum [1]. Therefore, an ideal relay channel would forward data with the speed of light and without any delay. Assuming such a channel exists, this would lead to the results shown in Table 3.1. The Round-Trip Time (RTT) denotes the time it takes for the data to travel the distance twice. It is calculated in Equation 3.1 with d being the distance and c being the speed of light.

$$RTT = \frac{1}{c} \times 2d \tag{3.1}$$

Table 3.1: Round-trip times of a relay channel with the speed of light.

Distance [m]	RTT [ns]
1	6.7
10	66.7
100	667.1
1 000	6 671.3
10 000	66 712.8

3.2.2 Relay-Channel Medium

There are a number of different mediums that can be used for a relay channel. Those mediums might be more or less useful in different scenarios. Also a relay channel could very likely be a combination of several different mediums.

The propagation speed of electromagnetic waves can be expressed by the velocity factor v_f. The velocity factor defines the ratio between the speed of the wave in a certain medium v_p, and the speed of light in vacuum [3]. The velocity factor can be calculated by

Table 3.2: Velocity factors of different mediums.

Medium	$\mathbf{v_f}$
Wireless radio (air) / Bluetooth	1.00
Coax [44]	0.78
Multi-mode fibre (glass) [11]	0.67
Twisted pair (copper) [45]	0.65 - 0.70

[1] $c = 299\,792\,458$ m/s $\sim 3 \times 10^8$ m/s.

$$v_f = \frac{v_p}{c}. \tag{3.2}$$

The velocity factor cannot be defined for a medium in general because it varies for different manufacturers. Table 3.2 shows mediums with common values for the velocity factor.

3.3 Relay-Attack Devices

A successful relay attack requires fast relay devices. Those two devices are the mole and the proxy. We look at characteristics the devices should have and other properties that should be taken into consideration. Note that we assume that the relayed data is digitized and not relayed as an analogue signal.

First of all, both devices need to work very fast. If the devices take too much time for their operations, the reader might detect the delay and recognize that it is being attacked. Because the relay channel already introduces a delay, the delay due to the device hardware should be kept to a minimum.

The delay firstly depends on the frequency of the processing unit being used, for example a microchip. It also depends on the communication interface of the relay device and its data rate. The communication interface may have a lower data rate than the relay channel, which would introduce an additional delay.

In order to be able to communicate with genuine devices, both attacking devices need to implement the applicable RFID protocol. They do not need to support the full set of commands, but enough to perform an attack. Note that an attacker is not bound to any limitation for power or frequency. Therefore, he could increase the signal power to achieve a higher transmission range or increase the operating frequency to achieve a higher processing throughput [14].

The physical appearance should be inconspicuous and as small as possible. The attacker must bring the mole in proximity to the transponder without being spotted. So it should fit in a briefcase, purse, or even a wallet. It could also be built into a mobile phone or any other electronic device. The proxy should ideally reassemble a real RFID tag. This makes it unsuspicious if held in front of a genuine reader. However, the requirements for the physical appearance strongly depend on the attack scenario.

3.4 Relay-Channel Protocols

This section presents and analyzes some existing protocols that could be used for the relay channel. We present only some of the most common used protocols and differentiate between wired and wireless communication. Although the protocols are analyzed as they are defined, an attacker might not have to implement existing specifications. He could remove parts or all of the overhead, depending on the scenario.

3.4.1 Wired Protocols

RS-232

The Recommended Standard 232 (RS-232) is an interface standard that was defined by the Electronic Industries Association (EIA), Bell-Labs, and communication system man-

Table 3.3: Effective data rate and delay per bit for typical RS-232 baud rates.

Data rate$_{\text{eff}}$ [bit/s] [5]	**Delay$_{\text{bit}}$ [µs/bit]**	**Data rate$_{\text{eff}}$ [bit/s] [5]**	**Delay$_{\text{bit}}$ [µs/bit]**
110	9 090.9	19 200	52.1
300	3 333.3	38 400	26.0
600	1 666.7	57 600	17.4
1 200	833.3	115 200	8.7
2 400	416.7	230 400	4.3
4 800	208.3	460 800	2.1
9 600	104.2	921 600	1.1

ufacturers in 1969. It defines serial binary data exchange between two devices, which are usually connected through a DB-25 connector [5].

The data rate of RS-232 is defined by the baud rate, which is given in bits per second. Both devices need to operate with the same baud rate to communicate [5]. The baud rate is also the effective data rate ($data\ rate_{eff}$) as no protocol overhead exists for RS-232. Knowing the effective data rate we can calculate the average delay per bit ($delay_{bit}$) by

$$\text{delay}_{\text{bit}} = \frac{1}{\text{data rate}_{\text{eff}}}. \tag{3.3}$$

Note that this equation applies to all following protocols as well. Typical baud rates and the corresponding delays per bit for RS-232 are presented in Table 3.3.

Ethernet

Ethernet is the most common used protocol for wired communication in Local Area Networks (LANs). It is a frame-based protocol that works on the physical and the data-link layer of the Open System Interconnection (OSI) model [18].

The data rate ($data\ rate_{wire}$) of Fast Ethernet is 100 Mbit/s, which is called 100BASE-TX for twisted pair cables. A Fast Ethernet block is shown in Figure 3.5. Ethernet defines the

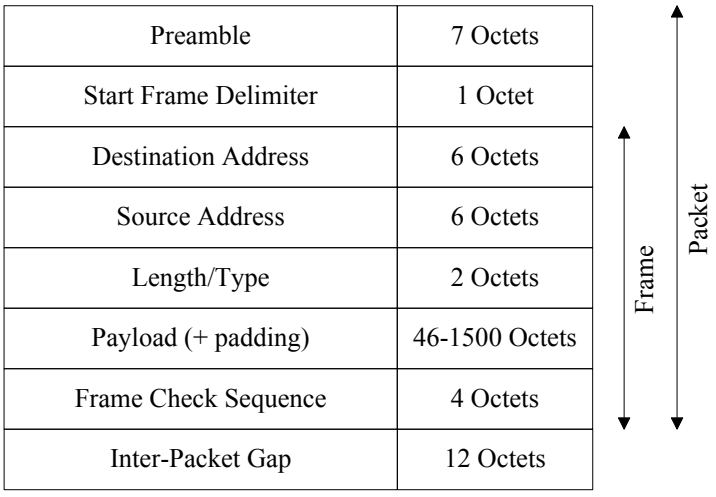


Figure 3.5: The structure of an Ethernet block [18].

Table 3.4: Effective data rate and delay per bit for Fast Ethernet with different payloads.

Payload [byte]	Overhead [byte]	Data rate_{eff} [Mbit/s]	$\text{Delay}_{\text{bit}}$ [ns]
1	83	1.19	801.00
8	76	9.52	100.00
46	38	54.76	17.41
512	38	93.09	10.24
1500	38	97.53	9.78

minimum size of a 100 Mbit Ethernet frame with 64 bytes. Therefore, if the payload is less than 46 bytes, an additional padding field is added. The standard overhead, including the gap between two packets, is 38 bytes. However, if the payload is less than the minimum of 46 bytes, the overhead increases due to the padding field. The effective data rate ($data\ rate_{eff}$) depends on the payload and is calculated by

$$\text{data rate}_{\text{eff}} = \frac{\text{payload}}{\text{total block size}} \times \text{data rate}_{\text{wire}}. \tag{3.4}$$

Values of $data\ rate_{eff}$ and $delay_{bit}$ (calculated using Equation 3.3) for different payloads are presented in Table 3.4. The *overhead*, which includes the inter-packet gap and the padding field, is calculated by

$$\text{overhead} = \text{total block size} - \text{payload}. \tag{3.5}$$

3.4.2 Wireless Protocols

Bluetooth

Bluetooth is a wireless short range communication system. The data rate of a Bluetooth channel is defined with 780 kbit/s. The most commonly used protocol version is 2.1, which defines an Enhanced Data Rate (EDR). EDR versions of the protocol use Differential

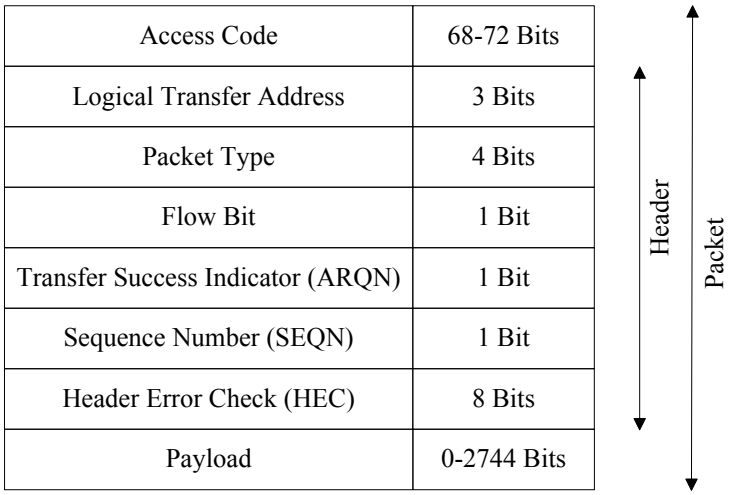


Figure 3.6: The structure of an ACL packet used for Bluetooth communication [43].

Table 3.5: Effective data rate and delay per bit for different Bluetooth packet types [43].

Type	Data rate_{eff} [kbit/s]	$\text{Delay}_{\text{bit}}$ [ns/bit]
2-DH1	346	2 826
2-DH3	1 174	831
2-DH5	1 449	674
3-DH1	531	1 838
3-DH3	1 766	553
3-DH5	2 178	448

Quaternary Phase-Shift Keying (DQPSK) and 8 Phase-Shift Keying (8PSK), and therefore support data rates up to 2 178 kbit/s [43].

Bluetooth uses Asynchronous Connection-Less (ACL) packets for data transfer, which can be seen in Figure 3.6. Therefore, the overhead of a single Bluetooth packet is between 86 bits and 90 bits. Table 3.5 shows effective data rates ($data\ rate_{eff}$) for different bluetooth packet types [43] and the resulting delay per bit, calculated by Equation 3.3.

The first number of the packet type denotes the modulation (2 for DQPSK and 3 for 8PSK), DH indicates that no forward error correction is used, and the last number (1, 3 and 5) indicates how many time slots are used by one packet [43].

802.11 Wireless LAN

The most commonly used protocol for wireless computer networks is defined in the Institute of Electrical and Electronics Engineers (IEEE) 802.11 specification. There exist many different appendices, but in 2007 they where merged into the 802.11-2007 standard [17]. We analyze the former appendix 802.11g, which supports a data rate ($data\ rate_{medium}$) up to 54 Mbit/s.

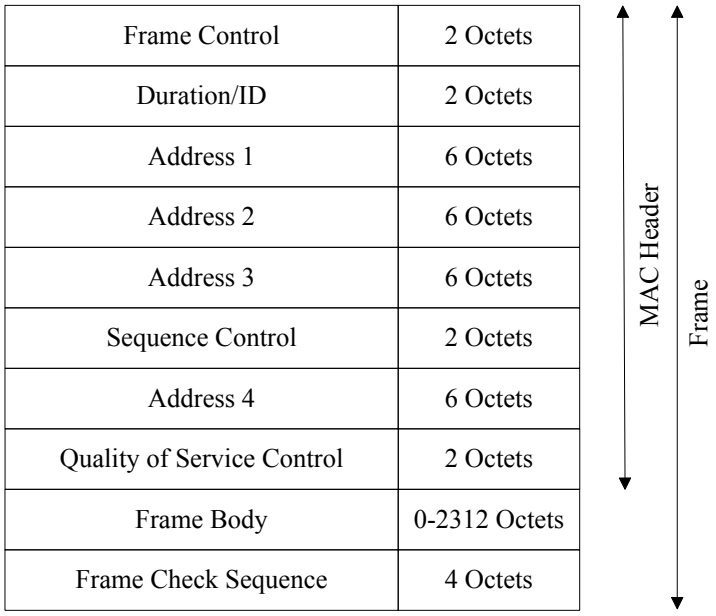


Figure 3.7: The structure of an 802.11 Frame [17].

Table 3.6: Effective data rate and delay per bit for 802.11 wireless LAN for different payloads.

Payload [byte]	Overhead [byte]	Data rate_{eff} [Mbit/s]	$\text{Delay}_{\text{bit}}$ [ns/bit]
1	36	1.46	653.0
8	36	9.82	97.0
46	36	30.29	31.5
512	36	50.45	18.9
1 024	36	52.17	18.3
2 312	36	53.17	17.9

A 802.11 frame is shown in Figure 3.7. The overhead adds up to a total of 36 bytes and the payload has a maximum size of 2 312 bytes. The effective data rate ($data\ rate_{eff}$) can be calculated by

$$\text{data rate}_{\text{eff}} = \frac{\text{payload}}{\text{total frame size}} \times \text{data rate}_{\text{medium}}. \tag{3.6}$$

Table 3.6 presents different values for the payload and the resulting $data\ rate_{eff}$ as well as the average delay per bit ($delay_{bit}$), which is calculated by Equation 3.3.

RFID protocols

Although RFID is not a protocol itself, there exist several ISO specifications that define protocols for many different applications. For an overview of the existing ISO standards for RFID systems we refer to Section 2.4. Here we look at the ISO 14443 standard for contactless smart cards because it is used in the experiments later.

The used RFID protocol influences the delay because during a regular RFID communication, there exists only one RFID link: between the reader and the transponder. However, during an attack, there exist two RFID links: one between mole and transponder and one between reader and proxy. This additional RFID link introduces an extra delay, which depends on the effective rate ($data\ rate_{eff}$) used for communication. The resulting average delays per bit ($delay_{bit}$) for the four data rates defined in the ISO 14443 standard are presented in Table 3.7.

Table 3.7: Data rates defined in ISO 14443 and the resulting delays per bit.

Data rate [kbit/s]	Delay per bit [µs]
106	9.21
212	4.61
424	2.30
848	1.15

3.5 Total-Delay Estimation

In the previous sections, we analyzed the different components of a relay attack separately. In this section, we combine the results and give an estimation of the total delay introduced

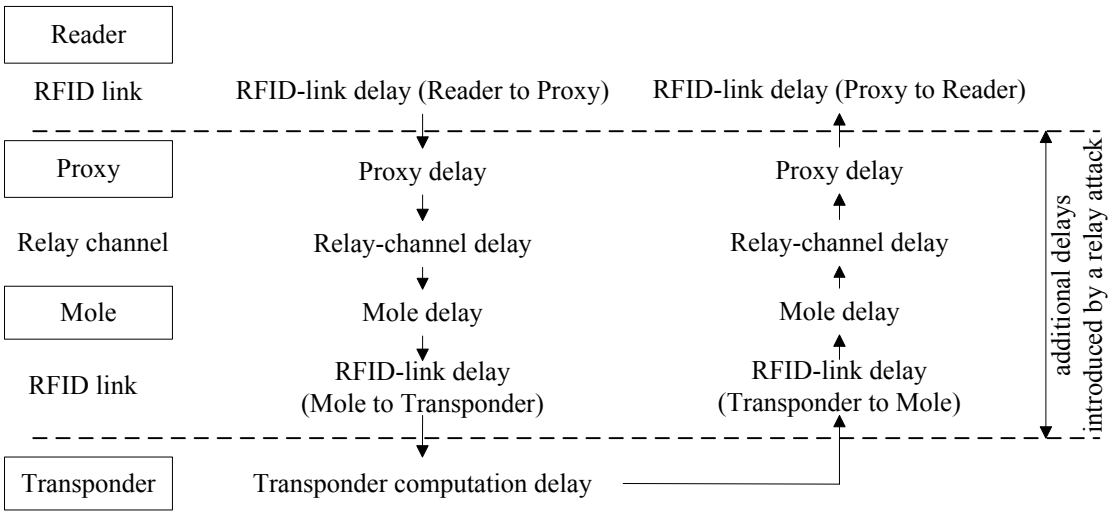


Figure 3.8: Delays introduced during a relay attack.

during a passive relay attack. Note that an active relay attack would introduce additional delay at the proxy and/or the mole.

3.5.1 Total-Delay Derivation

In order to estimate the additional delay introduced by a relay attack, we look at Figure 3.8. The communication between the reader and the proxy as well as the computation time of the transponder can be ignored, as those delays would also occur during a regular communication. For the introduced delay, we sum up the remaining delays and multiply them by two because the data needs to travel the distance twice. The resulting formula can be written as

$$\begin{aligned} \text{introduced delay} = (&\text{proxy delay}+ \\ &\text{mole delay}+ \\ &\text{relay-channel delay}+ \\ &\text{RFID-link delay}) \times 2. \end{aligned} \tag{3.7}$$

We can further split up the *relay-channel delay* into a *protocol delay* and a *propagation delay*. The protocol delay depends on the protocol p used, and the propagation delay depends on the length of the relay channel d_c between mole and proxy as well as on the velocity factor v_f of the relay medium. Therefore, the relay channel delay can be calculated by

$$\text{relay-channel delay} = \text{protocol delay}(p) + \text{propagation delay}(d_c, v_f). \tag{3.8}$$

We can also split up the *RFID-link delay* into a *protocol delay* and a *propagation delay*. The propagation delay depends on the distance between mole and transponder (or proxy and reader) d_r. The *protocol delay* for ISO 14443 depends on the used data rate r and is defined by

$$\text{RFID-link delay} = \text{protocol delay}(r) + \text{propagation delay}(d_r). \tag{3.9}$$

3.5.2 Estimation of Proxy Delay and Mole Delay

The delay estimation of the relay devices is difficult because there exist various factors. We assume the attacking devices to work with a clock frequency f of 16 MHz, which could for example be an ATMega128 [2]. Receiving and sending a single bit can probably be done within only a few clock cycles, as only a few register entries need to be shifted. The delay for a single bit within one device, assuming it takes ten clock cycles to forward it, can be calculated by

$$\text{device delay} = \frac{\text{clock cycles}}{f} = \frac{10}{16\,MHz} = 625\,\text{ns/bit}. \tag{3.10}$$

Note that while the genuine devices always work on the operating frequency, which is 13.56 MHz for ISO 14443, the attacking devices are not bound to this requirement. Also note that devices performing an analogue relay could reach lower delays.

3.5.3 Lower Bound for the Total Delay Introduced

This section is not meant to estimate the delay for a real relay attack implementation. It should rather give a lower bound for the delay introduced by a relay attack. Therefore, we use assumptions that might not be practical for real implementations.

We estimate the delay for Fast Ethernet relay channel as described in Section 3.4, as it introduces the lowest protocol delay. As mentioned we try to define a lower bound for the delay, and therefore assume a maximum payload of 1 500 byte. Furthermore, we assume a velocity factor for the Ethernet cable of 0.7. For a relay-channel length d_c of ten meters, an RFID data rate of 848 kbit/s, and an RFID-link distance of 0.2 m, the resulting delay per bit is calculated by

$$\begin{aligned}\text{introduced delay} &= (\text{proxy delay} + \text{mole delay} + \text{relay-channel protocol delay} \\ &\quad + \text{relay-channel propagation delay} + \text{RFID-protocol delay} \\ &\quad + \text{RFID-propagation delay}) \times 2 \\ &= (625\,\text{ns/bit} + 625\,\text{ns/bit} + 9.78\,\text{ns/bit} + \\ &\quad \frac{10\,\text{m}}{\text{c}} \times \frac{1}{0.7} + 1.15\,\mu\text{s/bit} + \frac{0.2\,\text{m}}{\text{c}}) \times 2 \\ &= 4.92\,\mu\text{s/bit}.\end{aligned} \tag{3.11}$$

Table 3.8: Total introduced delay per bit for different RFID data rates and relay-channel lengths.

	RFID rate [kbit/s]			
Distance [m]	106	212	424	848
1	20.96 µs	11.74 µs	7.14 µs	4.83 µs
10	21.04 µs	11.83 µs	7.22 µs	4.92 µs
100	21.90 µs	12.69 µs	8.08 µs	5.78 µs
1 000	30.48 µs	21.26 µs	16.66 µs	14.35 µs
10 000	116.25 µs	107.04 µs	102.43 µs	100.13 µs

The most significant delays result from the RFID link, the devices and the propagation delay that depends on the channel length. Table 3.8 shows the total introduced delay for different RFID-link rates and relay-channel lengths. We see that the longer the relay channel is, the less important the RFID-link rate and the device delay become because they are constant.

3.6 Relay Attacks in Literature

Kfir and Wool [28] already described a possible setup for a relay attack on RFID systems in 2005. One of the first practical attack implementations was done by Hancke in 2005 [12]. He implemented a relay attack connecting proxy and mole through an UHF antenna and relayed the analogue data between the two devices. He reached an introduced delay of only 15-20 µs. Note that such small delays are only possible if the communication is relayed as analogue data. If the data is digitized and forwarded as binary data, the processing time introduces a significant additional delay.

Recently, Francis et al. [10] presented a practical relay attack on NFC mobile phones. They used two NFC mobile phones as proxy and mole and established a Bluetooth link in order to relay data in the 2.4 GHz frequency band. However, although the attack worked as a proof of concept, it lacks in the fact that the required Unique IDentifier (UID) from off-the-shelf mobile phones cannot be changed, which restricts the attack to only a limited number of applications.

Weiss [47] obtained similar results during his experiments. He implemented a relay attack using NFC mobile equipment in the course of his master thesis and reached delays of about 50 ms. His setup included a USB-based NFC device as a relay proxy that requires the connection with a PC, which might not be practical for a real life scenario.

3.7 Countermeasures

The danger of relay attacks is that conventional application-layer security does not address those kind of attacks. The cryptographic primitives used are not designed to detect whether the information is relayed or not. Therefore, different countermeasures against relay attacks need to be found.

For wormhole attacks exists a variety of countermeasures. However, those countermeasures use additional information like the location of the nodes [39], assume highly synchronized clocks [15], or several nodes monitor each others traffic [29]. Therefore, those countermeasures cannot be applied to RFID systems. RFID systems operate with a very limited amount of information. There exist no location awareness or synchronized time clocks between sender and receiver, which makes it especially hard to prevent relay attacks.

We analyze four different classes of countermeasures, namely physical protection, additional verification, timing constraints, and distance bounding.

3.7.1 Physical Protection

One countermeasure against relay attacks is the physical protection of the RFID transponder. A Faraday cage can shield it from electromagnetic waves [30] and therefore make it not accessible for an attacker. However, this introduces significant inconvenience for

the user, and moreover does not protect if the attacker is in permanent or temporary possession of the transponder.

3.7.2 Additional Verification

For security reasons, RFID readers can request additional information from the owner of the transponder. Implementing such a Two-Factor Authentication (2-FA) would largely protect against relay attacks. The additional information could be a Personal Identification Number (PIN) or the fingerprint of the verifier. Also a picture of the verifier could be used. However, this leads to additional delays in RFID applications, which makes it more inconvenient for users [14].

3.7.3 Timing Constraints

As shown in Section 3.5, a relay attack introduces additional delays. Therefore, the answer to a reader request is delayed compared to a genuine response. By introducing time-outs into the communication protocol, the reader can ensure that the transponder is really in transmission range. However, those time-outs largely depend on the used transponder hardware and may vary significantly. Time-outs can also cause false rejections, which makes them inconvenient for the user [14].

3.7.4 Distance Bounding

Distance-bounding protocols try to determine the physical distance between the reader and the transponder. The basic idea is to create a challenge that only the genuine transponder can solve (similar to zero-knowledge proofs of knowledge). The transponder should take a fixed, usually very short amount of time, to solve this challenge. Therefore, the reader can compute a very good estimate of how far away the tag actually is, based on the Round-Trip Time (RTT) [14].

Distance bounding right now is the most promising countermeasure against relay attacks. It can be implemented without significantly increasing the costs of RFID systems and does not cause any additional inconvenience for the user. Hancke and Kuhn proposed a distance bounding protocol that uses an Ultra-WideBand (UWB) channel [13]. The protocol uses an UWB channel because Hancke and Kuhn state that accurate distance bounding is not possible using the 13.56 MHz communication link. However, a UWB channel is not available in current RFID systems and would cause significant additional costs.

Reid et al. [42] and Munilla and Peinado [31] proposed protocols that expect the response of the transponder in predefined clock cycles. This approach is vulnerable to overclocking [14] and might cause problems with compatibility. Implementations on the transponder may vary significantly and sending the response within a defined clock cycle might not always be possible.

Chapter 4

The ISO 14443 Standard

The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) developed the ISO/IEC 14443 standard that describes the operation method and operation parameters for proximity-coupling smart cards. Those cards operate in a range of up to approximately 10 cm. We refer to the standard only as ISO 14443 and describe the newest version of the standard released in 2010.

The standard labels the reader as a Proximity Coupling Device (PCD) and the transponder as a Proximity Integrated Circuit Card (PICC). We use the terms PCD and PICC whenever we refer to the reader and the transponder of an ISO 14443 system.

The standard is split into four parts. The first part covers the physical characteristics of the PICC [24]. The second part specifies the characteristics of the fields to be provided for power and bi-directional communication between the PCD and the PICC [25]. The third part defines the routines for the initialization of the PICC as well as an anticollision routine for multiple PICCs [26]. The fourth and last part defines a half-duplex block-transmission protocol satisfying special needs for contactless environments. It also defines the activation and deactivation sequence of the transmission protocol [27]. Note that the higher parts are designed to be used in conjunction with the lower parts.

The standard also defines two types of PICCs, A and B. Compliant PCDs need to support both types, whereas compliant PICCs only need to support one type of communication.

In the following chapter, all four parts are be described in detail with special focus on the elements that are important concerning relay attacks.

4.1 Part 1: Physical Characteristics

The first part defines the physical characteristics of an ISO 14443 smart card. It defines the antenna dimension with a maximum of 86 x 54 x 3 mm. However, most antennas are designed to fit on ISO/IEC 7816 ID-1 smart cards. ID-1 smart cards are similar to credit and telephone cards and are defined with a size of 85.72 x 54.03 x 0.76 mm. [9][22]

4.2 Part 2: Radio-Frequency Power and Signal Interface

This part specifies the characteristics of the fields to be provided for power and bi-directional communication between PCD and PICC. It specifies for both PICC types the power transfer, the signal interface, and the modulation technique.

The power for the PICC is provided by the PCD. Therefore, the PCD creates a magnetic field with a frequency of 13.56 MHz ± 7 kHz. The operating field strength is defined with a minimum H_{min} of 1.5 A/m and a maximum H_{max} of 7.5 A/m.

The signal interface defines four different data rates for communication. All four rates depend on the operating frequency f_c and are defined as $f_c/128$, $f_c/64$, $f_c/32$ and $f_c/16$, which result in 106 kbit/s, 212 kbit/s, 424 kbit/s, and 848 kbit/s, respectively.

The PCD uses Amplitude Shift Keying (ASK) to communicate with the PICC. Type A PCDs use 100 % ASK with modified Miller coding, whereas Type B PCDs use 10 % ASK with Non-Return-to-Zero (NRZ) coding. The PICC uses load modulation to respond to the PCD, applying different coding techniques, depending on type and data rate.

4.3 Part 3: Initialization and Anticollision

The third part of the standard defines how the polling for new PICCs is handled. It also specifies the initial phase of the communication and methods to deal with multiple PICCs in range (anticollision). For each type A and B, there exist different definitions. In the following, we only describe the specification for PICC Type A.

4.3.1 Frame Delay Time

The standard defines a certain frame format and frame timing for the initialization and anticollision. Note that the bit representation and the coding are defined in ISO 14443-2 and are not described in this part.

ISO 14443 communication follows a request-response scheme, which makes it mandatory that frames are always transmitted in pairs. The first frame is sent by the PCD to the PICC and the second frame by the PICC to the PCD. The delay between those two frames is defined as the Frame Delay Time (FDT). Figure 4.1 illustrates the basic communication concept.

The FDT measurement starts at the last rising edge of the frame. If the last bit of the frame is a logic 1 it starts earlier than if the last bit is a logic 0 because the rising edges occur at different times. Therefore, the standard defines two sets of FDTs, which are shown in Table 4.1. Note that commands used during the initialization (REQA, WUPA,

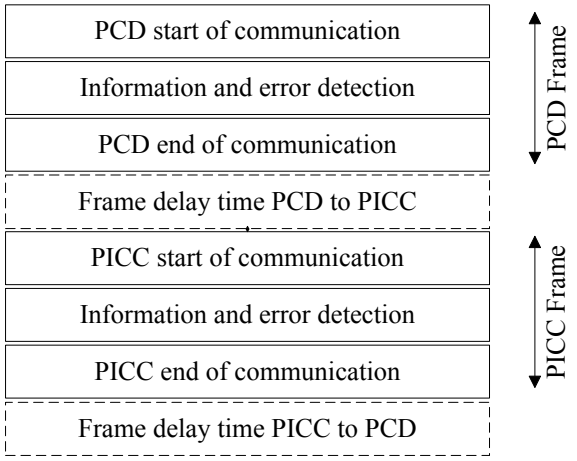


Figure 4.1: Sequence of a frame pair as defined by ISO 14443.

Table 4.1: Frame delay times for transmissions from PCD to PICC at an operating frequency of 13.56 MHz [26].

Command	**FDT**	
	last bit = (1)b	last bit = (0)b
REQA, WUPA, ANTICOLLISION, SELECT $f_c/128$	= 91.15 µs	= 86.43 µs
All other commands		
$f_c/128$	≥ 91.15 µs	≥ 86.43 µs
$f_c/64$	≥ 86.43 µs	≥ 84.07 µs
$f_c/32$	≥ 84.07 µs	≥ 82.89 µs
$f_c/16$	≥ 82.89 µs	≥ 82.30 µs

ANTICOLLISION, SELECT) require a fixed FDT. Those commands are explained later in this section. They are transfered at a rate of $f_c/128$ and allow a tolerance of -1/f_c to (+ 0.4 µs + 1/f_c).

4.3.2 Frame Formats

The ISO 14443 standard defines three types of frames: *short frames*, *standard frames*, and *bit-oriented anticollision frames*. They all serve different purposes and are briefly explained in the following paragraphs.

Short Frame. A short frame is shown in Figure 4.2 and consists of a part *S* (start of communication), 7 data bits, and a part *E* (end of communication). It is used to initiate communication. Note that all frames start with the Least Significant Bit (LSB) and end with the Most Significant Bit (MSB).

Standard Frame. A standard frame is shown in Figure 4.3 and consists of a part *S* (start of communication); *n* data parts, each consisting of eight data bits and one odd parity bit *P*; and a part *E* (end of communication). Note that the number of data parts is larger or equal to one. The frame shown is a PCD standard frame. PICC frames are constructed in a similar way and we refer to part 3 of the standard for further details [26].

Bit-Oriented Anticollision Frame. The bit-oriented anticollision frame is constructed as a standard frame with seven data bytes. It is only used during the anticollision loop, which is explained in the following section.

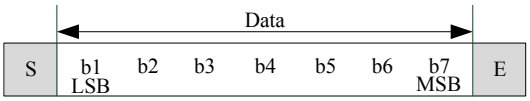


Figure 4.2: The structure of an ISO 14443 short frame [26].

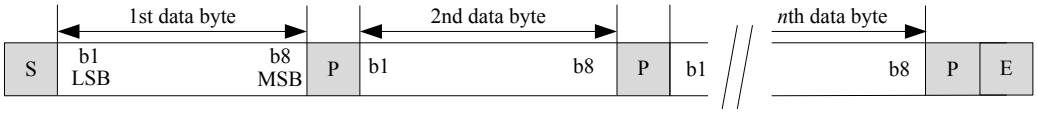


Figure 4.3: The structure of an ISO 14443 PCD standard frame [26].

4.3.3 Anticollision

Part 3 of ISO 14443 defines an anticollision procedure for Type A PICCs that is based on a dynamic binary-tree search algorithm. The PCD can select one of multiple PICCs in its range by their Unique IDentifier (UID). First we explain some basics of this algorithm before presenting the algorithm itself.

Basics of Anticollision

If multiple PICCs are in range, the PCD selects a single PICC by its UID. In order to retrieve the UID of a single PICC, the PCD has to run a procedure called the anticollision loop. Depending on the size of the UID, which can be four, seven, or ten bytes long, it runs this loop one, two, or three times, respectively. We call the number of times the anticollision-loop procedure is executed the *cascade level* [9].

Before the actual anticollision loop starts, the PCD sends a Request A (REQA) command to all PICCs in range. The PICCs respond with an Answer To Request A (ATQA), which contains the length of the UID and an information field that tells the PCD how many cascade levels are required [9].

One other requirement must be met for Type A anticollision to work. If multiple PICCs transmit information synchronously, the PCD must be able to determine exactly at which bit a collision occurred. Manchester coding defines the value of a bit by a level change (transition) within a bit window. A positive transition represents a logic 0 and a negative transition a logic 1. If two or more PICCs transmit different logic values for the same bit, the level does not change and stay high for the entire bit. As this state is not allowed in Manchester coding, the PCD recognizes the collision. Note that all PICCs must send their bits synchronously for this procedure to work [9].

Anticollision Loop

The flowchart for the anticollsion loop can be seen in Figure 4.4. It starts with the PCD sending an ANTICOLLISION command. It constists of a Select Code (SEL) and a Number of Valid Bits (NVB) code. SEL contains two bits indicating the current cascade level. NVB consists of two parts. The first part is the number of valid bytes and the second part is the number of additional valid bits. Therefore, in the first ANTICOLLISION command NVB equals '20', as only the two bytes (SEL and NVB) are valid.

When the PICCs receive the ANTICOLLISION command, they send their UID for the cascade level defined in SEL. Figure 4.5 shows which part of the UID is transmitted depending on the UID length and the cascade level. The UID is split into up to ten parts (*uid0 - uid9*). Note that *CT* is a cascade tag (88h) and cannot be used in *uid0* of 4-byte UIDs or *uid3* of 7-byte UIDs. Also note that every message ends with a Block Check Character (BCC).

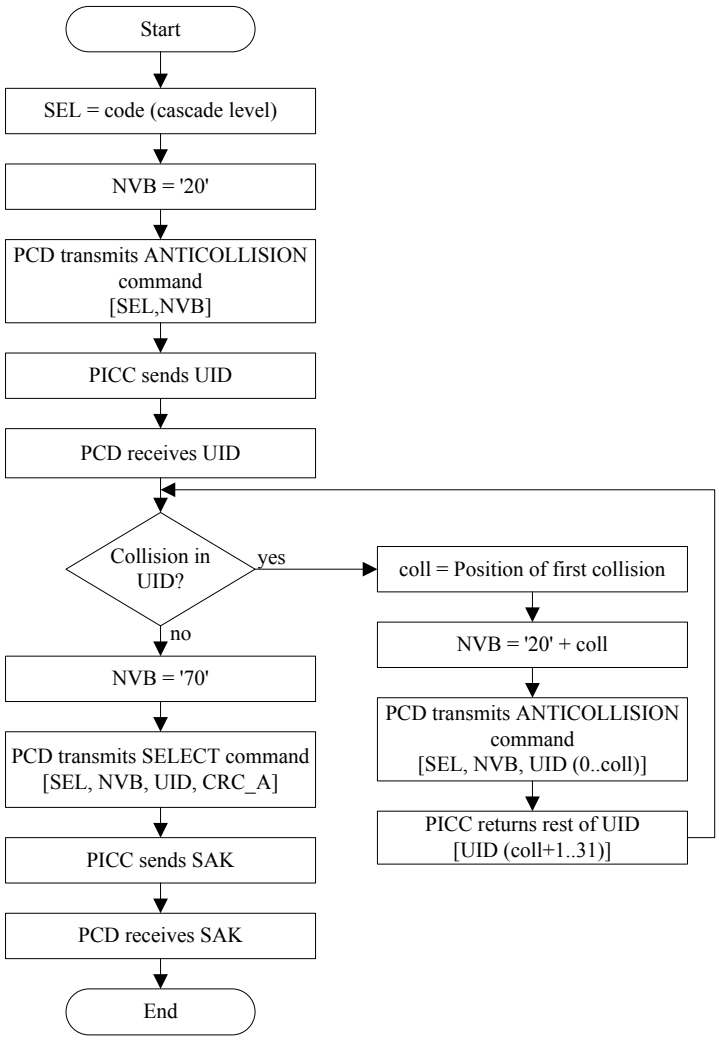


Figure 4.4: PICC Type A anticollision loop for a single cascade level [26].

Note that all PICCs transmit their UIDs synchronized. Therefore, a collision occurs at the first differing bit of their UIDs. This collision is detected by the PCD, which then calculates a new value for NVB. The PCD now knows that all UIDs are the same before the colliding bit. It then sends a new ANTICOLLISION command with the already valid part of the UID, plus a 1-bit or 0-bit. After that, the PCD gets a response only from the PICCs that still fit the received UID. Those PICCs answer with the remaining part of their UIDs. This loop is repeated up to 32 times until no collision is detected.

If no collision is detected, NVB is set to 7 bytes, therefore announcing that a SELECT command is coming. This SELECT command now contains the UID of the selected PICC at the current cascade level, plus a Cyclic Redundancy Check A (CRC_A). In the last step, the selected PICC sends a Select Acknowledge (SAK) command to the PCD, which ends the anticollision loop for the current cascade level. Note that this loop has to be executed for all three cascade levels if necessary. Also note that PICCs that are not selected at a lower cascade level do not answer to following anticollision commands in higher cascade levels.

For the precise coding specifications for NVB, SAK, SEL, REQA and ATQA, we refer

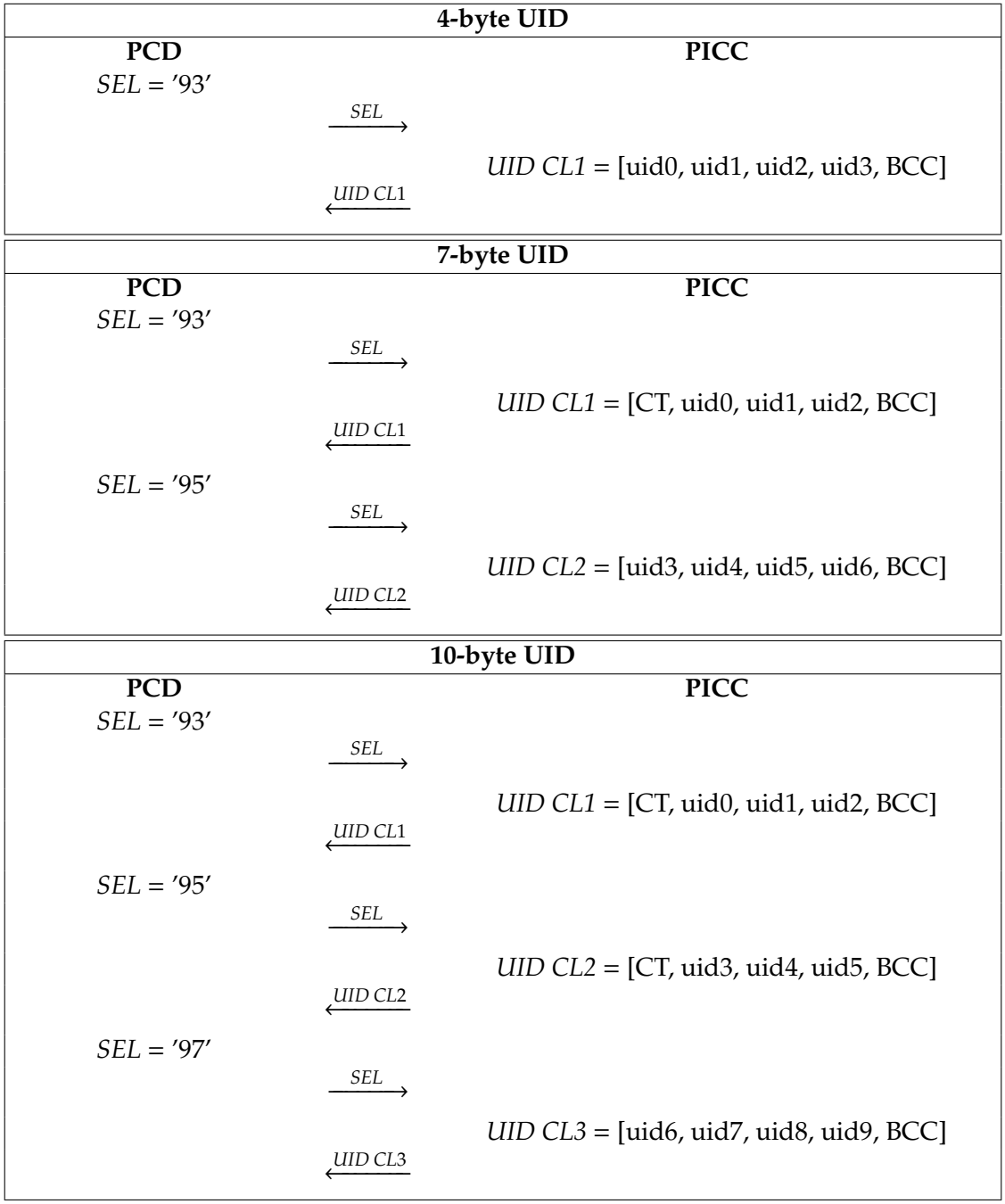


Figure 4.5: The distribution of the UID for different cascade levels [26].

to the ISO 14443 standard part 3 [26].

4.3.4 PICC States

The ISO 14443 standard also contains a state diagram that defines all states and transitions between those states for a PICC during activation and anticollision. This state diagram can be seen in Figure 4.6.

We see that every PICC has two different READY and ACTIVE states. One set is used

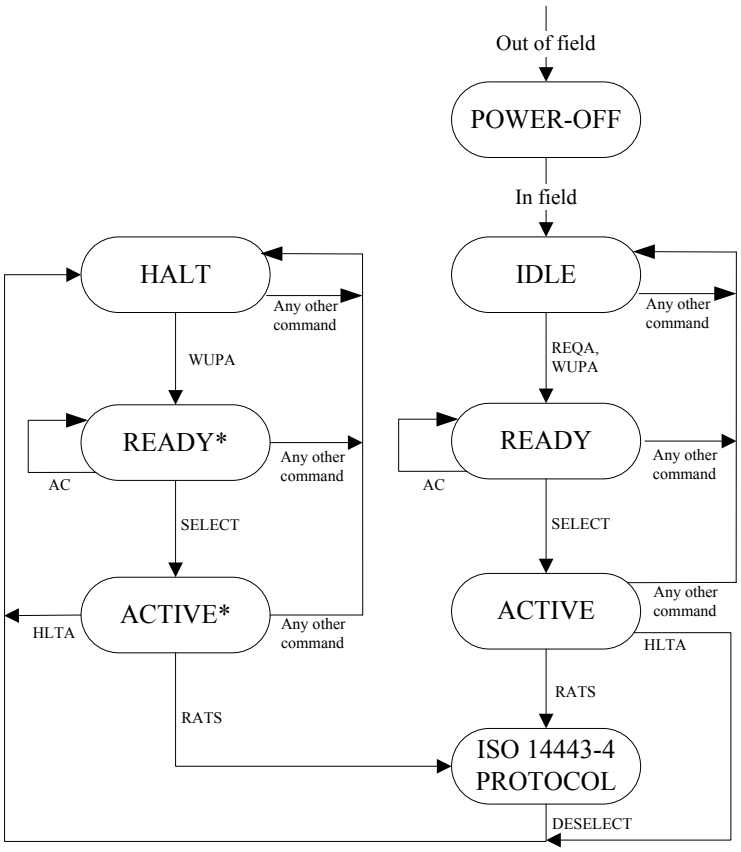


Figure 4.6: PICC Type A state diagram during activation and anticollision [26].

only during the first time the PICC is activated from the IDLE state. The second set is marked with an asterisk and is used every time the PICC is reactivated from the HALT state by a Wake-UP A (WUPA) command. Note that PICCs in HALT state do not respond to REQA commands.

4.4 Part 4: Transmission Protocol

The ISO 14443 standard defines a half-duplex block-transmission protocol, which considers special needs for contactless environments. Type A PICCs need to be activated before the transmission protocol can be started. This activation sequence is described in this part of the standard. PICCs of Type B do not need additional activation as all necessary information was already exchanged during anticollision [9].

4.4.1 Protocol Activation of Type A PICCs

The activation sequence builds on the anticollision described in Section 4.3.3. During anticollision the PICC sends a SAK command to the PCD. One bit in this command signals if the PICC is compliant with the ISO 14443-4 transmission protocol [26]. Note that a PICC might implement the anticollision according to the standard and support a different transmission protocol, such as it is in the case of MIFARE Classic [36]. The whole

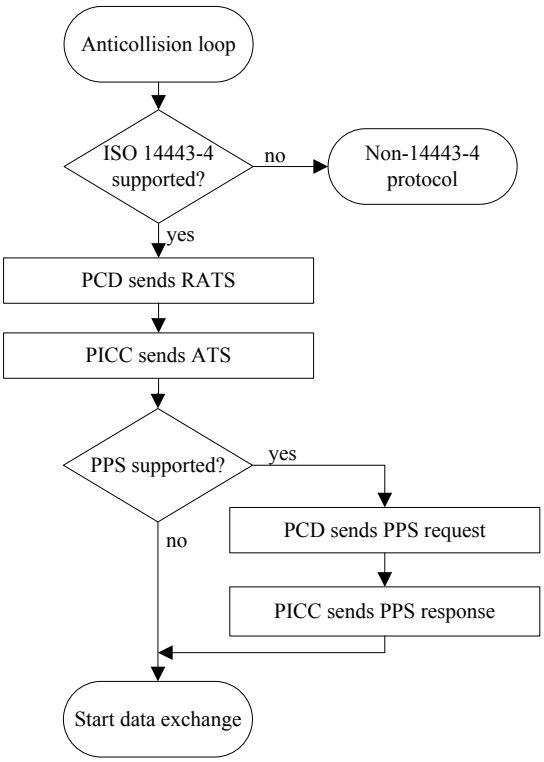


Figure 4.7: Activation sequence of a Type A PICC after anticollision [9][26].

flowchart for the protocol activation is shown in Figure 4.7.

If the PICC is compliant with ISO 14443-4, the PCD sends a Request Answer To Select (RATS) command. The RATS command contains two important parameters, namely the Frame Size Device Integer (FSDI) and the Card IDentifier (CID). The FSDI defines the maximum block-size (FSD) that the PICC can send to the PCD, where allowed values

Table 4.2: Fields contained in an ATS (Answer To Select) command [9][26].

Field	**Description**
FSCI Frame Size Card Integer	Maximum block-size (FSC) from PCD to the PICC.
DS Divisor Send	Possible data rates from the PICC to the PCD.
DR Divisor Receive	Possible data rates from the PCD to the PICC.
FWI Frame Waiting Integer	Integer that defines the Frame Waiting Time (FWT) the PCD must wait for a response of the PICC.
SFGI Start-up Frame Guard Integer	Integer that defines the FWT for the first command after the ATS.
CID Card IDentifier	Defines whether a CID is supported by the PICC.
NAD Node ADdress	Defines whether a NAD is supported by the PICC.

range from 16 bytes to 256 bytes. The CID is allocated by the PCD and allows to hold multiple Type A cards active at the same time and address them by their CID [9].

The following Answer To Select (ATS) is sent by the PICC and contains several important protocol parameters listed in Table 4.2. The exact coding for those parameters is described in Section 5.2 of ISO 14443 part 4 [27].

If changeable parameters are supported by the PICC, the PCD may send a Protocol and Parameter Selection (PPS) request. This request may set the parameters for Divisor Send (DS) or Divisor Receive (DR) and is acknowledged by a PPS response of the PICC. Afterwards, the application-data exchange can begin. Note that the PPS command does not need to be supported by ISO 14443 compliant PICCs.

Frame Waiting Time

As the Frame Waiting Time (FWT) is an important parameter for a relay attack, we have a closer look at it. As mentioned before, the FWT is not exchanged directly between PCD and PICC but rather the Frame Waiting Integer (FWI) is exchanged. The FWT is calculated using the FWI by

$$FWT = (256 \times 16/f_c) \times 2^{FWI}. \tag{4.1}$$

The FWI is defined in the range of 0 to 14. This results in 15 different FWTs shown in Table 4.3.

Table 4.3: Possible frame waiting times according to ISO 14443.

FWI	FWT	FWI	FWT
0	FWT_{MIN} = 302 µs	8	77.33 ms
1	604 µs	9	154.66 ms
2	1 208 µs	10	309.31 ms
3	2 517 µs	11	618.63 ms
4	4 833 µs	12	1 237.26 ms
5	9 666 µs	13	2 474.51 ms
6	19.33 ms	14	FWT_{MAX} = 4 949.03 ms
7	38.66 ms		

4.4.2 Half-Duplex Block-Transmission Protocol

The transmission protocol is very similar to the T=1 protocol [22], which is used for smart cards with contact. This makes the integration of the protocol in existing smart-card systems very easy [9].

Block Format

The structure of a block can be seen in Figure 4.8. It first consists of a prologue field, which can be seen as a header. It also holds an information field for application data and an epilogue field for error detection. In the following, we describe those fields and their purpose in detail.

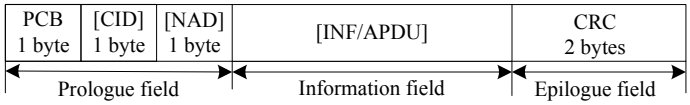


Figure 4.8: The structure of an ISO 14443-4 block [27].

Prologue Field. The prologue field contains the mandatory Protocol Control Byte (PCB) and the two optional bytes for the Card IDentifier (CID) and the Node ADdress (NAD). As mentioned before, the CID field is used to address one of multiple active PICCs. It also contains two bits that can be used as power-level indicator. The NAD field is used to build up and address multiple logical connections with a single PICC. The PCB byte also defines which type of block the frame represents. There exist three different types of blocks, which are presented in Table 4.4. For an exact definition of each block we refer to part 4 of the ISO 14443 standard [27].

Table 4.4: Different block types for the ISO 14443-4 protocol [27].

Block	**Description**
I-block:	Used to send information for the application layer
R-block:	Used for positive or negative acknowledgements
S-block:	Used to exchange control information between PCD and PICC

Information Field. The information field is optional and of variable length. It contains application data for I-blocks, and non-application data and status information for S-blocks. The length is calculated as the total number of bytes minus the prologue and the epilogue field.

Epilogue Field. The epilogue field contains a two-byte error-detection code (CRC_A) for the transmitted block. It is used to detect errors that occurred during the transmission. The exact calculation of the CRC_A is defined in the appendix of the ISO 14443-3 specification [26].

Protocol Operations

In the following, we describe some operations that are supported by the ISO 14443-4 protocol and that are of interest for relay attacks.

Waiting Time Extensions. Waiting Time Extensions (WTXs) are a way for the PICC to request additional time for complex and time consuming computations. They are sent as an S-Block with a one-byte Information (INF) field. The INF field contains two bits as power level indicator and six bits for the Waiting Time Extension Multiplier (WTXM). The standard defines the WTXM in the range of 1 to 59. Given the WTXM, the PCD grants a temporary Frame Waiting Time (FWT) computed by

$$FWT_{tmp} = FWT \times WTXM [27]. \tag{4.2}$$

However, the temporary FWT can never exceed the maximum FWT of 4.95 seconds. Note that it is possible to request several consecutive WTXs. Every WTX request, if received correctly, is answered with a WTX response by the PCD.

Multi-Activation. Multi-activation is a feature that allows the PCD to keep several PICCs in their ACTIVE-state simultaneously. Therefore, the PCD can access different PICCs without needing additional time for activation and deactivation. In order to distinguish between several active PICCs, the PCD assigns different CIDs, which it uses to address a single PICC.

Chaining. As the block-size is limited by FSC and FSD, the protocol defines a chaining method to split larger information into several blocks. Therefore, a chaining bit exists in the PCB of an I-block to notify the receiver that a chained message is following. The receiver acknowledges every received I-block during chaining with a corresponding acknowledge command. Note that the receiver of a chained block may be the PCD or the PICC.

Negative Acknowledges

The standard also defines a recovery procedure using Negative Acknowledges (NAKs). Whenever the PCD receives an erroneous response from the PICC or does not receive a response in time, it can request a retransmission by sending a NAK command. Note that this recovery mechanism is optional and the number of retransmission tries depends on the implementation of the PCD.

Chapter 5

Security Analysis of ISO 14443 Regarding Relay Attacks

In this chapter, we look at different methods that can be used to increase the chance for a successful relay attack. Those methods work on different layers of the ISO 14443 protocol and are applicable to all ISO 14443 compliant systems. First we show how modifications on the physical layer can be utilized by the attacker. We also show how to exploit the protocol layer for a relay attack.

5.1 Modifications on the Physical Layer

In this section, we look at the physical layer of the ISO 14443 protocol and how it can be exploited by an attacker. Therefore, we show how to modify the carrier frequency and the data rate in order to increase the chance for a successful relay attack.

5.1.1 Carrier-Frequency Modification

The carrier frequency (f_c) of an ISO 14443 system is defined as 13.56 MHz ± 7 kHz. The carrier frequency influences the speed of communication and the clock at which the PICC operates. Both of those factors contribute significantly to the delay during an RFID communication. Therefore, we suggest to modify the carrier frequency during a relay attack to gain additional time for the relay.

Although the carrier frequency is defined in the ISO 14443 standard, some PICCs might support higher frequencies. Most PICCs have a bandpass filter, however this filter might still allow slight variations of the frequency [13].

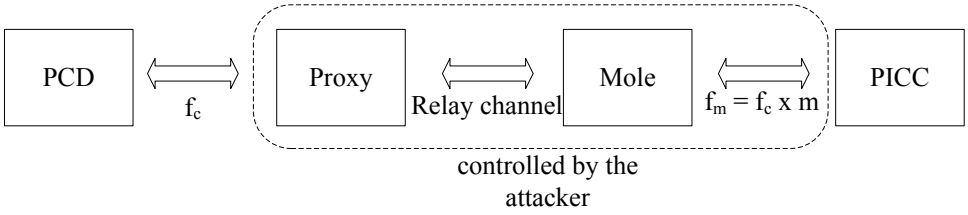


Figure 5.1: Setup of a relay attack where mole and PICC communicate at a higher frequency.

Figure 5.1 shows a possible setup for a carrier-frequency modification. The PICC is communicating with the mole on a higher frequency ($f_m = f_c \times m$, with the modifier $m > 1$). Therefore, the attacker has a speed advantage because the PCD uses a lower carrier frequency f_c to communicate with the proxy. Using this setup, the PICC operates at a higher clock frequency because it extracts its clock from the carrier. This leads to faster response times, which could be exploited by an attacker.

The speed advantage of this attack depends on the used modifier m. The RFID communication time between mole and PICC (in both directions) as well as the computation time of the PICC are both reduced by the same factor. The additional time for an attacker ($t_{additional}$) also depends on the time for a regular communication ($t_{regular}$), which is a function of f_c, and can be calculated by

$$t_{additional} = t_{regular}(f_c) \times (1 - \frac{1}{m}). \tag{5.1}$$

For f_c being 13.56 MHz, the additional times can be seen in Table 5.1. The table shows the values for different regular communication times $t_{regular}$ and different modifiers m. We see that $t_{additional}$ becomes quite significant for higher values of $t_{regular}$ and m.

Table 5.1: Additional relay times for a modified 13.56 MHz carrier.

$\mathbf{t_{regular}}$ [ms]	$\mathbf{t_{additional}}$[ms] m=1.1 f_m=14.92 MHz	m=1.25 f_m=16.95 MHz	m=1.5 f_m=20.34 MHz	m=1.75 f_m=23.73 MHz	m=2 f_m=27.12 MHz
1	0.09	0.20	0.33	0.43	0.50
2	0.18	0.40	0.67	0.68	1.00
5	0.45	1.00	1.67	2.14	2.50
10	0.91	2.00	3.33	4.29	5.00
50	4.55	10.00	16.67	21.43	25.00
100	9.09	20.00	33.33	42.86	50.00
1 000	90.91	200.00	333.33	428.57	500.00
2 500	227.27	500.00	833.33	1 071.43	1 250.00
4 950	450.00	990.00	1 650.00	2 121.43	2 475.00

5.1.2 Data-Rate Modification

The data rate is crucial for the delay during RFID communication. During a relay attack, there exist four different data rates: two (send and receive) between PCD and proxy and two between mole and PICC. An attacker can choose those data rates in a certain way to increase the chance for a successful attack.

The ISO standard defines four data rates, which depend on the carrier frequency. For a 13.56 MHz carrier, those data rates start at 106 kbit/s and go up to 848 kbit/s. This difference can be exploited by adversaries to obtain more time for relaying the data.

After the SELECT command, the PCD requests an Answer To Select (ATS) from the PICC. This ATS holds the Divisor Send (DS) and the Divisor Receive (DR), which define the supported data rates of the PICC. Therefore, the PICC can deliberately choose its own data rates.

Figure 5.2 shows a possible relay scenario. We assume the PICC supports a high data rate of 848 kbit/s (= $f_c/16$), which we use on one side of the attack. On the other side, the

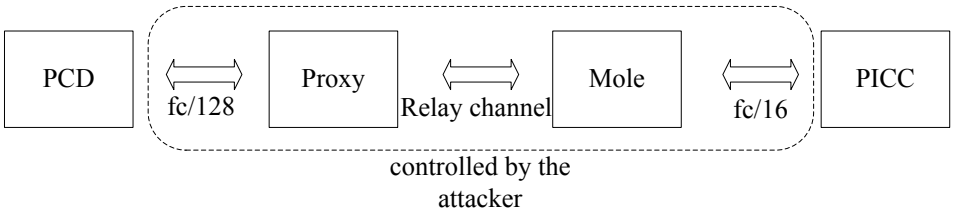


Figure 5.2: Setup for a relay attack where mole and PICC use a higher data rate.

attacker signals the PCD in the ATS command that the proxy only supports a low data rate of 106 kbit/s (= $f_c/128$). This leads to a faster transmission between PICC and mole than between PCD and proxy. Therefore, the mole receives responses from the PICC faster than the proxy has to forward responses to the PCD.

5.2 Modifications on the Protocol Layer

The half-duplex protocol defined in the ISO 14443-4 standard has several mechanisms that can be exploited for a relay attack. A description of those mechanisms and how they can be used is presented in the following sections.

5.2.1 Chaining

The maximum block-sizes for transmissions between PCD and PICC are agreed on during protocol activation. The PCD defines its maximum receiving block size in the RATS command (by the FSDI) and the PICC defines its in the ATS command (by the FSCI). During a relay scenario, we have two ISO connections and therefore four different maximum block-sizes. The attacker can choose two of them, the one from the PCD to the proxy, and the other from the PICC to the mole. The attacker might be interested in choosing small block-sizes at the proxy, so it can already forward a part of the whole message to the mole earlier.

Figure 5.3 and Figure 5.4 show the advantage of this approach on an example. In this example, the PCD wants to transmit a 160-byte data packet to the PICC. In Figure 5.3, the attacker uses a large maximum block-size (256 bytes) at the proxy and forwards the complete block at once. Note that the crucial Frame Waiting Time (FWT) starts after the last block is sent by the PCD. Therefore, in this scenario the attacker has to forward 160 bytes over the relay channel within the crucial FWT. In Figure 5.4, the attacker uses a smaller maximum block-size (16 bytes) at the proxy. Therefore, the attacker only has to

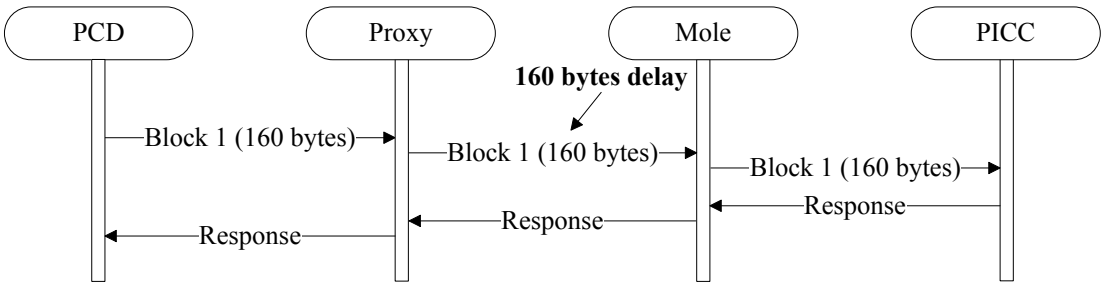


Figure 5.3: Example for a relay attack with a large maximum block-size (256 bytes).

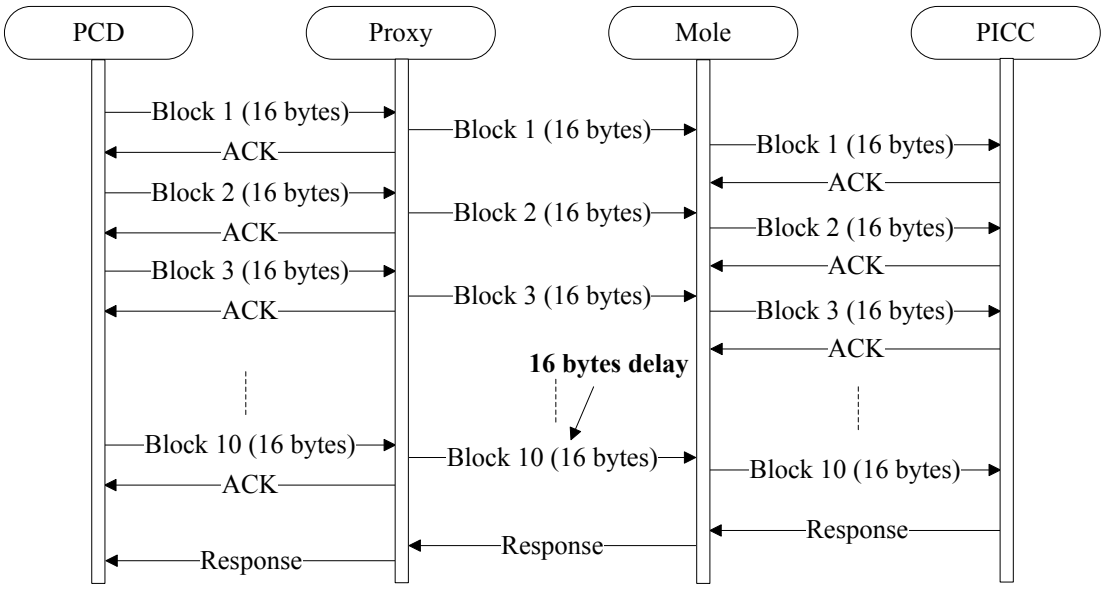


Figure 5.4: Example for a relay attack with a small maximum block-size (16 bytes).

forward 16 bytes after the last block is received from the PCD. As a result, the data that needs to be relayed within the crucial FWT is only a tenth of the original data.

Different block sizes can be used for the other relay direction as well. The attacker splits the response into smaller data packages before sending it from the mole to the proxy, as smaller blocks are transmitted faster and more reliable. The proxy then splits the data into even smaller blocks, as there is no minimum block-size defined in the ISO 14443 standard. Each small block would then trigger an acknowledge by the PCD and

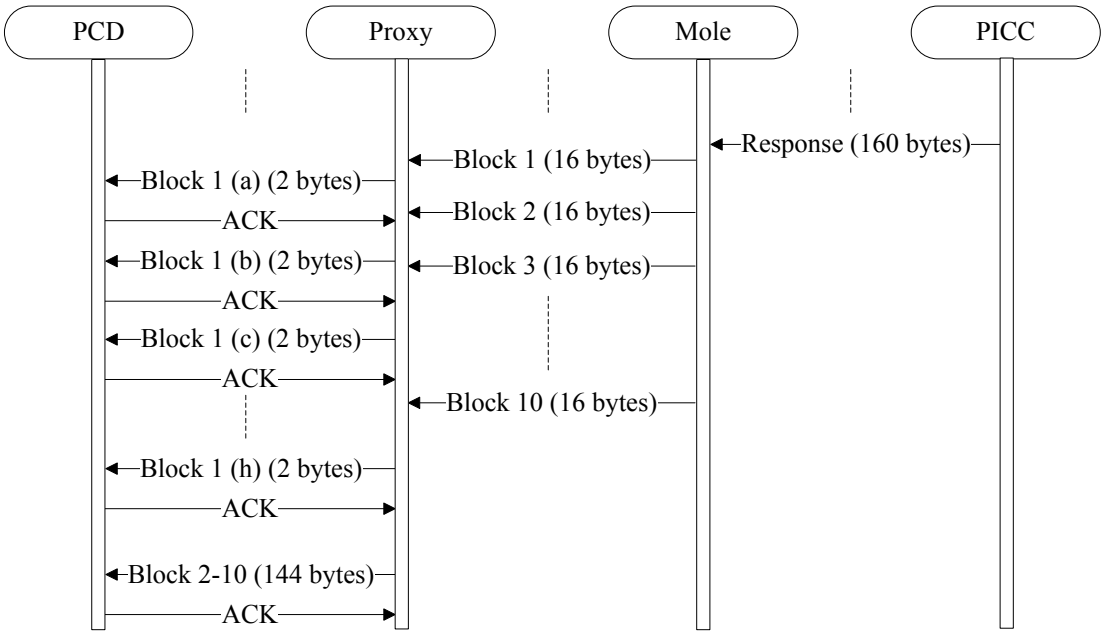


Figure 5.5: Example for splitting a response block into smaller blocks to gain additional time during a relay attack.

give the attacker an additional FWT before the proxy has to send the next small block.

Figure 5.5 shows an example for such an attack scenario. The PICC sends a 160-byte response to the mole. The mole splits it into ten 16-byte blocks (1-10) and sends those block separately to the proxy. The proxy splits those blocks into eight two-byte blocks (a-h) and sends those one after the other to the PCD. Note that as soon as all the data from the mole is received, the proxy can use larger block sizes to send data to the PCD. Also note that an attacker could start sending the response from the mole to the proxy earlier, if the mole requests smaller blocks from the PICC.

5.2.2 Waiting Time Extension

The ISO 14443 standard defines a command to request a Waiting Time Extension (WTX) for the PICC (see Section 4.4.2). This WTX can be requested by the PICC at any time during a transaction to request additional time for computation. This mechanism can be exploited to give an attacker additional time for a relay attack.

Figure 5.6 shows the basic concept of this attack. The proxy sends WTX requests as long as no response was received from the mole. Therefore, stalling the PCD, which grants the WTXs according to ISO 14443. As every ISO 14443 compliant PCD has to support WTXs, this is a very powerful attack strategy. Note that WTXs are defined in part 4 of the ISO 14443 standard and are not applicable during activation. Therefore, FWTs during the activation and anticollision cannot be extended by WTXs.

The protocol also defines no maximum number of consecutive WTXs that can be sent. Therefore, the attacker could delay its own answer for an arbitrary time. Consequently, all ISO 14443-4 protocols with no additional security mechanisms are highly vulnerable to those kind of attacks.

5.2.3 Negative Acknowledge

The Negative Acknowledge (NAK) is another mechanism that is defined for the half-duplex protocol. It is sent by the PCD whenever the answer of the PICC is erroneous or not received at all. Although the standard does not demand a certain behavior for lost or corrupted packages, vendors often implement their PCDs with an error-recovery routine. This routine sends NAK messages to the PICC to prompt it to resend the last response.

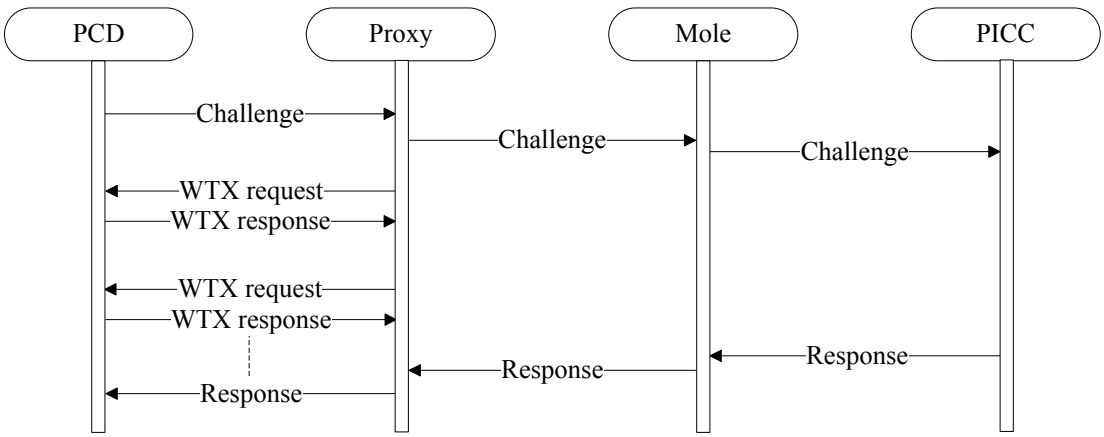


Figure 5.6: Using waiting time extensions to gain addition time for a relay attack.

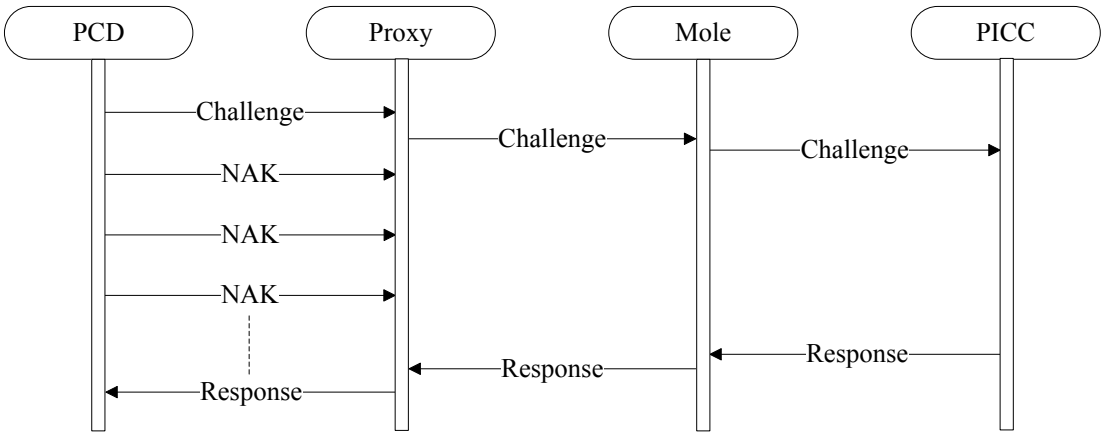


Figure 5.7: Using negative acknowledgments to gain additional time for a relay attack.

In case the proxy does not send an answer to the PCD before the FWT expires, the PCD assumes that the response was lost and sends a NAK to the PICC. After the PCD has sent the NAK message, the proxy again has one FWT to answer. A PCD could try to recover the communication using several NAKs, which would give the attacker more time to relay the data. The additional time for the attacker can be calculated as

$$t_{additional} = FWT \times NAK_{Sent\ by\ the\ PCD}\ . \tag{5.2}$$

An example of such an attack scenario is shown in Figure 5.7. As long as no response is received from the mole, the proxy simply ignores the PCD, which keeps sending NAKs to recover the connection. When the PCD receives the response from the proxy, it assumes that the response was lost before, and accepts it.

Chapter 6

ISO 14443 Compliant Countermeasures

There exist many proposals in literature on how to prevent relay attacks. Most of the proposed countermeasures are not compliant with the ISO 14443 standard. This RFID standard is widely used in the field of access control, ticketing, electronic payment, and electronic passports. In this chapter, we propose countermeasures to make relay attacks more difficult to perform. Note that we are aware that none of the presented countermeasures leads to full security. It should rather increase the effort for possible adversaries in order to make an attack more unattractive. One requirement of all proposed countermeasures is that they should be compliant to the ISO 14443 standard so that an integration into existing solutions can be made at minimal costs.

6.1 Preventing the Exploitation of Waiting Time Extensions

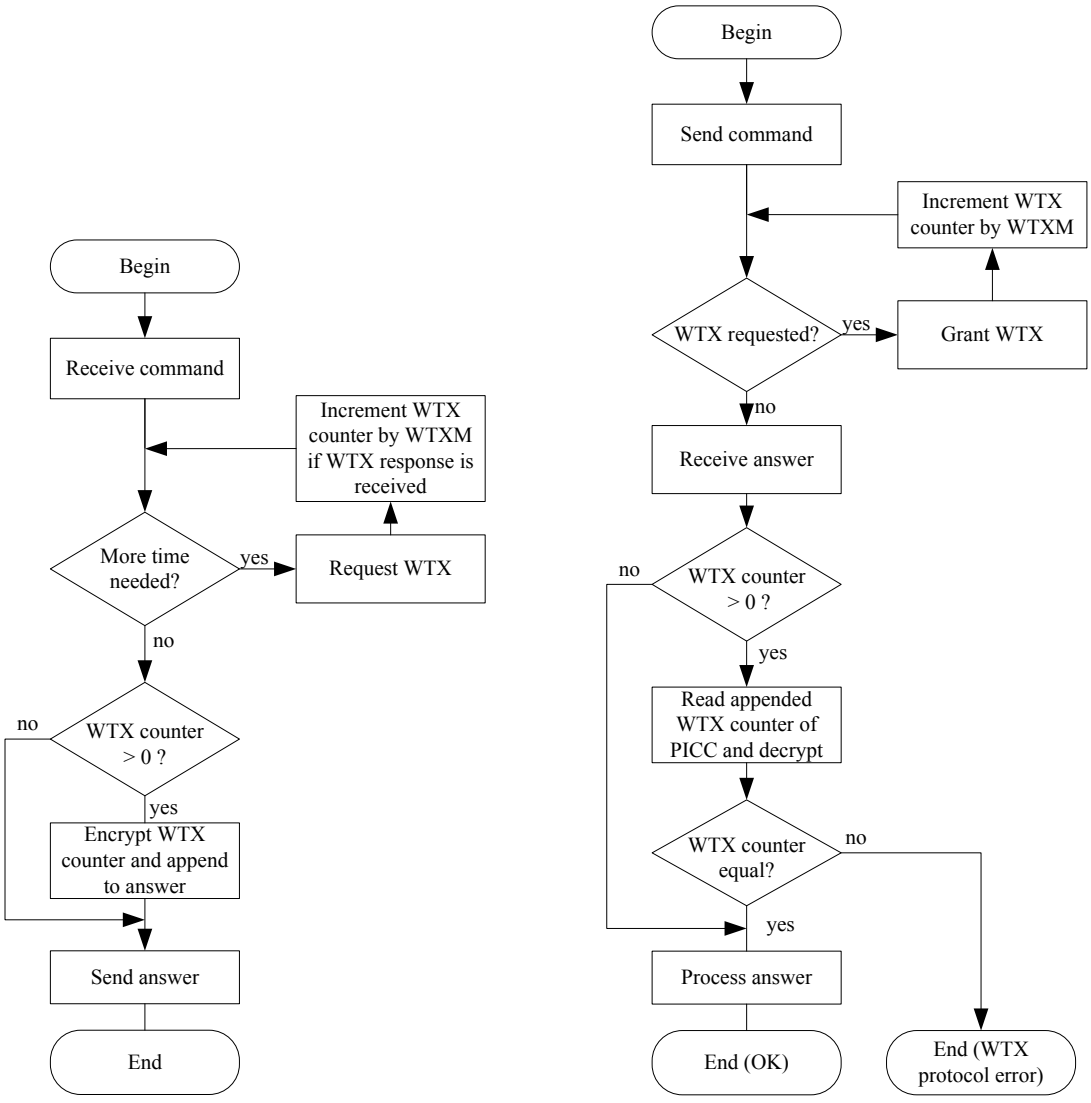
As described in Chapter 5, Waiting Time Extensions (WTXs) can be used to stall an ISO 14443-4 compliant PCD. An attacker can send WTX requests to the PCD, which has to grant a temporary FWT. This gives an attacker additonal time to relay the data. In the following, we propose a simple protocol to prevent the exploitation of WTXs for relay attacks, while still operating according to the ISO standard.

6.1.1 WTX-Exploitation Countermeasure

A WTX is always initiated by the PICC, or in case of an attack, by the proxy. The genuine PICC is not affected by a WTX sent by the proxy. Therefore, the PCD has knowledge of the WTX, whereas the PICC has no knowledge of it. This difference in knowledge can be utilized to prevent the use of WTXs by the attacker. Therefore, we propose the protocol shown in Figure 6.1a and Figure 6.1b for PICC and PCD, respectively.

WTXs are used to give the PICC more time to generate a response to a command sent by the PCD. Hence, every WTX is either followed by another WTX, or by the response to the command. A WTX command contains the Waiting Time Extension Multiplier (WTXM), which is used to calculate the temporary Frame Waiting Time (FWT) FWT_{tmp} by

$$FWT_{tmp} = FWT \times WTXM. \tag{6.1}$$



(a) State diagram PICC. (b) State diagram PCD.

Figure 6.1: Protocol to prevent the exploitation of WTXs for relay attacks.

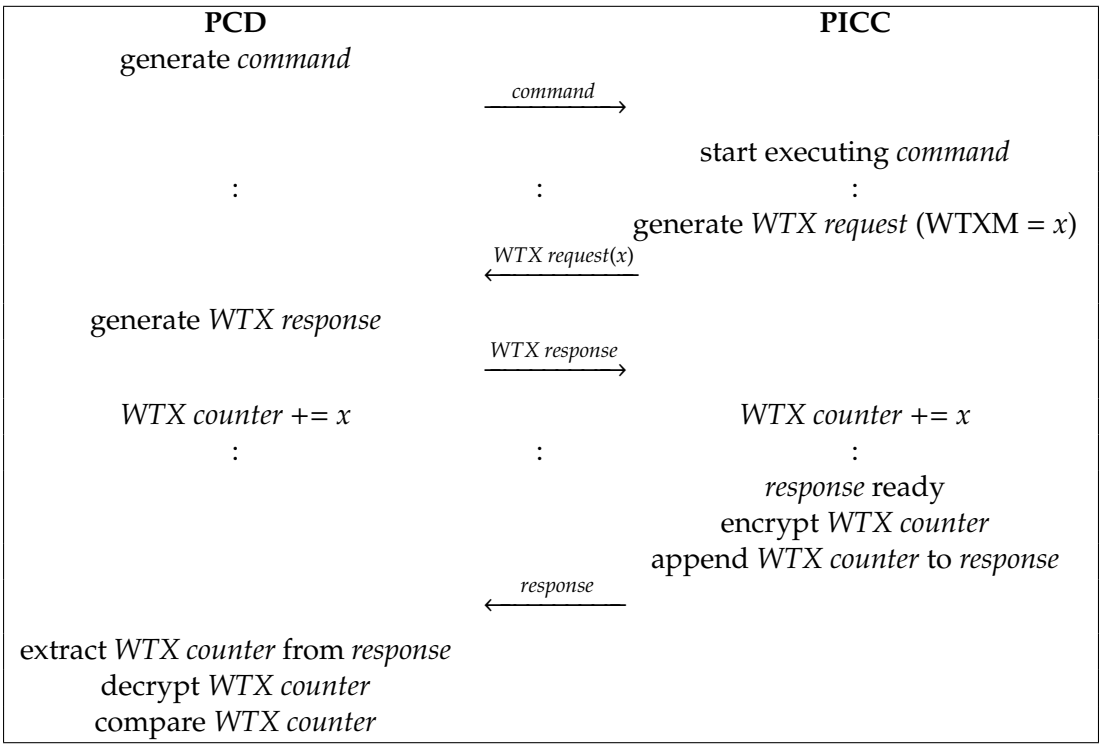


Figure 6.2: A message exchange between PCD and PICC requiring one WTX when using the WTX-exploitation countermeasure.

In the proposed protocol, the PICC adds up the WTXMs it sent, encrypts it, and appends this information to the response. The PCD then decrypts the information and compares its received sum of WTXMs with the sum of the PICC. If those two sums deviate, the PCD detects the relay attack and aborts the communication. Note that the protocol does only append the counter if at least one WTX was requested. Therefore, it does not produce any overhead for regular communication.

Figure 6.2 shows a message exchange between PCD and PICC when the protocol is activated. The example assumes that one WTX request is sent by the PICC.

We assume a secure encryption scheme (based on symmetric or asymmetric primitives), which is not vulnerable to replay attacks. This means that an attacker cannot capture the cipher for the WTX counter and reuse it during another attack. Most current systems already implement such encryption schemes (*e.g.* Advance Encryption Standard (AES), Data Encryption Standard (DES), Elliptic Curve Cryptography (ECC)), which makes the integration of the new protocol fairly easy.

However, the proposed protocol is still vulnerable if the attacker used different FWTs at the PCD and the PICC. If the FWT between PICC and mole was lower than between PCD and proxy, this would cause the PICC to send WTXs to the mole, before the waiting time of the PCD is up. Therefore, the system has to confirm that the FWT is the same for PCD and PICC. This can be done by requesting the encrypted or signed FWI of the PICC so that the PCD can ensure that both operate with the same FWT. More on checking the transmission parameters is discussed in Section 6.2.

	PCD		PICC	Comment
1.	$I(0)_0$	===>		
2.		<=//=	S(WTX) req.	
3.	$R(NAK)_0$	=//=>		
4.		<= =	-	time-out
5.	$R(NAK)_0$	===>		
6.		<===	S(WTX) req.	
7.	S(WTX) resp.	===>		
8.		<===	$I(0)_0$	

Figure 6.3: ISO 14443-4 Protocol Scenario 15: Request for a WTX [27].

	PCD		PICC	Comment
1.	$I(0)_0$	===>		
2.		<===	S(WTX) req.	
3.	S(WTX) resp.	=//=>		
4.		<= =	-	time-out
5.	$R(NAK)_0$	===>		
6.		<===	S(WTX) req.	
7.	S(WTX) resp.	===>		
8.		<===	$I(0)_0$	

Figure 6.4: ISO 14443-4 Protocol Scenario 16: Request for a WTX [27].

6.1.2 Known Issues

There is one case in which this protocol results in an error, even if no relay attack is performed. This case is explained in the following section.

The ISO 14443-4 standard describes a number of RFID-communication scenarios and how they should be handled by the protocol. We now have a close look at Scenario 15 and Scenario 16 in Figure 6.3 and Figure 6.4, respectively. Full arrows (===>) describe a successfully received message, broken arrows (=//=>) describe an erroneously received message, and open arrows (<= =) describe that no message was received within a given time. R, S, and I define the block types used.

Those two scenarios look exactly the same for the PICC, as it cannot determine if the erroneous message in Step 3 was a NAK or a WTX response. However, the PCD has sent one additional WTX response, and therefore incremented the WTX counter more often than the PICC. This causes the proposed protocol to fail. As this case is rather rare, we propose to retry the communication several times, until it succeeds. Note that the protocol still does not allow the exploitation of WTXs by the attacker.

6.2 Check Transmission Parameters

During the activation sequence of the protocol, a number of parameters are agreed on between PCD and PICC. Those parameters include, for example, the block size, the Frame Waiting Integer (FWI), and the data rate. Earlier in this chapter we showed that those parameters can be exploited for a relay attack.

In order to prevent the attacker from using different transmission parameters at both

ends of the attack, the PCD can request the PICC to send its parameters by using a secure encryption scheme. Therefore, the PCD can check if the PICC has agreed on the same transmission parameters as it did. An attacker would then not be able to use different parameters at both ends of the attack.

6.3 Distance Bounding on the Application Layer

Hancke and Kuhn [13] stated that the distance resolution for a communication channel with bandwidth B can be roughly calculated by

$$r = \frac{c}{B}. \tag{6.2}$$

For an RFID data bandwidth of 300 kHz, this results in a resolution of about one kilometer. Therefore, Hancke and Kuhn consider the standard RFID-communication link as insufficient for distance measurements. However, it is possible to detect relay attacks using distance bounding on the RFID channel. We try to stay compliant with the ISO 14443 standard, and therefore propose to implement distance bounding by using the ISO 14443-4 half-duplex block-transmission protocol.

6.3.1 Distance-Bounding Protocol

Distance bounding is based on giving the PICC a sufficient easy challenge, so the PCD is able to estimate the approximate response time. The PCD can then determine if the response was received within the expected time. We use the same cryptographic primitives as used by Hancke and Kuhn [13]. Note that any other distance-bounding protocol could be used as well.

Figure 6.5 shows the basic concept of the Hancke-Kuhn protocol. The PCD (verifier) and the PICC (prover) share a secret key (K). The prover generates a random bit stream (nonce, N) and sends it to the verifier. The prover and the verifier then use the secret key and the nonce to generate two bit-sequences, S_0 and S_1. They might use a keyed hash-function for instance. The verifier acknowledges the received challenge after it computed the two sequences.

After the generation of S_0 and S_1, the PCD generates n random challenges (c), which are either 0 or 1, and sends them individually to the PICC. The PICC then responds with the current bit of the corresponding sequence. For example, if the first challenge c is a 0, the PICC returns R_1^0 (the first bit of S_0). If the i-th challenge c is a 1, the PICC returns R_i^1 (the i-th bit of S_1) and so on. This process is repeated n times, with n being the size of the two sequences S_0 and S_1. Note that this challenge-response sequence is the time-critical part of the protocol. All the complex computations are done upfront so that the prover only has to do a simple look-up. If the response is not received in time, the protocol failed and a possible relay attack is detected.

6.3.2 Security Considerations

In this section, we analyze the security of the distance-bounding protocol regarding two aspects: the cryptographic aspect and the relay-attack aspect.

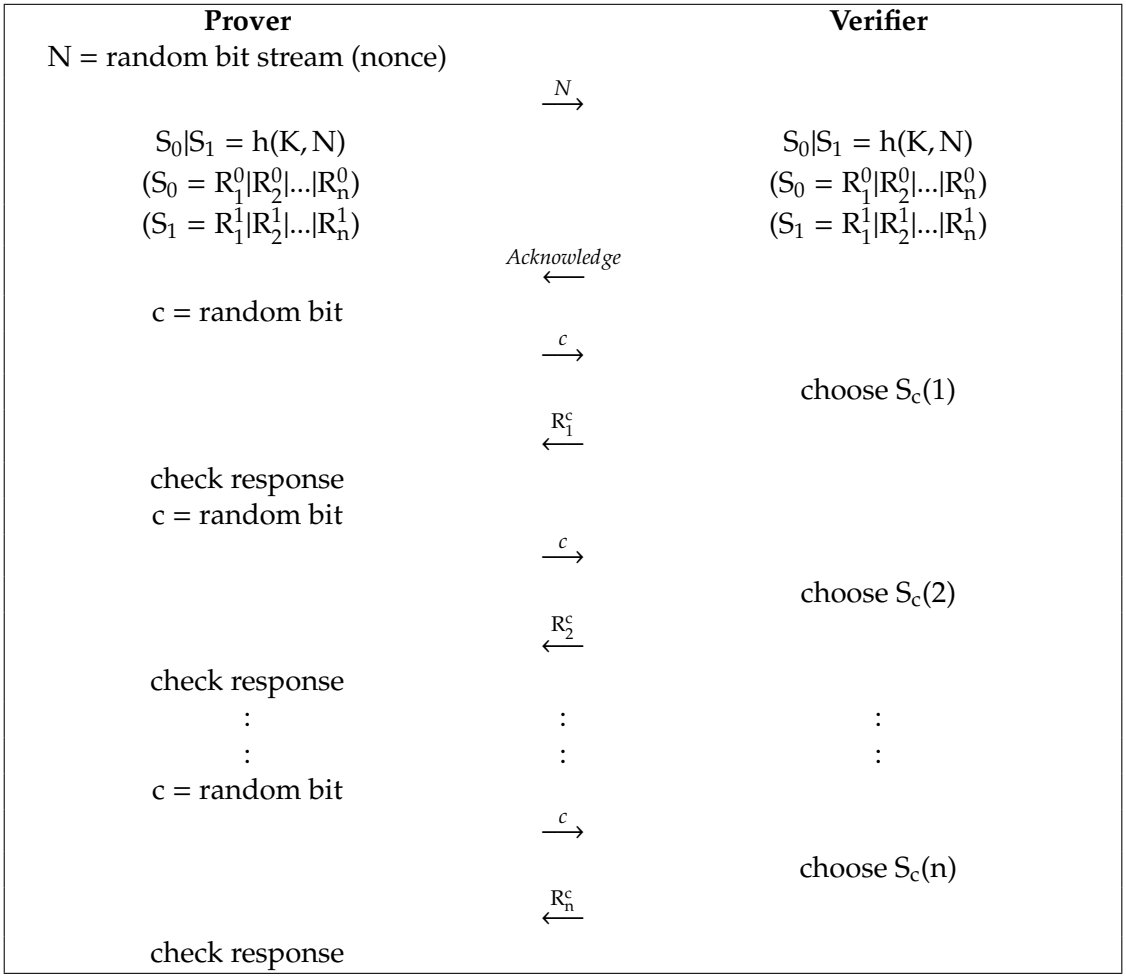


Figure 6.5: Communication between prover and verifier during the Hancke-Kuhn distance-bounding protocol.

Cryptographic Security

For our security considerations we assume that the secret key is in fact secret and the used hash function is secure. In this case, the cryptographic security of this protocol depends on the size of n. The probability P for an attacker to guess the right response can be calculated by

$$P = \frac{1}{2^n}. \tag{6.3}$$

Note that an attacker could request the responses from the PICC upfront himself and send it to the PCD. However, an attacker can never request all possible responses because the value that is not sent back for the requested index is discarded by the PICC. Therefore, the attacker has the same chance guessing the right response as requesting the right response from the PICC.

Relay-Attack Security

As mentioned before, this protocol used on the application layer is far from being an accurate distance measure. In order to reach good results, the system needs accurate timing at the PCD and a low variance in the response time of the PICC. This distance-bounding protocol is very easy and can be implemented on any ISO 14443 compliant device. Note that the minimum data size is one byte, and the protocol therefore has an additional overhead of seven bits.

As we are using the application layer of the ISO 14443 protocol, the PCD has to support WTXs and maybe NAKs as well. Therefore, an attacker could exploit those mechanisms to retrieve the correct responses from the PICC in time. However, the PCD can still measure the time elapsed before it received the response. Therefore, the PCD can decide if the response was received within a certain threshold.

We might consider allowing a certain number of delayed responses in order to support unreliable communication links. Otherwise, a PICC with an error-prone connection to the PCD may not pass the distance-bounding protocol. However, this could enable an attacker to retrieve the response from the PICC for those delayed responses. Therefore, we have to make sure that a sufficient number of challenges is answered within the defined threshold, so an attacker has only a small chance to guess the correct responses.

6.4 Summary

In this section we presented a number of ISO 14443 compliant countermeasures against relay attacks. We are aware that none of the proposed countermeasures leads to complete security against relay attacks. However, a combination of the proposed countermeasures would significantly increase the difficulty for an attacker.

In Chapter 5 we described a number of methods to increase the chance for a successful relay attack on ISO 14443 systems. By implementing all of the proposed countermeasures, we would be able to prevent all those attack scenarios but the carrier-frequency modification.

Chapter 7

The Hardware Setup

This chapter describes the hardware setup we used for our experiments. We first describe the attacking hardware, *i.e.* the mole and the proxy. Moreover, we list the remaining hardware used for our relay attack experiment, namely the PCD and the PICC.

7.1 The Proxy Device

The proxy impersonates the real PICC and therefore must be capable of operating as an RFID target. The main requirements for the proxy are speed, flexibility, and basic RFID and ISO 14443A functionality. Therefore, we used the IAIK HF DemoTag in version 3.0 [19] as our proxy device.

7.1.1 IAIK DemoTag

The heart of the DemoTag is a programmable ATxmega256A3 microcontroller from Atmel [1]. It further consists of a Printed Circuit Board (PCB) antenna, an analogue front end, and several interfaces like USB (used for monitoring and powering the device) or JTAG (used for programming the microcontroller). A library implements support for several RFID protocols such as ISO 15693, ISO 18000-3, ISO 18982, and ISO 14443A. It also allows changes in the layers two to four of the ISO 14443 protocol. Therefore, it can change the UID of the tag to any value (4-byte, 7-byte, and 10-byte UIDs are supported) and allows the exchange of Application Protocol Data Units (APDUs). Due to its flexibility, the DemoTag allows to perform passive as well as active relay attacks.

The USB interface allows to connect to the DemoTag through a hyperterminal on a virtual COM port. The hyperterminal can be used to request information from the DemoTag like the communication buffer or the current RFID protocol. The DemoTag also accepts commands through the COM port, which can be used to change the RFID protocol or perform a firmware reset.

The DemoTag is only about double the size of a regular ISO 7816 ID-1 card and runs completely independent with a portable power source (9 V battery). This makes the tag inconspicuous and very practical as a proxy. It also provides an additional interface for individual adapters. It allows other boards to be attached to it and to communicate with the microcontroller. We made use of this interface and attached a board with a Bluetooth module. This Bluetooth module can be accessed through an Universal Asynchronous Receiver Transmitter (UART) interface.

A picture of the DemoTag with the attached Bluetooth module can be seen in Figure 7.1a.

7.1.2 BTM-222 Bluetooth Module

We used the BTM-222 Bluetooth Module from Rayson [40] soldered on a Printed Circuit Board (PCB). It is a class 1 (+18 dBm) module and supports Bluetooth Ver. 2.0+EDR certification operating at speeds up to 3 Mbit/s. It also provides several interfaces such as an UART interface, which we use for communication with the DemoTag. The schematics and a picture of the Bluetooth board are included in Appendix A.

As soon as another Bluetooth device connects to the BTM-222, it forwards all the data received through Bluetooth to its interfaces and vice versa. Therefore, setting up a connection to the module is fairly easy, if the module is configured accordingly. The basic configuration steps are shown in the following section.

Configuration of the BTM-222

The BTM-222 has a Serial Port Profile (SPP) firmware that supports AT commands. The configuration is usually done through the UART interface with an RS-232 connection. However, our configuration is done by the DemoTag itself, which requires special handling. The module needs a guard time after every command character it receives in order to work properly. Therefore, the DemoTag sends every character separately, receives the echo, and waits for at least 40 milliseconds before sending the next character.

The default configuration of the UART interface is shown in Table 7.1. This is important because it is necessary to use the appropriate settings in order to start using the Bluetooth module.

Table 7.1: Default configuration of the BTM-222 Bluetooth module [41].

Baud rate	19 200 bit/s
Data bits	8
Parity	None
Stop bit	1
Flow control	Hardware or none

The complete set of AT commands and settings can be found in the datasheet of the BTM-222 [41]. We only present the most important commands and settings in Table 7.2. Each command starts with *AT*, followed by the command itself, and terminated by a *carriage return* (<CR >, \r in C, 10 in ASCII). We only changed a few settings, most importantly we increased the baud rate to 115 200 bit/s in order to reduce the communication delay between Bluetooth module and DemoTag.

By default, each command is answered with a result code, which can be *OK*, *CONNECT*, *DISCONNECT*, or *ERROR*. The BTM-222 sends those result codes to its interfaces and queues them until they are read. Therefore, the DemoTag must ensure that every byte sent back by the BTM-222 is read, so that no backlog is created for further commands. This also includes the last two characters of every response, *carriage return* (\r) and *new line* (\n).

Configuration of the BTM-222 is normally done when no Bluetooth connection is established. If a connection exists, the module forwards all characters it receives to the

Table 7.2: Important SPP AT commands for the configuration of the BTM-222 Bluetooth module [41].

Command	Description	
+++ (Escape Sequence)	When the device is in data mode, it can be force back into command mode while maintaining the connection to the remote device. The sequence characters should be sent with 1 000 ms guard time	
D (Set Remote Device Address)	The device can specify a unique remote device for security purposes	
	D=xxx	"xxx" is a string of 12 hexadecimal digits
	D0 (def.)	All remote devices are allowed
	D?	Inquire the current setting
E (Echo)	Specifies if the devices echoes received characters	
	E0	Characters received are not echoed back
	E1 (def.)	Characters received are echoed back
	E?	Inquire the current setting
L (Baud Rate Control)	Specifies the baud rate of the UART port	
	L2 (def.)	19 200 bit/s
	L4	57 600 bit/s
	L5	115 200 bit/s
	L?	Inquire the current setting
Q (Result Code Suppression)	Specifies if a result code is sent after receiving a command	
	Q0 (def.)	The device prompts result codes
	Q1	The device does not prompt result codes
	Q?	Inquire the current setting
Z (Restore)	Command to restore default settings	
	Z0	Restore default settings

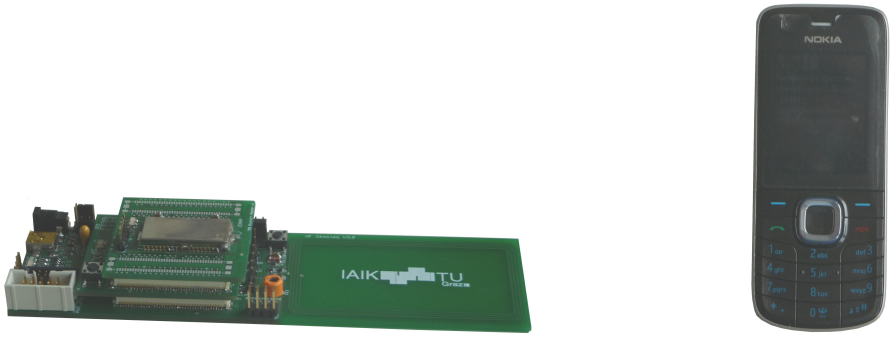
connected Bluetooth device. However, with an escape sequence shown in Table 7.2, it is possible to reconfigure the Bluetooth module while it is connected as well.

7.2 The Mole Device

We used a Nokia 6212 NFC [33] mobile phone as our mole device. A mobile phone is a very likely mole for a real life scenario because it is inconspicuous. The Nokia 6212 runs Java applications using the Java Platform, Micro Edition (Java ME). Java ME includes Application Programming Interfaces (APIs) for Bluetooth and Near-Field Communication (NFC), and therefore meets the requirements for our mole device. The phone can be seen in Figure 7.1b.

The Java applications running on the mobile phone are called MIDlets. Nokia provides a Software Development Kit (SDK) for many of its mobile phones, including the Nokia 6212. We used a designated version of Eclipse, called Eclipse Pulsar [8], which is specially designed for the implementation of MIDlets. Using the Nokia PC Suite [34], we were able to install our applications on the mobile phone.

The SDK also provides an emulator to test our applications without installing it on the mobile phone each time. However, this emulator has limitations and does not always



(a) Picture of the IAIK HF DemoTag v3.0 with the attached BTM-222 Bluetooth Module.

(b) Picture of a Nokia 6212 NFC mobile phone.

Figure 7.1: Attacking devices used for our relay attack.

behave the same way as the real phone, especially if the MIDlets use NFC and Bluetooth communication. Therefore, our MIDlets were tested on the mobile phone directly.

7.3 The PCD

The PCD could be any standard RFID reader that supports ISO 14443A. We used a Pegoda MF RD 700 RFID reader from NXP Semiconductors [37]. This contactless reader is designed for testing purposes and is easy to control through a USB interface. It supports all layers of ISO 14443A as well as the MIFARE-related protocols. The typical operating distance is defined with 75 mm.

We used a comprehensive C library provided by NXP, to implement a demonstration program that simulates a real life application. A picture of the PCD, with the proxy (DemoTag) in range, is shown in Figure 7.2a.

7.4 The PICC

In order to perform relay attacks, we could use any standard RFID tag that implements ISO 14443A. For our experiments, we used a MIFARE Plus S contactless smart card [36] from NXP. The MIFARE Plus S is used for public transportation, access management, electronic toll collection, car parking, and loyalty programs. It uses the open Advance Encryption Standard (AES) for authentication, integrity, and encryption and supports Unique IDentifiers (UIDs) of 4 bytes or 7 bytes. Moreover, it offers an Electrically Erasable Programmable Read-Only Memory (EEPROM) with a size of 2 kB or 4 kB and supports data rates up to 848 kbit/s. The card with the mole (mobile phone) in range can be seen in Figure 7.2b. Note that NXP also offers MIFARE Plus X smart cards [35], which are also compliant to the ISO 14443A standard and already implement a proximity check. Therefore, they are not suitable for our relay-attack experiments.

However, as we also want to implement and evaluate countermeasures against relay attacks, we need a more flexible PICC as well. Therefore, we used another HF DemoTag

(a) Picture of the PCD with the proxy in range.

(b) Picture of the mole with the PICC in range.

Figure 7.2: Setup of our relay-attack experiment.

v3.0 [19], developed by IAIK, as an RFID-tag emulator. Using the microcontroller on the DemoTag, we were able to analyze and test the countermeasures we proposed in Chapter 6.

Chapter 8

Implementation

In this chapter, we give implementation details of our relay-attack experiments. First, we describe the general concept of our implementation. Afterwards, we describe the implementation on the relay devices, *i.e.* the proxy and the mole. Finally, we give details about the implementation of countermeasures on the PCD and the PICC. For our experiments we used the hardware described in Chapter 7.

8.1 General Concept

In this section, we describe the basic process of our relay attack. A flow chart of the main steps at the different devices is presented in Figure 8.1.

We assume there exists a PCD that periodically queries for new PICCs in its reading range. This is a likely operation mode for many real life applications such as keyless-entry systems or payment systems.

The start of the attack is triggered at the proxy device, which acts as the commanding device (master). It might be in attack mode from the very beginning or might be triggered in any other way. In our implementation, it is triggered by a keyboard command send through a virtual COM port on the USB interface.

The first action the proxy performs is to request the Unique IDentifier (UID) of the PICC from the mole. This request is received by the mole which in turn starts querying for PICCs in range. If a PICC is detected, the mole activates it and returns the UID to the proxy. Note that the mole keeps the PICC active at all times to speed up the communication that happens later.

Once the proxy receives the UID, it sets its own UID to the one of the PICC. Now the proxy is ready to start communication with the PCD. The PCD is now able to find the proxy which in turn performs the ISO 14443A activation sequence with the PCD. Note that only the UID of the PICC is needed to perform a successful activation.

After the activation is done, the PCD sends application messages using ISO 14443-4 commands [27]. In fact, the PCD sends all application messages to the proxy, which immediately forwards them to the mole. The mole again forwards them to the PICC, receives the corresponding responses and sends those response back to the proxy. Therefore, the proxy has valid responses to the ISO 14443-4 messages, which it sends back to the PCD. The PCD does not realize that the messages are not coming from a valid PICC.

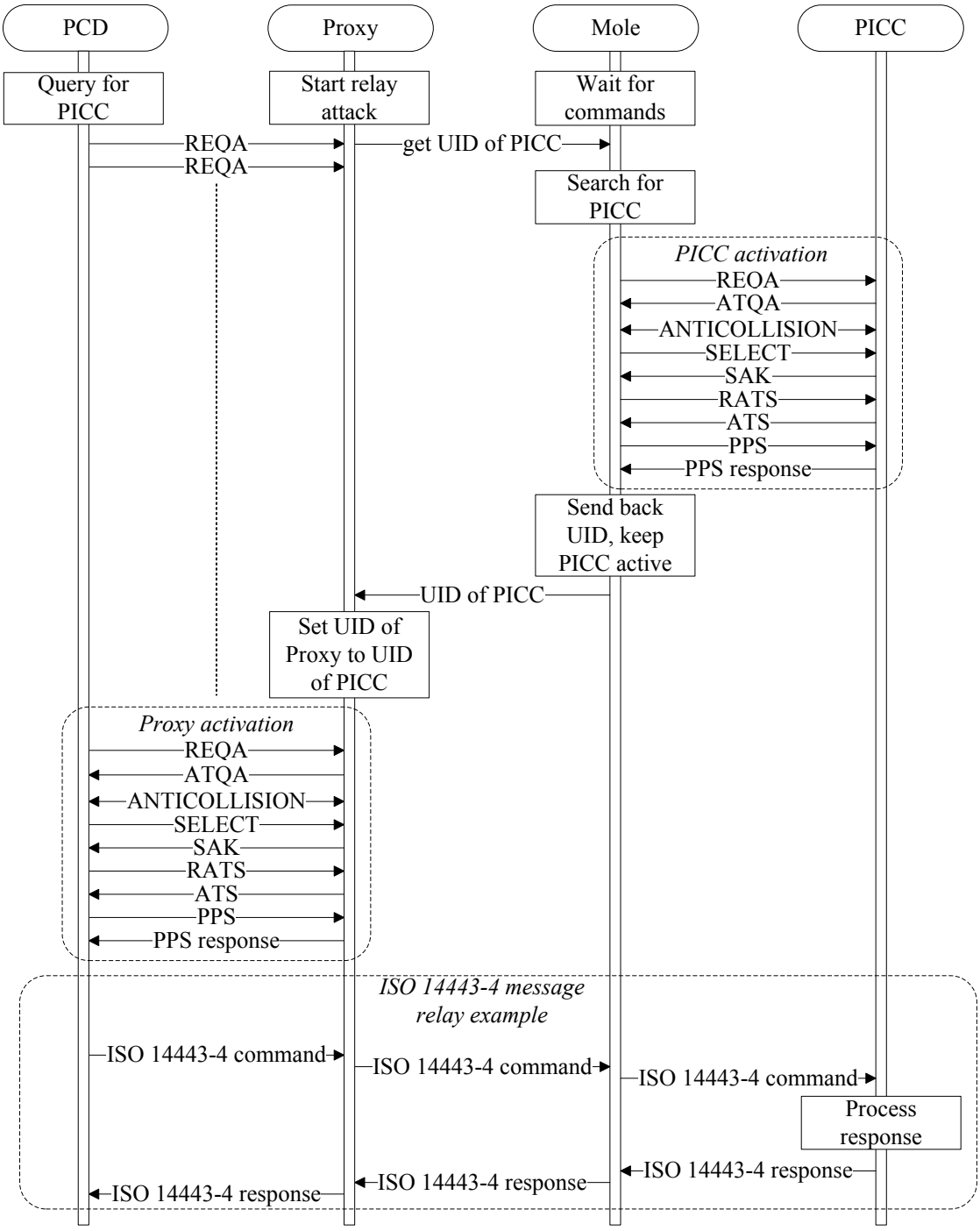


Figure 8.1: General concept of our relay-attack implementation.

8.2 Communication Between Proxy and Mole

The communication between proxy and mole is performed through a Bluetooth channel. Basically, we send single characters over the Bluetooth interface from one device to the other. In order to structure the communication, we defined a simple protocol to exchange commands and data between the two devices.

Every protocol exchange starts with a command of the proxy (the master). This command might need additional parameters, which are separated by a delimiter, in our case a semicolon. The end of a message is indicated by two semicolons. Following the command of the proxy, the mole response with an answer, which might be data, an error code, or a success code. Again, the response is terminated by two semicolons.

8.2.1 Protocol Commands

The protocol supports only a small number of commands but could be easily extended if necessary. Currently supported commands with their parameters are presented in the following paragraphs.

CHECK. Before starting a relay attack, the proxy ensures that a Bluetooth connection is established with the mole. In that case, the mole responds with *OK* and signals that it is ready. If no connection exists, the Bluetooth module simply echos the command and the proxy recognizes that no connection is established at the moment.

- Parameters: none
- Response: OK
- Example:

```
Proxy: CHECK;;
Mole:  OK;;
```

GETUID. After the proxy ensured that a Bluetooth connection is established, it tries to get the UID of a PICC. The mole responds with the UID of a PICC in range. Our implementation supports 4-byte, 7-byte, and 10-byte UIDs.

- Parameters: none
- Response: UID of a PICC
- Example:

```
Proxy: GETUID;;
Mole:  9D19C605;;
```

FWD. ISO 14443-4 commands sent from the PCD to the proxy are forwarded using this command. If the mole has lost connection to the PICC, it returns *ERROR*.

- Parameters: PCD message
- Response: PICC response or *ERROR*
- Example:

```
Proxy: FWD;0x60;;
Mole:  0x0B;;
```

END. Signals the mole that the relay attack on the current PICC is over. The mole confirms the message, deactivates the PICC, and waits for a new relay attack to start.

- Parameters: -
- Response: OK
- Example:

```
Proxy: END;;
Mole:  OK;;
```

8.3 Proxy Implementation

This section describes implementation details of the proxy on the DemoTag. We describe the most important features of the implementation on the DemoTag and show how specific situations are handled by our implementation.

8.3.1 General Proxy Implementation

By default, the proxy (DemoTag) is not triggered to perform a relay attack, although this would be possible. For testing purposes, it is initially configured as a regular ISO 14443A target, with no functionality on the application layer.

The DemoTag can be accessed by a hyperterminal through its USB interface (a virtual COM port). The user can send commands to the DemoTag, like switching to another protocol or reseting the firmware. The DemoTag in return sends information back, like the communication buffer or the current protocol.

Using this hyperterminal, we start the relay attack by sending the character *r*. This starts a routine that disables the target functionality and checks if the Bluetooth connection to the mole is established. If a connection exists, the DemoTag sets a global relay-attack flag. This ensures that every command received by the PCD is forwarded to the mole if the flag is set. It then sends a GETUID command to the mole and blocks until an answer is received.

Once the mole sends back the UID of the PICC, the DemoTag is again available as a target. Note that now the global relay-attack flag is set. It is then discovered by the PCD, which performs the anticollision and activation sequence. Note that the DemoTag uses the received UID for this procedure.

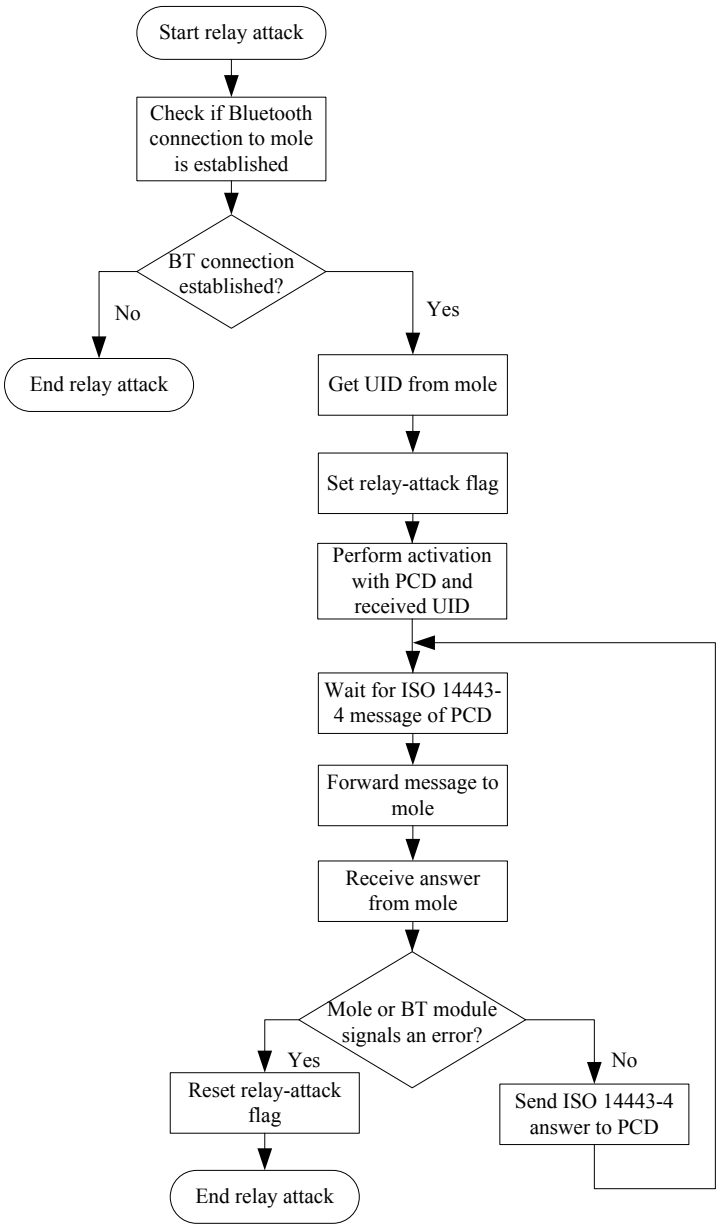


Figure 8.2: Flow chart of the relay-attack implementation on the proxy.

After the activation is finished, the PCD starts sending application-layer messages (ISO 14443-4). The proxy now sends those messages to the mole and waits for a response. If the response is an error, either by the mole or the Bluetooth module, the proxy ends the relay attack. If it receives an ISO 14443-4 response, it forwards it to the PCD and waits for the next message.

A flow chart of the described functionality is shown in Figure 8.2.

8.3.2 Handling of Chaining

Our relay-attack implementation supports chained messages in both directions. We handle chained messages by first receiving every block at the proxy and answering with

an acknowledge. We repeat this process until all blocks are received and combine them into one message. The complete message is then relayed to the mole. In case the response of the PICC has to be chained, we again receive the complete message at once from the mole. We then split the message and send it to the PCD using chaining.

8.3.3 Exploitation of Waiting Time Extensions

As described in Chapter 4, ISO 14443 defines a Waiting Time Extension (WTX). The theoretical exploitation of WTXs for a relay attack is explained in Chapter 5. In the following, we focus on the practical implementation of the proxy.

In order to exploit WTXs, we start a timer at the proxy when we receive a message. This timer expires before the default Frame Waiting Time (FWT) expires and the PCD would assume an error. If the proxy receives the relayed answer of the PICC before the timer expires, it sends the answer to the PCD and stops the timer. If the timer expires before an answer is received, an interrupt routine is executed.

Within this interrupt routine, the proxy sends a WTX request to the PCD and waits for an answer. The answer should be a WTX response, in which case the WTX request was received correctly and the timer starts again. In case the answer is a Negative Acknowledge (NAK), the WTX request was not received correctly by the receiver. In this case the proxy tries to retransmit the WTX request. The PCD could also send a different response to the WTX request, like a DESELECT for example. In this case an error occurred and the relay attack has not succeeded. A flowchart of this procedure is shown

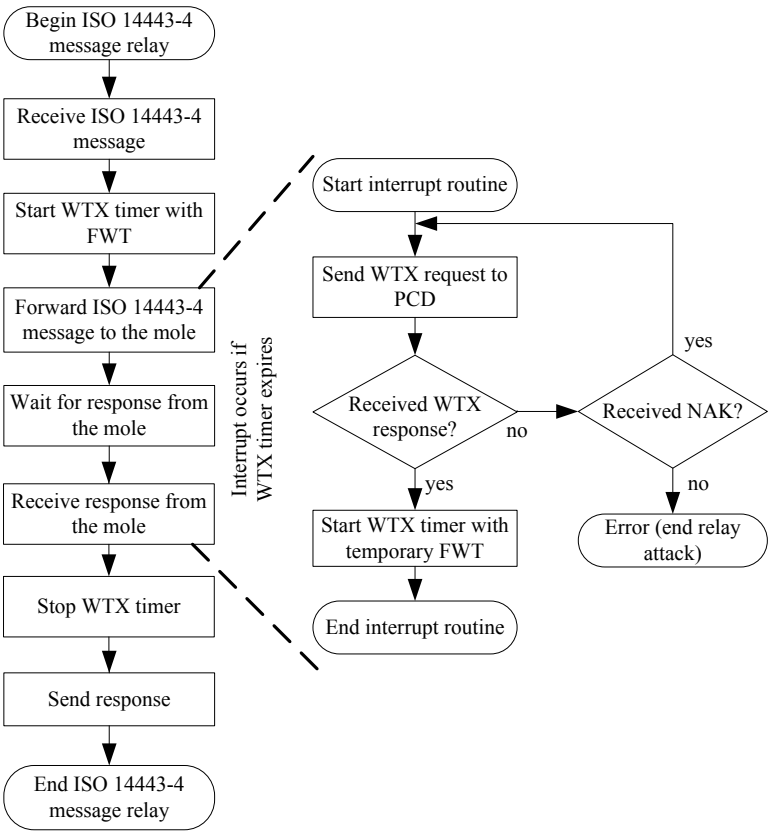


Figure 8.3: Implementation of the exploitation of WTXs on the proxy.

in Figure 8.3.

8.4 Mole Implementation

This section describes the implementation of the Java MIDlet we used for our relay attack. First, we describe the used Java packages. Second, we present the class diagram of our Java implementation. Finally, we show the basic concept of the implementation by giving a flow chart diagram.

8.4.1 Java Packages

For the Bluetooth connection with the proxy we used the *javax.bluetooth* package. It provides all necessary listeners for device and service detection as well as functionality for communication over the Bluetooth link.

For the NFC connection with the PICC we used the *javax.microedition.contactless* package. The available listeners support different target types, one of them being *ISO14443_CARD*, which we used for our experiment. Note that for one target type

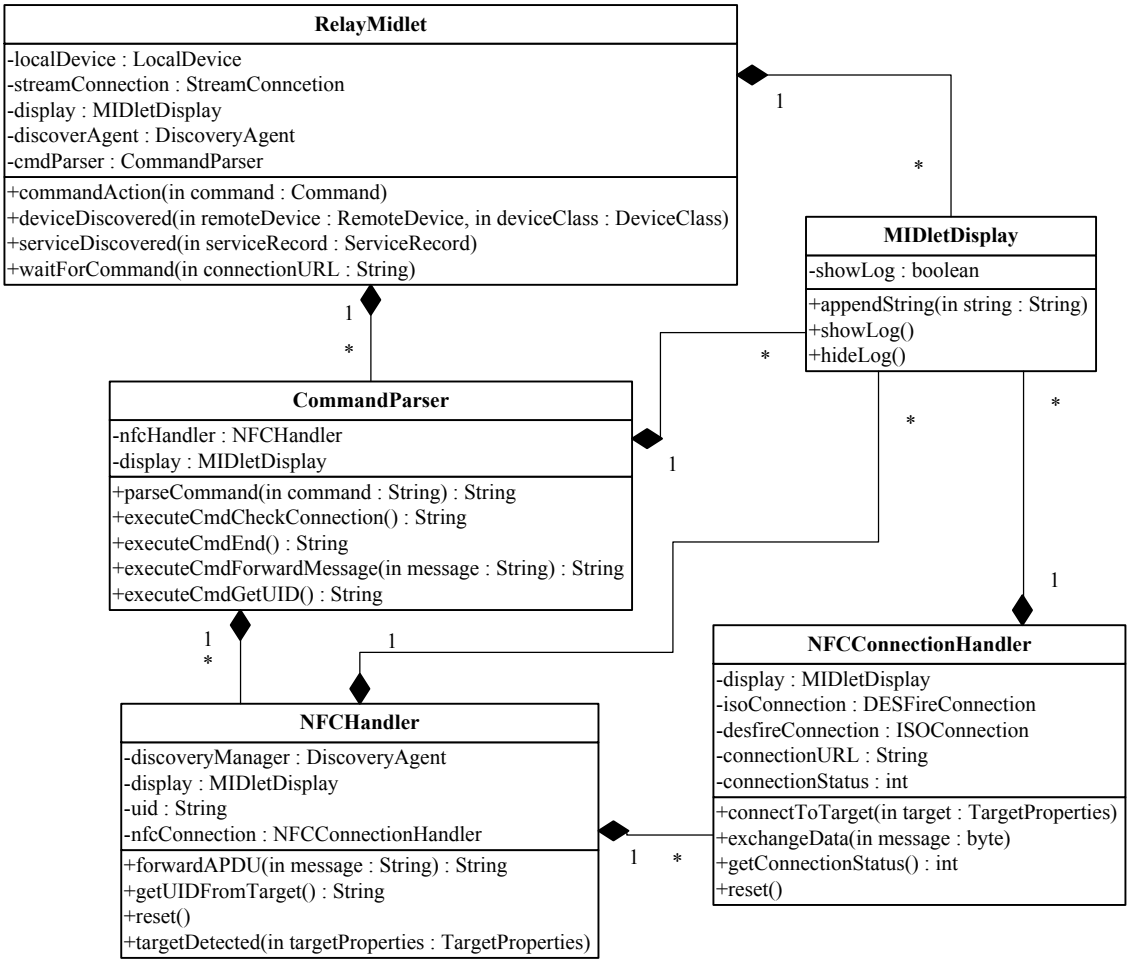


Figure 8.4: Exploitation of WTXs at the proxy.

there may exist several connection classes, which support PICCs from different series and vendors.

8.4.2 UML Class Diagram

Figure 8.4 shows the main parts of the UML diagram of our MIDlet implementation in Java. The central class of our implementation is *RelayMidlet*. It is responsible for handling the MIDlet functionality as well as the Bluetooth interface. Commands received through Bluetooth are forwarded to the *CommandParser*, which determines the command to execute. For the commands FWD and GETUID the *NFCHandler* is used to communicate with the PICC. The *NFCConnectionHandler* is needed because we support different NFC connection classes. Currently, those connection classes are *ISO14443Connection* and *DESFireConnection*. Note that both are ISO 14443 compliant. Finally, the *MIDletDisplay* class is used by all classes to print log messages.

8.4.3 Flow Chart

Figure 8.5 shows the basic flow chart of our Java MIDlet implementation. The chart shows the execution procedure as well as the responsible classes for each process.

Once the MIDlet starts the relay attack, it connects to the BTM-222 module on the DemoTag. After the connection is established, the MIDlet waits for a command from the proxy. If a command is received, the *CommandParser* determines which command should be executed and calls the corresponding method of the *NFCHandler*, if applicable. The *NFCHandler* executes the corresponding method using the *NFCConnectionHandler*. The resulting response is then sent back to the proxy by the *RelayMidlet*.

8.5 Countermeasure Implementation

The implementation of the countermeasures described in Chapter 6 is explained in this section. We first describe the implementation of the WTX-exploitation countermeasure and present distance bounding on layer 4 of the ISO 14443 standard. We used a HF DemoTag v3.0 [19], developed by IAIK, as a PICC and a Pegoda MF RD 700 [37], controlled by a C library (from NXP), as a PCD. Note that we did not implement any cryptographic primitives, as this is not the scope of this work. Instead, we performed XOR operations, using a shared secret key, to simulate encryption.

8.5.1 WTX-Exploitation Countermeasure

The WTX-exploitation countermeasure is a protocol designed to prevent the exploitation of WTXs by the attacker. In this section, we describe how the protocol was implemented on the PCD and the PICC.

We introduced a new variable, *wtxm_counter*, at the PCD, to sum up all Waiting Time Extension Multiplier (WTXM) values the PCD received from the PICC. Every time it sends a WTX response to the PICC, it increments the *wtxm_counter* accordingly. If one or more WTX requests are sent during a message exchange, the response of the PICC (to the original message) should contain the encrypted WTXM counter of the PICC. The PCD decrypts the counter and compares it with its own. If the two do not match, the protocol

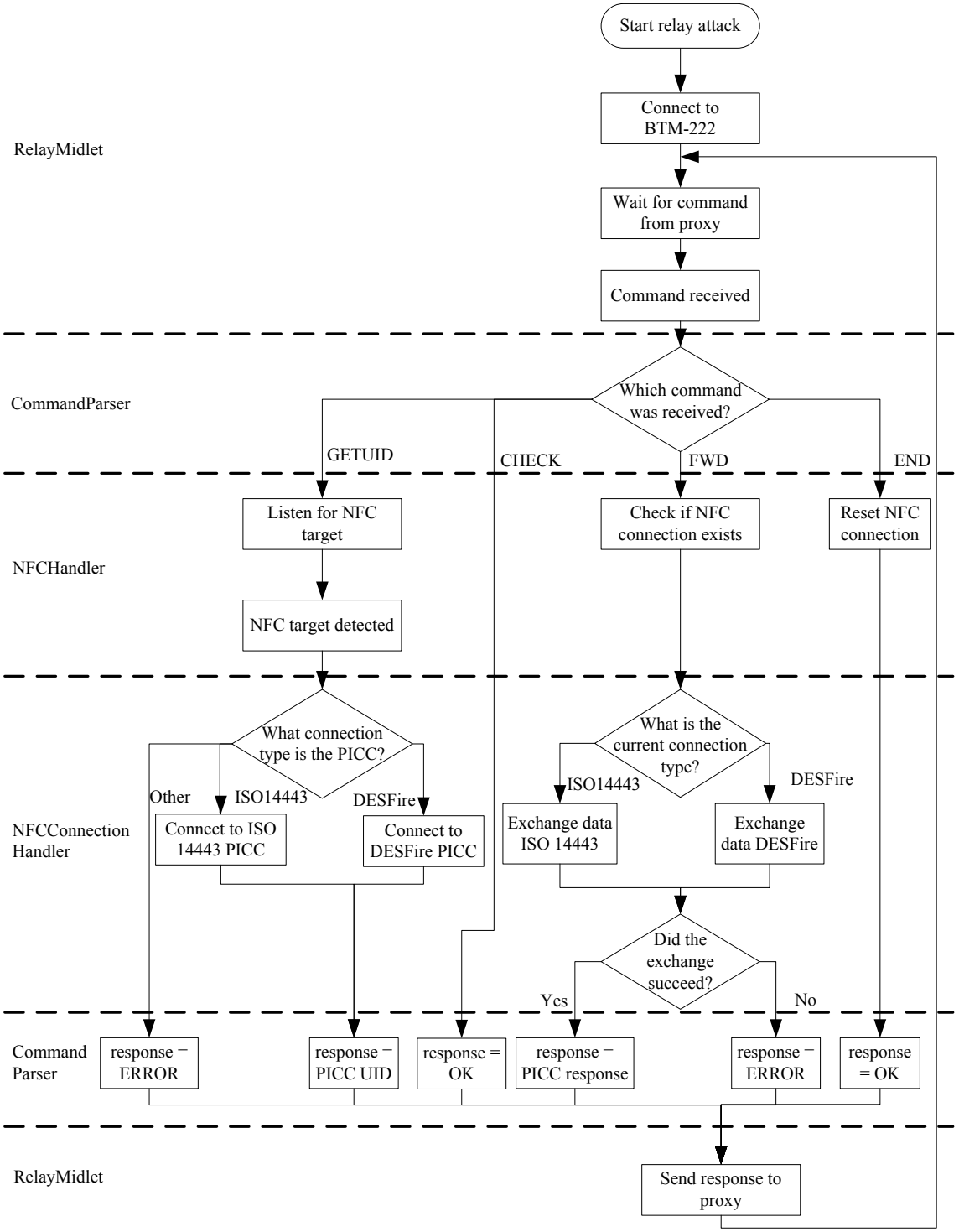


Figure 8.5: Flow chart of the Java MIDlet implementation.

failed, otherwise, it succeeded. Note that it is up to the PCD to check if the protocol succeeded.

In order to test the protocol, we triggered the exchange of WTX messages at the PICC by delaying the response using a waiting loop. We then implemented a timer on the PICC that triggers a WTX request before the default FWT expires. Every time the PICC sends a WTX request and receives a correct response, it increments its internal WTXM counter accordingly. If the WTXM counter was incremented, it is encrypted and appended to the response, otherwise, nothing is appended to the response.

By default, PCD and PICC do not use the WTX-exploitation countermeasure. The PCD must first activate the countermeasure, by sending a specified command to the PICC. We also exchange a random initial vector to ensure a secure encryption. The PICC acknowledges the received data and activates the countermeasure.

We used a 16-bit WTX counter, which allows the PICC to send 1110 WTX requests, with a maximum WTXM of 59, before an overflow occurs. This keeps the overhead to a minimum (two bytes), while still allowing a sufficient number of possible WTXs.

8.5.2 Distance Bounding

For our distance-bounding protocol, we defined a command byte on the application layer. The PCD sends this command byte, followed by a 12-byte random challenge, to the PICC. The PICC uses the 12-byte challenge, computes two six-byte sequences (*sequence0* and *sequence1*), and sends an acknowledge message to the PCD. The PCD generates the same two sequences before it starts generating random bits. Note that the minimum size of data is one byte, and we have an additional seven bit overhead for every challenge. The PICC is now expecting single-bit challenges and responds with the current bit of the corresponding series. This process is repeated 48 ($\frac{12}{2} \times 8$) times and should succeed for every challenge-response pair without transmission errors. The implementation could be less restrictive by allowing a certain number of NAKs.

In addition to receiving the correct response, the PCD also expects the PICC to respond within a certain threshold. This threshold should be chosen according to the application and the PICC response time. Also the implementation should allow a certain number of delayed responses for error-prone connections. We discuss error tolerance and reasonable thresholds in Chapter 9.

Chapter 9

Results

In this chapter, we present the results of our experiments. First, we present the results we gathered during our relay-attack experiment. Therefore, we look at the results of the protocol exploitations and the delays introduced by our relay attack. Second, we look at the performance of the countermeasures and how they could be implemented in a real system. Finally, we compare the obtained results with the literature and conclude with a summary.

9.1 Relay-Attack Results

Using our implementation, we were able to successfully attack the ISO 14443 compliant system described in Chapter 7. By fetching the UID of the PICC and by using it at the proxy, the PCD could not distinguish between the real PICC and the proxy device. In this section, we see that the delay introduced was well below the maximum Frame Waiting Time (FWT). The PCD accepted the relayed response of the PICC without any problems.

9.1.1 Protocol Exploitations

In this section, we analyze the behavior of the system regarding the exploitation of certain ISO 14443 mechanisms described in Chapter 5.

Waiting Time Extension

During our experiment, we successfully attacked the ISO 14443 test system. This relay attack succeeded without the exploitation of Waiting Time Extensions (WTXs). Therefore, we simulated a larger relay delay by introducing an artificial delay of 12 seconds at the mole (mobile phone) before sending back the response of the PICC. Therefore, the proxy was forced to utilize WTXs to stall the PCD until it received the response.

We performed our experiment with a Frame Waiting Integer (FWI) of 10 which results in a FWT of 309 milliseconds. By requesting a WTX with a Waiting Time Extension Multiplier (WTXM) of 59, the PCD granted an additional 4.95 seconds (FWT_{max}) for the response. Therefore, the proxy had to request three WTXs to stall the PCD for the artificial delay time of 12 seconds. As expected, the ISO 14443 compliant PCD accepted multiple WTX requests and granted the temporary FWTs. Therefore, the PCD was stalled until the mole forwarded the response of the PICC. Theoretically an attacker could stall the PCD for an arbitrary long time by sending WTX requests.

Note that even if the PCD does not implement a recovery mechanism in case of timeouts (*i.e.* NAKs), our implementation prevents those timeouts by sending a WTX request before the default FWT is over.

Negative Acknowledge

In order to test the exploitation of Negative Acknowledges (NAKs), we introduced an artificial delay of two seconds at the mole. Further, we configured the PCD to perform ten recovery tries before dropping the connection. The default FWT was again 309 milliseconds. The additional time for the relay $t_{additional}$ can be calculated by

$$t_{additional} = FWT \times NAK_{Sent\ by\ the\ PCD}\ . \tag{9.1}$$

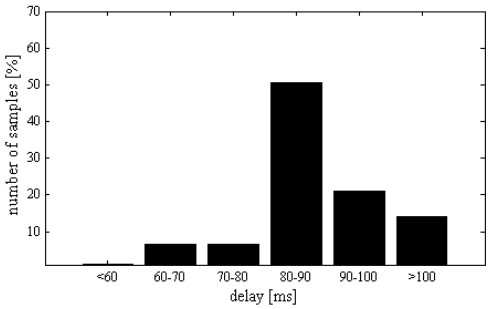
Therefore, the attacker had an additional 3.09 seconds to relay the data. The proxy simply ignored the PCD which kept sending NAKs until the configured maximum was reached, even though the proxy never sent a response. As the additionally introduced delay was only two seconds, the PCD accepted the response because it was still trying to recover the connection.

9.1.2 Delay Measurements

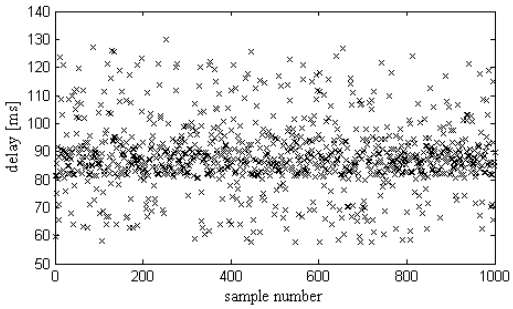
The theoretical delay introduced during a relay attack is described in Chapter 3. The total delay is defined as the sum of RFID-link delays, proxy delays, mole delays, relay-channel delays, and a computation delay at the PICC. Note that it is not always possible to perform measurements at the exact boundaries of those delays. Therefore, we focused on four different delays that give a good insight into the behavior of the system.

For every delay measurement, we gathered 1 000 samples to get solid and accurate results. We sent a single byte challenge (0x60) and received a single byte response (0x0b) for each measurement. However, the method used for the measurement differed depending on the measured delay. Finally, we took 1 000 samples of a regular RFID communication with a MIFARE Plus S card to compare the resulting delays with our relay-attack measurements.

In this section, we first analyze the distributions of the measured delays separately. We conclude this section by evaluating the delays and by comparing their statistics.

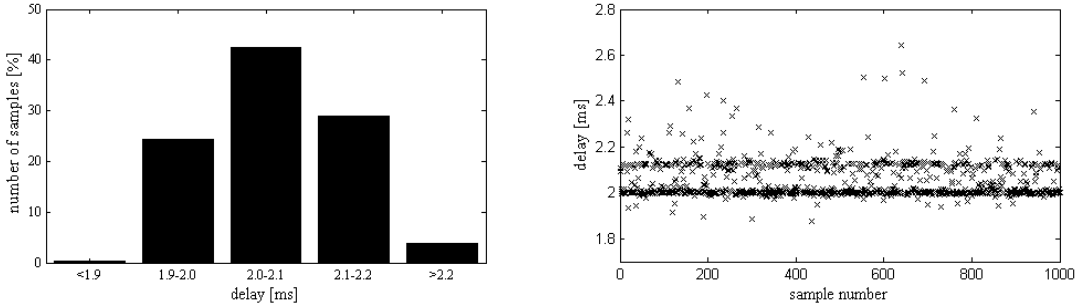


(a) Distribution of the total delays grouped into ranges of 10 milliseconds.



(b) Plot showing all total-delay samples.

Figure 9.1: Results for the total delay during the relay-attack experiment.



(a) Distribution of the proxy delays grouped into ranges of 0.1 milliseconds.

(b) Plot showing the proxy-delay samples between 1.7 ms and 2.8 ms.

Figure 9.2: Results for the proxy delay during the relay-attack experiment.

Total Delay

The total delay describes the time that the PCD waits for a response after sending a command. As we only relay data on the application layer, we measured the delay of an ISO 14443 layer 4 protocol exchange.

In order to measure the total delay, we used the C library of the PCD. Therefore, we applied functions defined in *windows.h*, which support microsecond precision. The measurement started before the ISO 14443 layer 4 protocol-exchange command of the PCD library and stopped after the command was finished. The library as well as the processing of the library commands at the PCD introduce an additional delay that is included in our results. However, this delay is fairly constant and also occurs for regular communication.

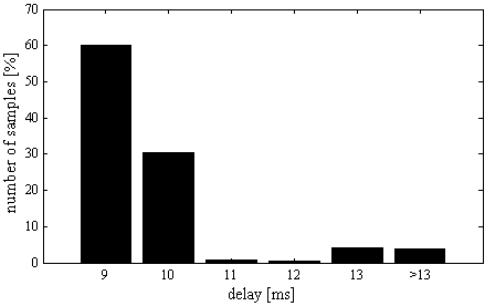
The results for our total delay measurements are shown in Figure 9.1. Figure 9.1a shows the distribution of the delays grouped into ranges of ten milliseconds. Most relays took between 80 ms and 90 ms. However, we see a high variance with some delays being less than 60 ms, and others being over 100 ms. All 1 000 samples are plotted in Figure 9.1b.

Proxy Delay

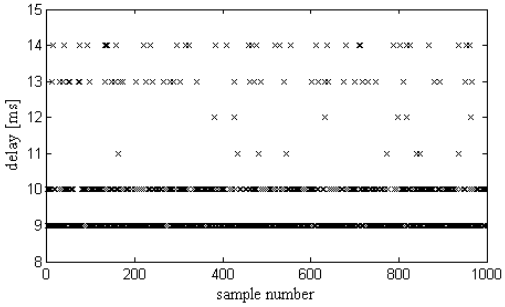
The proxy delay represents the time that the proxy needs to receive and prepare PCD commands for the relay. As we measured the proxy delay at the PCD, it also includes the RFID communication and an additional delay due to the C library and the PCD itself. However, those delays are fairly constant and also exist for regular communication. Therefore, we left those delays included in the results.

In order to measure the proxy delay, we changed the proxy implementation. We removed the relaying procedure and replaced it with a procedure that generates a fixed response. Therefore, we only measured proxy delay without any delay because of the relay.

Figure 9.2 shows the results of our proxy delay measurements. Figure 9.2a shows that the proxy delay mainly ranges between 1.9 ms and 2.2 ms and has a low variance. A plot showing the proxy-delay samples in the range of 1.7 ms and 2.8 ms is shown in Figure 9.2b.



(a) Distribution of the mole delays.



(b) Plot showing all mole-delay samples between 8 ms and 15 ms.

Figure 9.3: Results for the mole delay during the relay-attack experiment.

Mole Delay

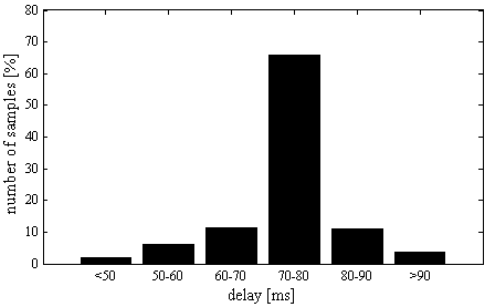
The mole delay represents the processing delay at the mole (mobile phone), two RFID-link delays, and the processing time of the PICC. Due to limitations in Java ME it is not possible to distinguish between processing delay of the phone and the RFID communication.

The measurement starts after the first byte from the PCD is available at the Bluetooth channel and stops after the response is flushed out on the Bluetooth connection. In order to measure the mole delay, we used the Java ME function *System.currentTimeMillis()* on the mobile phone. It allows time measurements with a resolution of one millisecond.

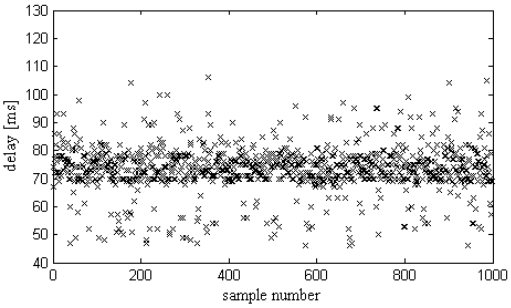
Figure 9.3 shows the results of our measurements. The distribution of the delays can be seen in Figure 9.3a, where 9 ms and 10 ms are most frequent. Interestingly, there are only a few samples that took 11 ms or 12 ms, which we assume has to do with the Java ME implementation for RFID communication. A plot of all mole-delay samples in the range of 8 ms and 15 ms is presented in Figure 9.3b.

Relay-Channel Delay

The relay-channel delay consists of the delay introduced by the two Bluetooth modules (at the mole device and at the proxy device) and the delay of the Bluetooth transmission



(a) Distribution of the relay-channel delays grouped into ranges of 10 milliseconds.



(b) Plot showing all relay-channel delay samples.

Figure 9.4: Results for the relay-channel delay during the relay-attack experiment.

itself.

In order to measure the relay-channel delay, we used a dedicated timer at the proxy, which was configured at a resolution of 1 millisecond. It started before the first byte was sent to the Bluetooth module and stopped after the last byte was received. The time measured by the timer also contained the delay at the mole. Therefore, we measured the corresponding mole delays and subtracted them accordingly in our results.

In Figure 9.4, we see the results of our measurements. Figure 9.4a shows a similar distribution as seen in the total-delay measurements but reduced by the mole delay and the proxy delay. Figure 9.4b shows a plot of all 1 000 samples of the relay-channel delay.

Delay Evaluation

Table 9.1 shows the statistics of the delays during our relay attack. In the first row, it shows the delays occurring during the communication with a genuine PICC (a MIFARE Plus S card). Note that the total delay is not a sum of the remaining delays because it was gathered in a separate experiment and the boundaries between the delays are not always precise. Moreover, all delays could not be gathered within a single experiment because the measurement of the delays takes additional time and would distort the results. However, the results give a good idea about the composition of the total delay.

We took 1 000 sample measurements of the communication with the genuine PICC. We see that the PICC had an average response time of about 3 milliseconds. Therefore, our relay attack took about 30 times longer than a regular communication and could easily be prevented by a strict timeout implementation. However, if the command sent to the PICC would require complex computational operations, the relay delay would stay fairly constant but the computation delay would increase significantly. This would make it more difficult to find a suitable threshold for a timeout.

We see that the implementation of the proxy is fast and introduces little additional delay. It even responses faster then the genuine PICC. Moreover, it has a low standard deviation of only 0.2 ms. The mole delay is also rather constant with a standard deviation of 1.65 ms.

The relay-channel delay proved to be the dominant factor of our setup. This delay contains not only the Bluetooth transfer itself but also the delay of the two Bluetooth modules at the mole device and at the proxy device. Therefore, we cannot determine precisely how this delay is composed. However, it introduces about 80 % of the total delay.

Table 9.1: Statistical evaluation of the delays during the relay-attack experiment.

Component	**Average delay [ms]**	**Standard deviation [ms]**	**Min [ms]**	**Max [ms]**
PICC	3.01	0.23	2.49	9.03
Proxy	2.07	0.20	1.87	6.38
Mole	9.74	1.65	9.00	38.00
Relay channel	73.51	8.50	46.00	106.00
Total	**88.36**	**12.41**	**57.65**	**129.92**

9.2 Countermeasure-Evaluation Results

This section evaluates the performance of the proposed protocol to prevent the exploitation of WTXs for relay attacks. We also evaluate the performance of our distance-bounding protocol (Section 6.3) on layer 4 of the ISO 14443 protocol and derive a reasonable threshold and error tolerance. The IAIK DemoTag v3.0 [19] was used for our evaluation.

9.2.1 WTX-Exploitation Countermeasure

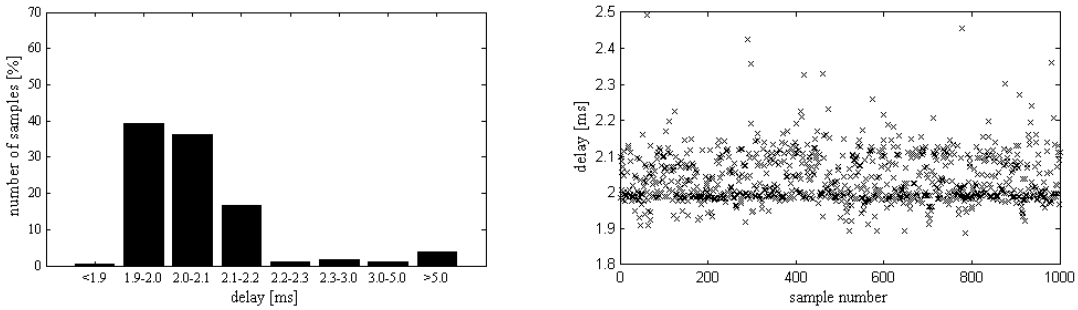
We propose a protocol to prevent the exploitation of Waiting Time Extensions (WTXs) for relay attacks in Section 6.1. The protocol is based on the fact that if an attacker sends a WTX request to the PCD, the PICC has no knowledge of it. Therefore, the PICC can encrypt and append its knowledge about WTXs (WTX counter) to the final response so the PCD can compare with its own knowledge (WTX counter). If we assume a secure encryption scheme, the attacker cannot modify this knowledge.

By activating the WTX-exploitation countermesure on PCD and PICC, we were able to successfully detect the exploitation of WTXs by the attacker. If no WTX was sent by the PICC, the attacker was not able to append a correctly encrypted WTX counter to the message. If the PICC sent a WTX, the attacker was not able to change the appended counter because of the encryption.

The protocol only introduced a small additional delay and only a small additional overhead of two bytes. However, note that we did not implement a real encryption for the WTX counter. Therefore, the protocol could cause additional delay if a complex encryption scheme was implemented.

9.2.2 Distance Bounding

In this section, we present the results for the distance-bounding protocol described in Section 8.5.2. In order to show the behavior of the protocol, we measured the delays of bit-size challenge-response pairs. Because our nonce had a length of 12 byte, this resulted in 48 ($\frac{12}{2} \times 8$) challenge-response pairs. Furthermore, we used those delays to derive a reasonable threshold and error tolerance for the protocol. The measurement was done by the C library at the PCD and contains overhead of the PCD hardware and the library.



(a) Distribution of the delays during the distance-bounding protocol.

(b) Plot showing delay samples of the distance-bounding protocol.

Figure 9.5: Results for the delay during the distance-bounding protocol.

However, if the protocol is implemented on the application layer of a real system, those overheads exist as well.

In our results we only measured challenge-response pairs that where exchanged without a transmission error (*i.e.* no NAK was sent). Our observations showed that the connection between the PCD and real PICCs was far more reliable than the connection between the PCD and the proxy (DemoTag). As we assume that the protocol is implemented on a real PICC, we assume a good connection. Therefore, we only use measurements gathered when no NAKs where sent.

Figure 9.5 shows the results of our measurements. In Figure 9.5a we see that 92.6 % of the delays where below 2.2 milliseconds, which indicates a mainly constant behavior. A plot of all samples within 1.8 ms and 2.5 ms can be seen in Figure 9.5b. The statistical evaluation of the results is presented in Table 9.2. The rather high average and standard deviation result from the high number of delays above five milliseconds (3.7 %).

Table 9.2: Statistical evaluation of the delays during the distance-bounding protocol.

Delay type	**Average delay [ms]**	**Standard deviation [ms]**	**Min [ms]**	**Max [ms]**
Challenge-response protocol	2.23	0.99	1.89	16.35

As we use an RFID-tag emulator (the IAIK DemoTag), the behavior of a real PICC might be different. Different types of PICCs also result in different computation delays and response times. Therefore, we propose to treat PICCs according to their type and performance features. The PICC could transmit its type (or performance features) during the activation sequence of the protocol. Note that this transmission needs to be encrypted so that an attacker cannot modify it. The PCD could then decide which threshold and error tolerance to use for the given PICC.

A possible procedure to decide on the threshold and error tolerance is presented in the following section. The example is based on the data we gathered for the DemoTag.

Threshold and Error-Tolerance Calculation

For the DemoTag, we used a threshold of 2.2 milliseconds because 92.6 % of the measured responses met this threshold ($P_{in\ time}$). As the fastest response was received within 1.89 ms, this would definetly detect any relay that introduces more than 310 microseconds.

The probability of a single response not to meet this threshold ($P_{not\ in\ time}$) is calculated by

$$P_{not\ in\ time} = 1 - P_{in\ time}. \tag{9.2}$$

The probability for at least one of the N iterations (in our case $N = 48$) of the protocol not to meet this threshold ($P_{false\ rejection}$) is calculated by

$$P_{false\ rejection} = 1 - (P_{in\ time})^N. \tag{9.3}$$

The protocol leads to a false-rejection rate of 97.5 %. This means that in 975 of 1 000 cases a legitimate PICC would be rejected because of a delayed response. Hancke and Kuhn [13] proposed strategies to deal with noisy environments for their distance-bounding protocol. We can adjust those measurements to deal with varying response times.

Table 9.3: False-rejection and false-acceptance rates of the distance-bounding protocol for different numbers of allowed delayed responses.

Allowed delayed responses (x)	**False-rejection rate [%]**	**False-acceptance rate $\times 10^{-13}$ [%]**
0	97.504	0.036
1	87.927	0.071
2	69.944	0.114
3	47.908	0.284
4	28.097	0.569
5	14.165	1.137
6	6.186	2.274
7	2.360	4.548
8	0.793	9.095
9	0.236	18.190
10	0.063	36.380
11	0.015	72.760
12	0.003	145.519

In order to reach more reasonable false-rejection rates, we propose to accept a certain number of delayed responses. The probability for a certain number k of responses being delayed can be calculated by

$$P_{k\ responses\ delayed} = ((P_{not\ in\ time})^k \times (P_{in\ time})^{N-k})\binom{N}{k}. \tag{9.4}$$

The probability for at most x responses being delayed is a summation of all probabilities for delayed responses $\leq x$. Therefore, the false-rejection rate can be calculated by

$$P_{false\ rejection} = 1 - \sum_{i=0}^{x}(((P_{not\ in\ time})^i \times (P_{in\ time})^{N-i})\binom{N}{i}). \tag{9.5}$$

Table 9.3 shows the false-rejection rates for different numbers of allowed delayed responses. Moreover, it shows the false-acceptance rates by assuming the attacker relays the allowed number of delayed responses and guesses the remaining challenges. Therefore, it is calculated by

$$P_{false\ acceptance} = \frac{1}{2^{(N-x)}}. \tag{9.6}$$

We see that by allowing 12 delayed responses, we reach a false-rejection rate of only 0.003 % and still keep the false-acceptance rate at 1.455×10^{-11} %. Those rates can be adjusted by allowing more delayed responses, using more challenge-response iterations (longer challenges), or using a different threshold.

9.3 Comparison to Relay Attacks in Literature

In order to evaluate the results of our relay attack, we compare them to other relay attacks in literature. One of the first relay attacks was implemented by Hancke [12]. He used two

UHF antennas to establish a relay channel on which he transmitted the analogue data. By saving the time for digitizing the data, he reached a introduced delay of only 15-20 µs and successfully attacked an ISO 14443 system. However, his attack fails if additional PICCs are in range of the PCD because the answer to the anticollision commands is not synchronized due to the delay.

A more meaningful comparison is the master thesis of Weiß [47]. He performed a relay attack on ISO 14443 systems using NFC mobile equipment, connected by a Bluetooth relay channel. He reached an average introduced delay of 54 milliseconds. However, note that the used NFC mobile equipment is not very practical as it has to be controlled by a PC or by a laptop. Also note that our implementation did not focus on the speed of the relay but rather on the exploitation of the ISO 14443 protocol.

9.4 Summary

In this section, we presented the results of our relay-attack experiment. Using our setup, we saw that ISO 14443 systems without additional security features are vulnerable to relay attacks. By exploiting certain mechanisms of the standard, we further increased the chance for an attacker.

The introduced delays during our relay attack where significantly higher than the delays during regular communication. The main contributors to the relay delay proved to be the Bluetooth modules and the Bluetooth channel. However, the introduced delay was still way below the maximum Frame Waiting Time (FWT) for ISO 14443 systems and did not affect the success of the relay attack.

Finally, we also looked at the proposed countermeasures and their success in preventing relay attacks. The proposed WTX-exploitation countermeasure can prevent the abuse of WTXs by an attacker. The distance-bounding protocol proved to be a good method to detect relay attacks. Although the implementation on the application layer does not allow exact distance measurements, distance bounding makes a relay attack much more difficult because the attacker requires a much shorter introduced delay of less than 310 microseconds. A combination of all proposed countermeasures would lead to a high standard of security.

Chapter 10

Conclusions

In this thesis, we presented the concept of relay attacks on RFID systems. During a relay attack, the attacker uses two devices (*i.e.* the mole and the proxy) to establish a relay channel between the reader and the transponder. The attacker forwards every message between reader and transponder through this channel in order to let the two devices assume that they are in each others transmission range. Cryptographic measures on the application layer are circumvented because the attacker simply forwards the encrypted messages.

The security of the ISO/IEC 14443 standard regarding relay attacks was analyzed in this thesis. Therefore, we performed a relay attack using an off-the-shelf mobile phone with NFC-capabilities as the mole device and an RFID-tag emulator as the proxy device. Both devices where connected through a Bluetooth channel that operates up to 100 meters. Using this hardware, we were able to successfully perform a relay attack on an ISO/IEC 14443 test system. The test system consisted of a Pegoda MF RD 700 reader [37] and a MIFARE Plus S transponder [36].

Our measurements showed that the relay attack introduced an average additional delay of 88 milliseconds. Given that according to the ISO/IEC 14443 standard the transponder can decide on the Frame Waiting Time (FWT) used (up to 4.95 seconds), this is easily fast enough for the attacker to relay the data. In addition to that, we performed a number of experiments in which we exploited different mechanisms of the ISO/IEC 14443 standard. The exploitation of Waiting Time Extensions (WTXs) allowed us to delay the response for an additional 12 seconds. However, we could have stalled the reader for an arbitrary amount of time. We also exploited the error-recovery mechanism of the standard and showed that the additional time for the relay depends on the implementation of the reader and how many recovery tries (Negative Acknowledges, NAKs) it supports. During our experiments we were able to gain 3 seconds of additional time for the relay.

In literature, there exist several countermeasures against relay attacks. However, most of the proposed countermeasures do not comply with the ISO/IEC 14443 standard. We developed an ISO/IEC 14443 compliant protocol to eliminate the exploitation of WTXs. The proposed protocol allows the reader to detect WTXs introduced by the attacker, and therefore prevents their exploitation. We also implemented a distance-bounding protocol on the application layer of the ISO/IEC 14443 protocol. Our experiments showed that distance bounding prevents a large number of relay attacks because it limits the maximum time introduced by the relay. Our implementation of distance bounding prevents relay attacks which introduce more than 310 microseconds. Although the proposed countermeasures do not provide full security, they would make a relay attack significantly more

difficult to perform. We suggest to use several of the proposed countermeasures together to reach better security.

The ISO/IEC 14443 standard is widely used for security-critical applications like payment systems, public transportation, and access-control management. In this thesis, we have shown that those systems are vulnerable to relay attacks. With the wide distribution of RFID technology in mobile phones, those systems become more attractive for attackers. Therefore, future work should focus on protecting those systems against relay attacks.

Appendix A

BTM-222 Bluetooth Board Picture and Schematics

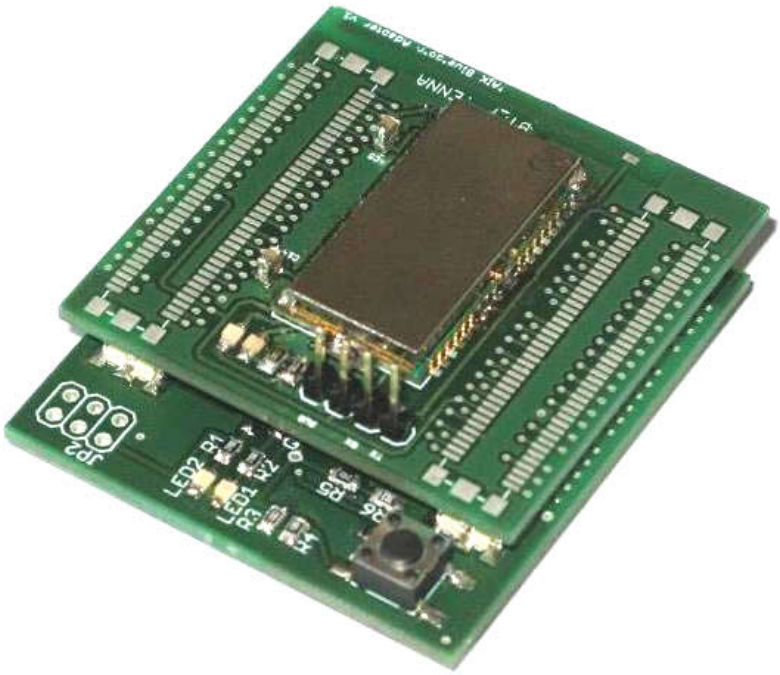


Figure A.1: Picture of the BTM-222 Bluetooth module.

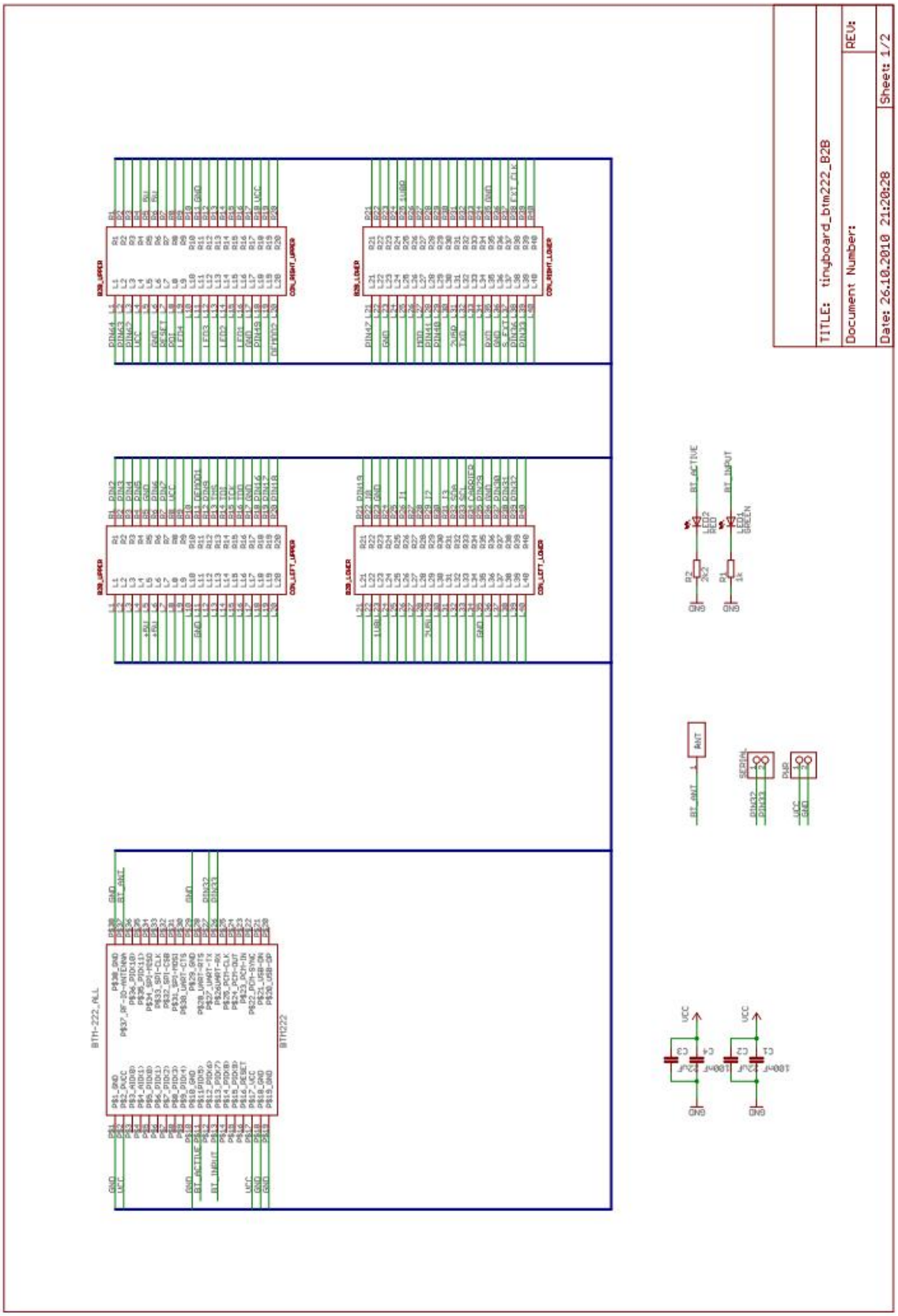


Figure A.2: Schematic of the BTM-222 board (sheet 1).

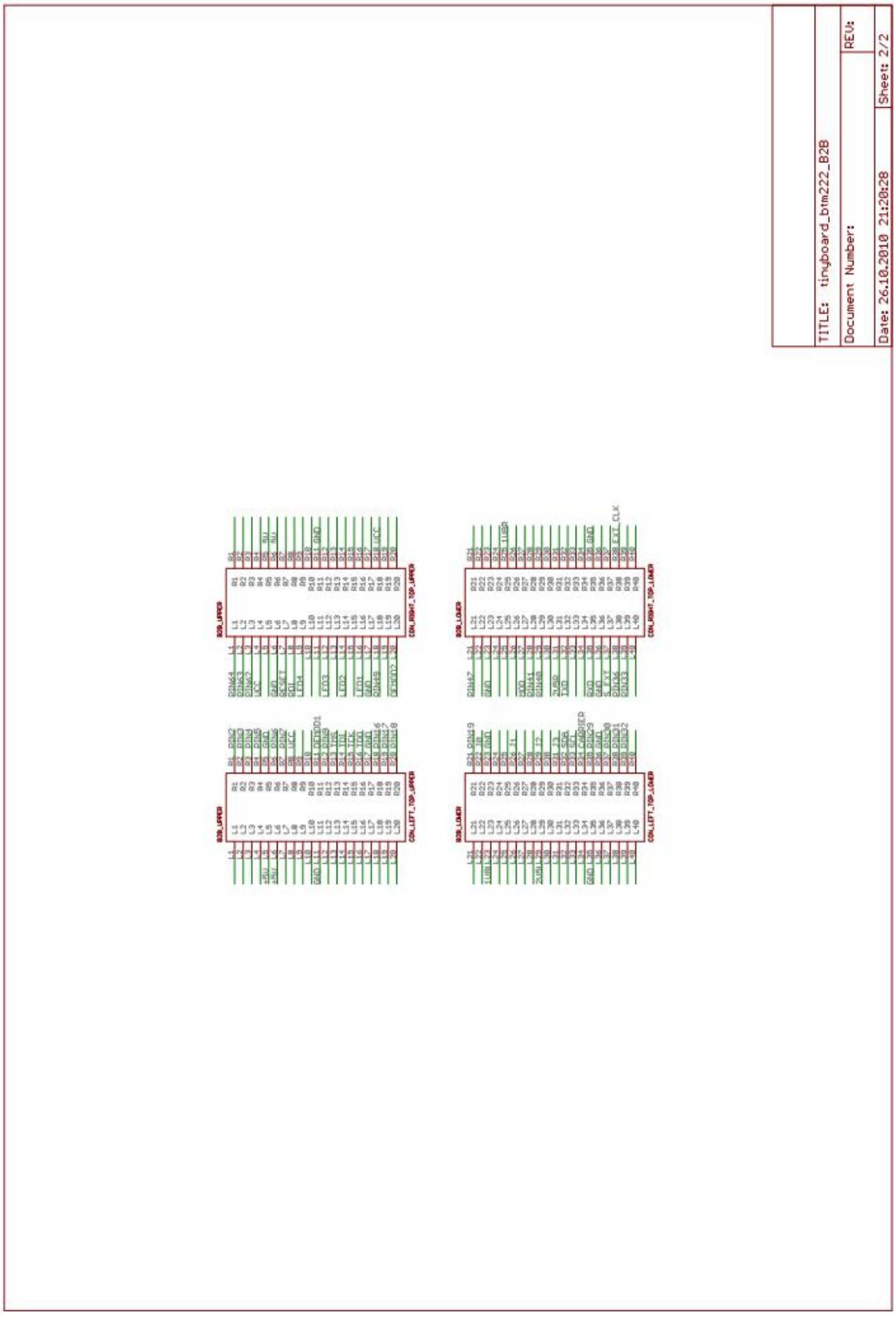


Figure A.3: Schematic of the BTM-222 board (sheet 2).

Appendix B

Definitions

2-FA	Two-Factor Authentication
8PSK	8 Phase-Shift Keying
ACL	Asynchronous Connection-Less
AES	Advance Encryption Standard
APDU	Application Protocol Data Unit
API	Application Programming Interface
ATQA	Answer To Request A
ATS	Answer To Select
BCC	Block Check Character
BTM	Bluetooth Module
CID	Card IDentifier
CRC_A	Cyclic Redundancy Check A
CT	Cascade Tag
DES	Data Encryption Standard
DQPSK	Differential Quaternary Phase-Shift Keying
DR	Divisor Receive
DS	Divisor Send
EAS	Electronic Article Surveillance
ECC	Elliptic Curve Cryptography
EDR	Enhanced Data Rate
EEPROM	Electrically Erasable Programmable Read-Only Memory

EIA	Electronic Industries Association
FDT	Frame Delay Time
FSC	Frame Size Card
FSCI	Frame Size Card Integer
FSD	Frame Size Device
FSDI	Frame Size Device Integer
FWI	Frame Waiting Integer
FWT	Frame Waiting Time
HF	High Frequency
IAIK	Institute for Applied Information Processing and Communications
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
INF	Information
ISO	International Organization for Standardization
Java ME	Java Platform, Micro Edition
JTAG	Joint Test Action Group
LF	Low Frequency
LSB	Least Significant Bit
LAN	Local Area Network
NAD	Node ADdress
NAK	Negative Acknowledge
NFC	Near-Field Communication
NRZ	Non-Return-to-Zero
NVB	Number of Valid Bits
MITM	Man-In-The-Middle
MSB	Most Significant Bit
OSI	Open System Interconnection
PC	Personal Computer
PCB	Protocol Control Byte

PCB	Printed Circuit Board
PCD	Proximity Coupling Device
PICC	Proximity Integrated Circuit Card
PIN	Personal Identification Number
PPS	Protocol and Parameter Selection
RATS	Request Answer To Select
REQA	Request A
RFID	Radio-Frequency Identification
RS-232	Recommended Standard 232
RTT	Round-Trip Time
SAK	Select Acknowledge
SDK	Software Development Kit
SEL	Select Code
SFGI	Start-up Frame Guard Integer
SPP	Serial Port Profile
UART	Universal Asynchronous Receiver Transmitter
UHF	Ultra-High Frequency
UID	Unique IDentifier
USB	Universal Serial Bus
UWB	Ultra-WideBand
VDI	The Association of German Engineers
WUPA	Wake-UP A
WTX	Waiting Time Extension
WTXM	Waiting Time Extension Multiplier

Bibliography

[1] Atmel. *ATxmega256A3*. Available from http://www.atmel.com/dyn/products/product_card.asp?category_id=163&family_id=607&subfamily_id=1965&part_id=4304 (accessed on February 17, 2011).

[2] Atmel. *ATMega128*. Available from http://www.atmel.com/dyn/products/product_card.asp?part_id=2018 (accessed on May 4, 2011).

[3] V. S. Bagad and I. A. Dhotre. *Data Communication Systems*. Technical Publications Pune, 2009.

[4] Calypso. *Calypso Network Association*. http://www.calypsonet-asso.org (accessed on April 19, 2011).

[5] J. Campbell. *V24 RS-232 Kommunikation*. Sybex-Verlag, Köln, Germany, 1986.

[6] J. H. Conway. *On Numbers And Games*. The Academic Press, London, UK, 1976.

[7] C. Douligeris and D. N. Serpanos. *Network Security: Current Status and Future Directions*. John Wiley & Sons, Inc., 2007.

[8] Eclipse. *Eclipse Pulsar*. Available from http://www.eclipse.org/pulsar/ (accessed on February 17, 2011).

[9] K. Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*. John Wiley & Sons, Ltd., Chippenham, Wiltshire, UK, 2010.

[10] L. Francis, G. P. Hancke, K. Mayes, and K. Markantonakis. Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones. In *Radio Frequency Identification: Security and Privacy Issues*, volume 6370 of *Lecture Notes in Computer Science*, pages 35–49. Springer Berlin / Heidelberg, 2010.

[11] Fujitsu. *Dense Wavelength Division Multiplexing Tutorial*, 2002. http://www.fujitsu.com/downloads/TEL/fnc/pdfservices/dwdm-prerequisite.pdf (accessed on April 28, 2011).

[12] G. P. Hancke. *A Practical Relay Attack On ISO 14443 Proximity Cards*. Technical report, 2005. Available from http://www.rfidblog.org.uk/hancke-rfidrelay.pdf (accessed on May 4, 2011).

[13] G. P. Hancke and M. G. Kuhn. An RFID Distance Bounding Protocol. In *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in*

Communications Networks, pages 67–73, Washington, DC, USA, 2005. IEEE Computer Society.

[14] G. P. Hancke, K. Mayes, and K. Markantonakis. Confidence in Smart Token Proximity: Relay Attacks Revisited. *Computers & Security*, 28(7):615–627, 2009.

[15] Y.-C. Hu, A. Perrig, and D. B. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computerand Communications. IEEE Societies*, pages 1976 – 1986 vol.3, 2003.

[16] IEC. *International Electrotechnical Commission*. `http://www.iec.ch/` (accessed on April 20, 2011).

[17] IEEE. *IEEE 802.11 - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2007. Available from `http://standards.ieee.org/getieee802/download/802.11-2007.pdf` (accessed on April 27, 2011).

[18] IEEE. *IEEE 802.3: LAN/MAN CSMA/CDE (Ethernet) Access Method*, 2008. Available from `http://standards.ieee.org/getieee802/802.3.html` (accessed on April 2, 2011).

[19] Institute for Applied Information Processing and Communications, Graz University of Technology. *IAIK Demo Tag*. Available from `http://www.iaik.tugraz.at/content/research/rfid/tag_emulators/` (accessed on February 17, 2011).

[20] ISO. *International Organization of Standardization*. `http://www.iso.org` (accessed on April 21, 2011).

[21] ISO/IEC. *ISO/IEC 7810 - Identification cards - Physical characteristics*, 2003.

[22] ISO/IEC. *ISO/IEC 7816 - Identification cards - Integrated circuit cards*, 2006.

[23] ISO/IEC. *ISO 14443: Identification cards - Contactless integrated circuit(s) cards - Proximity cards*, 2010.

[24] ISO/IEC. *ISO/IEC 14443 - Identification Cards - Contactless Integrated Circuit(s) Cards -Proximity Cards - Part 1: Physical Characteristics*, 2010.

[25] ISO/IEC. *ISO/IEC 14443 - Identification Cards - Contactless Integrated Circuit(s) Cards -Proximity Cards - Part 2: Radio Frequency Power and Signal Interface*, 2010.

[26] ISO/IEC. *ISO/IEC 14443 - Identification Cards - Contactless Integrated Circuit(s) Cards -Proximity Cards - Part 3: Initialization and Anticollision*, 2010.

[27] ISO/IEC. *ISO/IEC 14443 - Identification Cards - Contactless Integrated Circuit(s) Cards -Proximity Cards - Part 4: Transmission Protocol*, 2010.

[28] Z. Kfir and A. Wool. Picking Virtual Pockets using Relay Attacks on Contactless Smartcard. In *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pages 47–58, Washington, DC, USA, 2005. IEEE Computer Society.

[29] I. Khalil, S. Bagchi, and N. B. Shroff. LITEWORP: A Lightweight Countermeasure for the Wormhole Attack in Multihop Wireless Networks. In *International Conference on Dependable Systems and Networks, 2005. Proceedings.*, pages 612 – 621, 2005.

[30] A. Mitrokotsa, M. Rieback, and A. Tanenbaum. Classifying RFID attacks and defenses. *Information Systems Frontiers*, pages 1–15, 2009.

[31] J. Munilla and A. Peinado. Distance Bounding Protocols for RFID Enhanced by Using Void-Challenges and Analysis in Noisy Channels. *Wireless Communication and Mobile Computing*, 8(9):1227–1232, 2008.

[32] NFC Times. *Vendor Group Seeks to Crack Mifare Dominance*, 2010. Available from `http://www.nfctimes.com/report/vendor-group-seeks-crack-mifare-dominance` (accessed on April 18, 2011).

[33] Nokia. *Nokia 6212 classic*. Available from `http://www.nokia.at/produkte/alle-modelle/nokia-6212-classic` (accessed on April 13, 2011).

[34] Nokia. *Nokia PC Suite*. Available from `http://www.nokia.at/support/software/nokia-pc-suite` (accessed on May 3, 2011).

[35] NXP Semiconductors. *MIFARE Plus Leaflet*. Available from `http://www.mifare.net/index.php/download_file/view/8/` (accessed on May 3, 2011).

[36] NXP Semiconductors. *MIFARE*. Available from `http://www.mifare.net` (accessed on April 13, 2011).

[37] NXP Semiconductors. *Pegoda MF RD 700 Datasheet*, 2011. Available from `http://www.nxp.com/documents/data_sheet/066120.pdf` (accessed on February 17, 2011).

[38] C. Paar, J. Pelzl, and B. Preneel. *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer Verlag Berlin Heidelberg, 2010.

[39] R. Poovendran and L. Lazos. A Graph Theoretic Framework for Preventing the Wormhole Attack inWireless ad hoc Networks. *Wireless Netwetworks*, 13(1):27–59, 2007.

[40] Rayson. *Rayson Technology Co., Ltd.* `www.rayson.com` (accessed on May 5, 2011).

[41] Rayson. *BTM-222 Datasheet*, 2005. Available from `http://www.taiwantrade.com.tw/resources/member/276716/productcatalog/b9ee22e4-e94a-45cd-8db0-ac88b3a11a75_BTM222%20DataSheet.pdf` (accessed on May 5, 2011).

[42] J. Reid, N. J. M. Gonzalez, T. Tang, and B. Senadji. Detecting Relay Attacks with Timing-based Protocols. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 204–213, New York, NY, USA, 2007. ACM.

[43] M. Sauter. *Grundkurs Mobile Kommunikationssysteme*. LinkFriedr. Vieweg & Sohn Verlag — GWV Fachverlage GmbH, Wiesbaden, 2006.

[44] H. W. Silver and M. J. Wilson. *The ARRL Handbook For Radio Communications*. The American Radio Relay League, Inc., 2010.

[45] The Siemon Company. *Propagation Delay and Delay Skew*. Available from `http://www.siemon.com/us/white_papers/97-06-03-delayskew.asp`(accessed on September 15, 2010.

[46] Verein Deutscher Ingenieure. *Richtlinie VDI 4470 (Entwurf): Warensicherungssysteme*, 2009.

[47] M. Weiß. *Picking Virtual Pockets using Relay Attacks on Contactless Smartcard Systems*. Master thesis, Technische Universität München, May 2010. Available from `http://www.sec.in.tum.de/assets/studentwork/finished/Weiss2010.pdf` (accessed on May 5, 2011.