

A RELAY PREVENTION TECHNIQUE FOR NEAR FIELD
COMMUNICATION

A Thesis

by

ERIC S. WILCOX

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Chair of Committee,	Riccardo Bettati
Co-Chair of Committee,	Radu Stoleru
Committee Member,	A. L. Narasimha Reddy
Head of Department,	Dilma Da Silva

August 2015

Major Subject: Computer Science

Copyright 2015 Eric S. Wilcox

ABSTRACT

The use of near field communication (NFC) has expanded as rapidly as Bluetooth or similar technologies and shows no signs of slowing down. It is used in many different systems such as contactless payment processing, movie posters, security access and passport identification. NFC enabled devices include cell phones, credit cards and key chains. With the spread of any new technology come security vulnerabilities that malicious users will try to exploit. NFC is particularly vulnerable to what is known as a relay attack. The relay attack is similar to the man-in-the-middle attack but the data need not be unencrypted to be vulnerable. The relay attack is currently undetectable and unstoppable. Many solutions have been proposed but no real-world solution has been found that does not require significant changes to the NFC protocol, or even the hardware. In this work we propose a technique that uses careful timing analysis of tag communication to identify a transaction as dangerous and thus set off an alert of the potential threat. This could be built into mobile devices and readers already deployed and provide a level of security to the market not currently available while maintaining the protocols set forth by the ISO. A proof of concept has been built and tested on custom hardware as well as on an Android Nexus 4 to detect and prevent the relay attack. In this thesis we give an overview of security issues in NFC communication, describe the relay attack in detail, present our timing based countermeasures and its implementation, and give results of our evaluation of timing based relay prevention.

DEDICATION

To my family who support me in all things.

ACKNOWLEDGEMENTS

I would like to thank my adviser and committee chair, Dr. Riccardo Bettati. He has shown great patience and dedication while guiding me through this work. His support, financially, morally and educationally, from the beginning to the late nights throughout this project was truly invaluable. His trust and faith in me accomplishing tasks which seemed daunting at times was immensely encouraging.

I am grateful to my family who helped guide me down this path and encouraged me at every turn, especially when things looked like they were going sideways.

I would also like to thank my committee, Dr. Narasimha Reddy and Dr. Radu Stoleru, as well as Dr. Jyh-Charn (Steve) Liu (who graciously sat in at the last minute) for their time and suggestions on this project.

NOMENCLATURE

AES	Advanced Encryption Standard
CE	Consumer Electronics
CM	Centimeters
ECDH	Elliptic Curve Diffie-Hellman
ECMA	European Association for Standardizing Information and Communication Systems
I ² C	Inter-Integrated Circuit (I-squared-C)
ICT	Information and Communication Technology
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
KBPS	Kilobit Per Second
MHz	Megahertz
MS	Milliseconds
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
PC	Personal Computer
PCD	Proximity Coupling Device
PICC	Proximity Inductive Coupling Card
RFID	Radio-frequency Identification
SCH	Secure Channel service
SPI	Serial Peripheral Interface

SSE Shared Secret Service
UHF Ultra High Frequency
UID Unique Identifier
USB Universal Serial Bus

TABLE OF CONTENTS

	Page
ABSTRACT	ii
DEDICATION	iii
ACKNOWLEDGEMENTS	iv
NOMENCLATURE	v
TABLE OF CONTENTS	vii
LIST OF FIGURES	ix
LIST OF TABLES	xi
1. INTRODUCTION	1
1.1 Participants	2
1.2 Importance of Problem	5
1.3 Remote Eavesdropping of NFC Communication	7
1.3.1 Theoretical Studies of Eavesdropping Ranges	7
1.4 Relay Attacks on NFC Communication	10
1.5 Countermeasures against Relay Attacks	13
2. CURRENT WORK	15
3. IMPLEMENTING A SOLUTION	19
3.1 Equipment Setup	21
3.1.1 Tag Timing	22
3.1.2 Analyzing Tag Communication	23
3.2 Tag Analysis	25
3.3 Building a Relay	27
3.4 Preventing the Relay Attack	28
3.4.1 Android NFC	30
4. EVALUATION	35
5. CONCLUSION	38

REFERENCES	40
APPENDIX A. VERIFICATION OF TAG TIME MEASUREMENTS	46
APPENDIX B. REDUCING THE TIMING IN ARDUINO CODE	48

LIST OF FIGURES

FIGURE	Page
1.1 NFC Forum specifications for peer-to-peer mode, reader/writer mode and card emulation mode [17]. Communication shown across different operating modes.	4
1.2 Man-in-the-middle attack scenario.	11
1.3 Relay attack example. Tag A and Reader D are authentic. C is the emulated tag (proxy) and B is the fake reader (mole). B and C are connected over some communication medium [33].	12
3.1 Read request from host controller to NFC radio to receive a section of data from the tag (16 bytes at a time - 1 section).	24
3.2 Reduced delay on tag time read, timing from logic analyzer.	25
3.3 Mifare Ultralight A tags page and section setup. One section consists of 4 pages of data, or 16 bytes.	26
3.4 Timing of 1 section of a tag read operation, 1 section = 4 pages (x axis). 5 different sections read here. Measurement (y axis) is in milliseconds (<i>ms</i>).	27
3.5 Arduino and two PN532 boards used to create our NFC relay system.	28
3.6 Timing analysis of relay attack using logic analyzer. Three different PN532 boards are pictured here, top: security point, middle: proxy on relay, bottom: mole on relay.	30
3.7 Code change in the <i>BasicTagTechnology.java</i> file to time the sections.	31
3.8 Histogram plot of tag read times in Android. The majority of reads are at about 4.5 <i>ms</i> or less. Y axis is sections read over 62 tags, X axis is time for read on each section.	32
3.9 Android Architecture [44] with timing shown. Timing starts before HAL layer which sits between the kernel and library layer.	34

B.1 Graph showing the timing results of sections being read with the extra 5 *ms* jump. X axis is section being read, Y axis is time in milliseconds. 49

LIST OF TABLES

TABLE		Page
1.1	Maximum downlink eavesdropping ranges for BER of 0.1% [39]. . . .	8
1.2	Maximum downlink eavesdropping ranges for BER of 0.01% [39]. . . .	8
1.3	Maximum uplink eavesdropping ranges for BER of 0.1% [39].	9
1.4	Maximum uplink eavesdropping ranges for BER of 0.01% [39].	9
4.1	Complete Tag Reads. Tag failure if one or more section takes over 10 <i>ms</i> to read.	37
4.2	Timing results per section with and without AnTuTu benchmark running. Average time of a successful or failed read.	37
A.1	Tag timing using <i>micros()</i> function on Arduino. All times are in milliseconds (<i>ms</i>).	46

1. INTRODUCTION

Near field communication is a form of wireless radio communication designed to work over a very short range, usually up to 10 cm [30]. It was originally based off of RFID (Radio-frequency identification) [36] and operates at 13.56 MHz communicating very small amounts of data [25], at rates between 106 and 424 kbps. NFC is designed to communicate this small amount of information wirelessly between two devices, commonly between a passive *tag* and an active *reader*. NFC has been growing significantly in popularity since its development and can be found in a large number of devices, including transit cards, movie posters, hobby tags, mobile devices, business cards, cell phones, health care devices, passports, credit cards, contactless payment terminals and even identification cards used to access secure locations [17]. The technology has gained significant popularity especially as a payment method and for security access. The disbursement of NFC technology shows no signs of slowing down [43].

Whether the devices employing NFC technology are for fun, or for more sensitive communication, security is always going to be a concern. With movie posters, where a user may just tap his or her phone to visit a web site promoting the movie, malicious parties could simply add a tag to the poster taking the user to their website or attempt to activate functions not intended. If more secure transactions are taking place, such as financial, the need for security is even higher [35]. Credit card fraud is already widespread [19] and moving to a contactless medium for conducting business adds another layer of complexity [30]. The security of NFC is a significant concern and as the use of this technology grows, that concern grows with it [7].

1.1 Participants

NFC has two basic types of devices that communicate, passive and active. *Active devices* have their own power source and can usually act as either a reader or a tag, some with additional capabilities depending on how they are programmed. *Passive devices* do not have a power source but instead get their power through electromagnetic induction when brought within range of an active device [26]. An example of this would be the passive NFC chip embedded in a credit card which can be brought into the range of a credit card payment terminal (an active device). Bringing the credit card into close proximity to the terminal would activate the NFC inside the card, powering it and beginning an exchange of data between the two devices. The typical range for communication is up to about 10 cm between devices following the ISO 14443 standard.

NFC devices have three basic modes depending on the type of device. Active devices can work in *target mode*, *initiator mode* or *peer-to-peer mode* where passive devices work only in target mode. An *initiator* is a reader or electrical component of some sort that generates the electromagnetic field that in turn provides the power for passive devices. *Peer-to-peer mode* involves two active devices with their own power supply such as two cell phones. Passive devices are commonly thought of as tags but can exist in a variety of other mediums such as credit cards, transit cards, stickers and even wearable rings [17].

There are several different types of tags available on the market today, all of which fall into four main categories [17, 36]:

- Type 1 Tags based on ISO/IEC 14443A. These tags can be read from or written to and re-written. They can be locked to prevent re-writing. They hold between 48 bytes and 2 Kbytes of data. They communicate at 106 kbps.

Innovision produces one such tag product called Topaz [36]

- Type 2 Tags based on ISO/IEC 14443A. These tags can be read from or written to and re-written. They can be locked to prevent re-writing. They hold between 48 bytes and 2 Kbytes. They communicate at 106 kbps. NXP produces the chips for tags called Mifare Ultralight [36].
- Type 3 Tags based on the Japanese Industrial Standard (JIS) X 6319-4 or FeliCa. They are configured at the manufacturer to be either readable and re-writable or read only. Memory is variable but should cap at about 1 Mbyte. They communicate at 212 kbps. Sony produces the FeliCa tag [36].
- Type 4 Tags are fully compatible with the ISO/IEC 14443 standards, A and B. They are configured at the manufacturer to be either readable and re-writable or read only. They hold up to 32 Kbytes and communication can occur according to the ISO/IEC 14443 or the ISO 7816-4 with speeds up to 106 kbps. Type A & B cards differ by initialization procedures and coding, but use the same transmission from Part 4 of the standards. NXP Semiconductor produces both DESfire and SmartMX tags [36].

There is an additional type of tag called the Mifare Classic which does not fall into these four categories but, with its chip from NXP Semiconductor, it will communicate with most NFC devices [17] at some level. Figure 1.1 shows the various forms of NFC messages that can be communicated and the communication mode allowed for them. For example, basic tag operations for type 1-4 tags are done in reader and writer mode, but some messages can be passed in peer-to-peer mode also.

The International Organization for Standardization (ISO) is an independent membership based organization of 163 member countries that sets standards for prod-

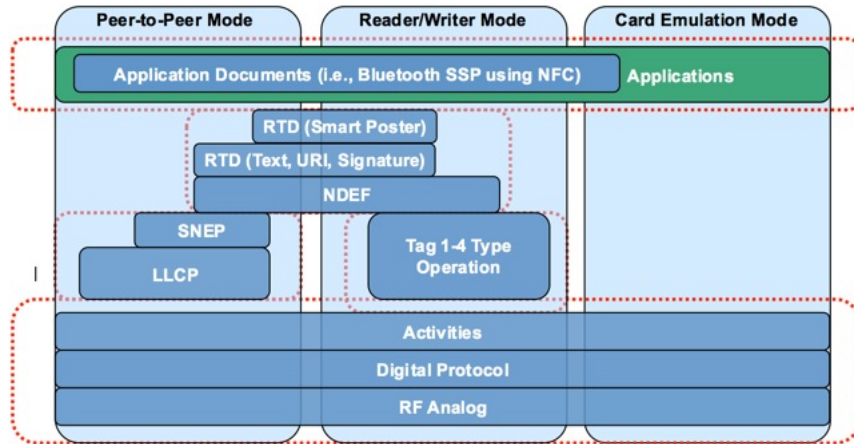


Figure 1.1: NFC Forum specifications for peer-to-peer mode, reader/writer mode and card emulation mode [17]. Communication shown across different operating modes.

ucts and services in an attempt to ensure interoperability, quality, safety and efficiency [27]. Included among their 19,500+ publications are specifications for NFC communication, ISO/IEC 14443: Identification cards - Contactless integrated circuit cards - Proximity cards [26]. There are four parts to the standard detailing NFC as follows:

- ISO/IEC 14443-1:2008 Part 1. Physical characteristics of proximity cards (PICC) [26]. This details the physical traits of the devices, how they hold up under certain circumstances and how they will function.
- ISO/IEC 14443-2:2008 Part 2. Radio frequency power and signal interface [26]. This details how power will be transferred and communication will occur between proximity cards and proximity coupling devices (PCD).
- ISO/IEC 14443-3:2008 Part 3. Initialization and anti-collision [26]. This includes how a PCD will react in the presence of multiple cards at the same time, including how they will initially connect and what will happen when there is

interference.

- ISO/IEC 14443-4:2008 Part 4. Transmission Protocol [26]. This section details the transmission protocols which will be used including the needs of the contactless environment and specifications on the activation and deactivation sequences for type A and type B cards.

ECMA International was originally the European Computer Manufacturers Association but changed to a standards organization and kept the ECMA title. They produce industry driven standards for Information and Communication Technology (ICT) and the Consumer Electronics (CE) industry, while cooperating with the ISO [15]. Through cooperation between ISO and ECMA the standards are published and acknowledged by each other and both work towards the same objective of interoperability and facilitation of standardization [15, 27]. The basic standards for the operational ranges, initialization and radio frequencies were established by ISO [26] and ECMA has published documents relating to the cryptographic standards, wireless interface and other material similar to the ISO 14443 documents [10–14].

1.2 Importance of Problem

With the widespread disbursement of NFC technology, prevention of malicious activity is a very real concern. One of the major benefits of the technology is the ease with which transactions can occur through the contactless state, which also becomes one of the major downfalls. As with any technology where the transfer of potentially sensitive data can occur, those with malicious intent will try to intervene. This can take several forms. Eavesdropping is secretly listening into a conversation usually without the knowledge of the victim. Eavesdropping on communication and stealing sensitive data, such as credit card or transit token information, security keys and even personal passport data, is a significant threat with NFC [7].

Other common threats associated with NFC have already been discussed, and following is a brief summary of these threats addressed in a recent paper [7]. Tags and emulated devices have a unique identification number (UID) that are used to identify the tag. These UID numbers are set by either the manufacturer in passive devices or by the active device when they are initialized. They can be thought of as a label. The UID is inherently unsafe as it is transmitted in the open regardless of the rest of the communication, mainly to allow for proper handling of multiple tags being present in the same field. Cloning a tag, the process of creating an exact duplicate is a possibility. The UID numbers on passive devices are encoded at the manufacturer, are set as unchangeable, and are supposed to be controlled. There are devices however that can emulate these numbers [31] such as the ProxMark3 [41], and some manufacturers in China are offering tags with a programmable UID [32]. This prevents the UID from being used for any type of security. Denial of service attacks are a concern where mobile devices can either be locked up while in an NFC field or communication could be completely prevented [7]. Eavesdropping in both peer-to-peer and card emulated mode can be a problem even though distances are more limited [30]. Identity authentication is a process that lets one device know who the other device is when communicating. If identity authentication is used, the manufacturers tag identity would need to be known, and even do some of its own verification, otherwise a nearby malicious device could steal data simply when a user tried to read a movie poster. Phishing is where a malicious entity tries to steal sensitive information from a user by simulating a bona-fide source, such as a fake web page to log into a bank account. Once the sensitive data is entered into the fake site the malicious party then will then have access to the account. Ticket cloning for e-payment systems can present two problems. First, tickets could be issued and then cloned prior to verification creating many fake but valid tickets. Second, tickets

could be cloned from the verified original and then used from each separate source, exhausting its limit more quickly than intended.

One of the inherent security structures built into NFC technology is the proximity for communication [26]. Because tags are only able to communicate over about 10 cm eavesdropping on the communication can prove to be challenging. Kfir has already shown that distance can actually be increased to at least 50 cm [30]. While any attack involving the transfer of secure information is a concern, the relay attack is one of the most critical.

1.3 Remote Eavesdropping of NFC Communication

Eavesdropping distances are varied and the true distances allowed are still being debated [39]. Some reports have even made false statements claiming distances of 30 feet were successfully achieved, but the experiment only detected communication 30 feet away between devices that were only inches apart communicating with each other [30]. The data in this transmission could not actually be eavesdropped. Regardless of ranges the threat presented by remote eavesdropping is high, and by increasing the ranges that NFC can communicate increases the risk posed by relay attacks as well [30].

1.3.1 Theoretical Studies of Eavesdropping Ranges

A study of the available literature points to the possibility of receiving useful NFC signals from a significant distance. For example, Pfeiffer *et al.* [39] give a number of theoretical bounds on the distance limits for eavesdropping on 14443 class A communication. Assuming that the attacker wishes to capture 64-byte frames, for example, the bit-error-rate (BER) would have to be in the order of 0.01 or 0.001 to achieve success rates above 90% in the frame detection. They assume two types of loop antennas: Antenna type *Reader 1* has a radius of 3cm and a zero-distance

magnetic field strength of 1.5A/m (rms), while antenna type *Reader 2* has a radius of 7.5cm and magnetic field strength 7.4 A/m (rms). Their results for eavesdropping on the downlink signal (i.e. the signal generated by the NFC reader) are summarized in Table 1.1 and Table 1.2.

demodulation	Noise Source		
	Business	Residential	Galactic
Reader 1			
non-coherent	7.9m	12.8m	76.3m
coherent	10.9m	18.4	107.8m
Reader 2			
non-coherent	ca. 0.6km	ca. 1.0km	ca. 6km
coherent	ca 0.9km	ca. 1.5km	ca. 8.5km

Table 1.1: Maximum downlink eavesdropping ranges for BER of 0.1% [39].

demodulation	Noise Source		
	Business	Residential	Galactic
Reader 1			
non-coherent	7.2m	10.5m	63.4m
coherent	8.8m	15.2m	89.4m
Reader 2			
non-coherent	ca. 0.5km	ca. 0.9km	ca. 5km
coherent	ca 0.7km	ca. 1.2km	ca. 7km

Table 1.2: Maximum downlink eavesdropping ranges for BER of 0.01% [39].

The theoretical bounds on *uplink* (i.e. the signal generated by the tag) eavesdropping are significantly smaller, however, as seen in Table 1.3 and Table 1.4.

These numbers contradict previous results from other studies, for example [37], where significantly longer eavesdropping ranges were predicted. This discrepancy is

demodulation	Noise Source		
	Business	Residential	Galactic
1.5 A/m (rms)			
non-coherent	2.8m	3.4m	7.2m
coherent	3.2m	3.9m	9.4m
4.5 A/m (rms)			
non-coherent	2.0m	2.4m	4.7m
coherent	2.2m	2.7m	5.5m

Table 1.3: Maximum uplink eavesdropping ranges for BER of 0.1% [39].

demodulation	Noise Source		
	Business	Residential	Galactic
Reader 1			
non-coherent	2.6m	3.2m	6.6m
coherent	3.0m	3.6m	7.7m
Reader 2			
non-coherent	1.8m	2.2m	4.4m
coherent	2.1m	2.5m	5.1m

Table 1.4: Maximum uplink eavesdropping ranges for BER of 0.01% [39].

due largely to the low bit error rates that are assumed by Pfeiffer *et al.*. The other studies assume that communication can be eavesdropped at bit error rates that are significantly higher than those assumed by Pfeiffer *et al.*. These authors discuss two ways that would enable successful communication at comparatively high error rates:

1. The use of forced retransmissions of each frame by selectively interfering with the transmission of single bits. In this way the detection can be improved despite high bit error rates.
2. The use of wires, wiring in walls, door frames, etc., to increase the range through coupling effects and so in effect decrease the bit error rates at range.

1.4 Relay Attacks on NFC Communication

The relay attack has been well described as an attacker burrowing the victims card [21]. The man-in-the-middle attack is another common way of picturing the relay attack. A man-in-the-middle attack is shown in Figure 1.2. Here an intruder intercepts communication between two devices and can listen in, copy, and change data as she sees fit before forwarding it on to the intended target. The relay attack is similar to this where the unwanted party 'sits' in the middle of the communication. Different to the man-in-the-middle attack, the tag owner and the reader might never even intend for the communication to happen in the first place. The intruder launches a relay attack by placing a malicious reader near the tag owners authentic tag and a malicious tag near the authentic reader. The malicious tag and the malicious reader are connected and form a link to pass all activity back and forth between the authentic tag and authentic reader. The relay can be connected over any medium such as the Internet or Bluetooth, and all traffic between the authentic tag and authentic reader is transferred directly to the other end. Figure 1.3 illustrates a relay attack. Here we can see the malicious tag C comes into range of the authentic reader D which requests communication. This request is passed through the relay to the malicious reader B and the request is made to the authentic tag A. Tag A responds and that data is transferred back through the link to the authentic reader D. The authentic tag and authentic reader both believe they are speaking directly to the authentic device in close proximity. This faith of proximity is bypassed by the relay, and the inherent security feature fails. Because tags are designed to be activated without user interaction they are particularly susceptible to this attack. A malicious party could bring one end of their relay up to an authentic tag while it is in a pocket or bag. Without the owner of the tag knowing, a connection between

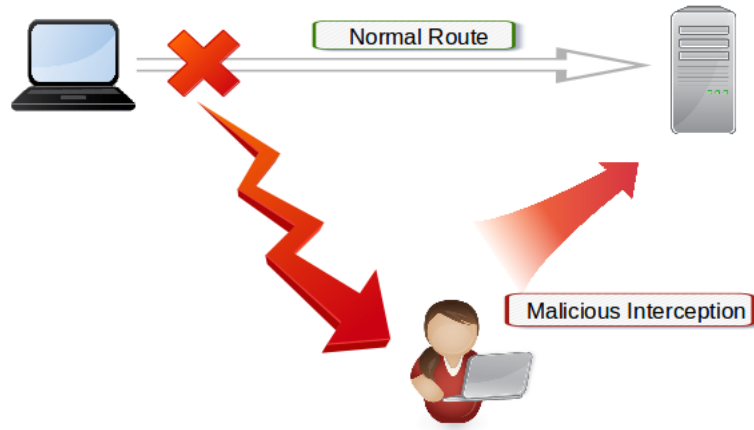


Figure 1.2: Man-in-the-middle attack scenario.

tag and reader can be established and a complete transaction could be processed in seconds. Neither of the authentic devices on either end would be aware a relay attack had even happened [28]. If this attack is performed efficiently the victims will never know it occurred until it is far too late. This would take a bit of set up by the intruder, as she would have to get close to the victims tag with the proxy reader disguised, and would have to place a proxy tag at the other end that would appear close to what would be expected at the authentic reader if it was under any scrutiny [24]. While proper setup would be required in this scenario, the setup is not particularly difficult. This relay attack example is both difficult to detect and hard to stop [28]. The relay attack as presented is a significant security threat to the technology that needs to be addressed.

Cryptography protocols, the encryption of data for transmission, can be implemented to protect NFC against various forms of attacks [25]. Having two devices lacking prior knowledge of secret keys, the standards suggest a public key exchange scheme which is used to establish a Shared Secret Service (SSE) and Secure Channel Service (SCH) [9, 10]. The two devices would share data publicly and through a

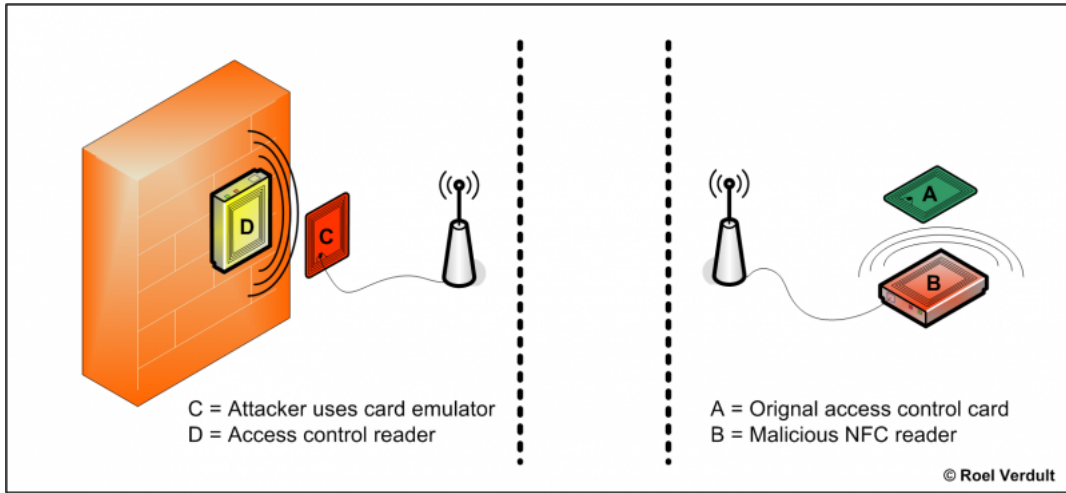


Figure 1.3: Relay attack example. Tag A and Reader D are authentic. C is the emulated tag (proxy) and B is the fake reader (mole). B and C are connected over some communication medium [33].

combination of this data establish a key that is then used to encrypt and decrypt the rest of the communication. Even using these cryptographic methods can not protect against man-in-the-middle attacks as it lacks device authentication [9] and all data can still just be transferred between endpoints with a relay attack [24, 28]. *Device authentication* is more secure in peer-to-peer mode using public key cryptography, but active devices do not check identification of passive targets and remain susceptible to certain attacks. For example, a device in close proximity to a tag that can store key data during setup and later use it to take malicious actions, like activating Bluetooth and transferring data to the victim device at will [7, 9]. The standards actually suggest that the proximity of NFC provides risk at a low enough level to be acceptable and that the end user should remain cautious of any implementation vulnerability [9]. Security through proximity still remains a problem for relays.

While the ISO 14443 standards create the protocol for the hardware and tag communication, they do not include any protocols to deal with the security vulner-

abilities addressed so far [26, 28]. The ISO protocol does provide details which could be exploited by attackers to gain access to information such as waiting time extensions and error recovery [28, 31]. Through waiting time extensions an attacker can continually request additional time that can be used to complete the relay attack. This time can be extended to at least 4.95 seconds. Error bits can be introduced into the communication that will cause the reader to send additional requests to the tag, giving the attacker more time to complete the attack [28]. This error bit technique was shown to only be a weakness depending on the implementation in the reader.

Tag data can be encrypted, excluding the UID, but this only protects the data. If eavesdropping attacks are being performed then encryption seems like a good solution. The relay attack remains a threat as the attacker does not need to understand any of the transmitted data to take full advantage of the transfer. Completely encrypted communication between endpoints is still susceptible to relay attacks [21].

Authentication, privacy protocols, tag blocking, encryption, behavioral analysis, timing, distance-bounding protocols and even hardware to attach to the back of mobile devices have all been proposed by various authors [1, 16, 22, 23, 28, 46]. An extremely important aspect of any solution to the relay attack problem will fit within the existing technology and protocols put forth by the ISO 14443 standard. A solution creating a minimal amount of change to existing devices as well as a low cost for implementation is required.

1.5 Countermeasures against Relay Attacks

A good countermeasure solution against the relay attack addressed here will be an implementation that can work within the operating constraints of the existing ISO and ECMA standards. By providing a small change to the software at the reader or initiator we can essentially provide a solution with a software or firmware update.

This would not create changes to any of the existing tags, which have already been manufactured and distributed. Rather changes would remain limited to the readers, and could create an affordable and easy-to-implement solution to this problem. We do not propose a change to any of the standards, to the communication methods or anything to compromise alternate tag type compatibility, but instead offer a warning mechanism to detect and counter possible relay attempts.

This solution has been implemented and tested with Type 2 Mifare Ultralight A tags. It is functional and consistent. We only used Mifare Ultralight A tags in this project, but if alternate tag types not tested here were to present additional timing artifacts, such as anomalies in the time readings, then at the worst warnings would be generated. The implementation could allow for disabling such warnings if necessary. In a case where those other kinds of cards were seldom seen, the occasional warning may even be acceptable. We would advise against allowing a system to present warnings constantly as users could tend to become complacent and unaware when real problems occurred [20]. If different tag types were to be used additional research into timing for those tags should be evaluated. We need this solution to be robust against attacks and consistent in its warnings. In Android devices for example a quick code update could create the warning and verification system needed to protect against relay attacks and could be pushed out to the devices as any normal software update would be.

2. CURRENT WORK

Some of the more promising work done in relay attack prevention use distance-bounding protocols, which set an upper bound to the distances allowed [16,24]. This is achieved by requiring a cryptographic challenge-response mechanism to measure the proximity to the authentic device. This approach can present challenges, such as time for processing such responses, elimination of time extensions for responses, and dealing with noisy channels. The more error resistance is incorporated into them the more open to attacks these bounding protocols can become [23,24]. Some promising work incorporates a full-duplex secret key agreement scheme into the distance-bounding [29]. This would hopefully help with power consumption and computation demands due to the cost of cryptographic operations. Even when proper encryption techniques are used the protocols still remain vulnerable to relay attacks [28].

Without authentication of the reader an attacker could query the tag early on and replay the information at a later time [30], which is sometimes referred to as *terrorist fraud*. Eavesdropping, or listening in on such conversations, has been identified as a problem. Unless encryption of data is used in some form this remains an issue. The ECMA even state in one of their documents [9] that users should plan implementations carefully around possible vulnerabilities. As encryption is not built into NFC, security is left up to the application developer, or the end user of the technology, which is not a comforting prospect [35]. Even encryption schemes may be ineffective without prior sharing of device authentication [9].

With the relay attack if we could store the data from the tag and send it to the proxy to emulate the tag then detection in this manner becomes impossible. There would be no additional timing delays in such a setup. This is also not accurately

described as a relay attack however. This is more of a true man-in-the-middle or just store and replay attack. This type of security vulnerability has already been seen in attacks such as ticket cloning [7]. Even a replay attack for passport control was discussed and is vulnerable [30], but again, this is a replay, not a relay attack.

The UID, which is supposed to uniquely identify each tag or other NFC communication participant, can not be used in any security protocol, as it can be faked or manipulated. With custom made equipment, attack windows can be extended through waiting-time extensions and bit faults up to very large amounts of time in which to perform attacks [31].

To add to security a user behavioral profile can be created and updated on mobile devices [1]. These profiles monitor a users normal activity. Once created they can compare the current activity with the profile and perform a security risk assessment based on that activity. This could create additional authentication needs, such as requiring a pin to be entered to activate functions considered unusual for that user. This kind of detection and profiling, called active authentication, is still in the early stages of development but shows promise.

Additional prevention techniques also include gesture recognition and position data [45]. These methods use the device location to verify close proximity to the actual reader for security.

Malicious software installed onto devices could even take advantage of the NFC technology, and software-based attacks could be activated, without the victim's knowledge [45]. Google Wallet [18] provides users with a convenient method to store credit card data and uses NFC by only carrying their phones, but software based attacks can create vulnerabilities in programs like this [40]. In addition, these contactless payment methods provide no protection against relay attacks.

There have been more extreme techniques suggested, such as enclosing readers

inside of Faraday cages [21]. This limits all signals from traveling outside of a predefined physical area. Faraday cages could also be used to enclose tags when not in use, but if used on a phone would limit connectivity or create a burden on the user. There is work suggesting attaching such a device to phones to prevent eavesdropping [46]. Devices such as these seem impractical and would most likely not be popular in the market unless they could be included in the phone at the manufacturer. Asking consumers to purchase and attach additional hardware is an unreasonable solution and only addresses eavesdropping, not relay attacks, which would remain a security risk. There have been RFID blocking wallets [42] available for some time, but they are not wide spread.

If efficient analog systems over a close transmittable distance are set up, even timing analysis may not be efficient enough to prevent relay attacks, most certainly not during the anti-collision and initialization phases of tag communication [21]. Such systems would not take advantage of longer-distance communication mediums such as the Internet but still provide viable attack scenarios.

Techniques for formal analysis such as probabilistic model checking can provide insight into complex systems and help find additional security threats [3]. These methods provide a good technique for evaluating the NFC protocols and finding weaknesses.

NFC is being looked at in health care applications as well, and with it come all the same security vulnerabilities [4]. Naturally negative side effects of security threats affecting the application of NFC technology in medical devices or health care applications in general would be devastating. Proposals to secure the use of NFC in this domain include multi-dimensional classifications of NFC devices and suggest evaluation at many different points for determining the systems vulnerabilities. These types of applications in health care will most certainly have to be scrutinized

at many levels as the author argues. The susceptibility of these kinds of devices in regard to the relay attack has not been evaluated yet.

3. IMPLEMENTING A SOLUTION

In the following we present an approach to detect relay attacks by monitoring the communication delays and by responding to timing anomalies. The premise for this work is that (a) we can observe how long it takes to communicate with a tag and (b) if our timing is accurate enough predict the likelihood of whether or not a relay attack is occurring. In order to be feasible, this solution requires an accurate analysis of the consistency of timing between tags. A timing based anomaly detection is only as accurate as the underlying behavior is predictable. Once timing is shown to be consistent we proceed to demonstrate that the implementation of a relay detector is possible at low cost.

In order to collect tag read latency and relay latency data we implemented a relay system. A breakdown of the tag communication steps at a low level will help us with understanding where any timing anomalies occur. We will show that by eliminating all extra time built into this tag communication library the actual delay caused is minimal and then by evaluating that delay over the relay system we can show that it is detectable and preventable.

We implemented a relay system consisting of an Arduino microcontroller to run two NFC PN532 breakout boards, each of which can act as a tag or NFC reader. This setup will provide the fastest possible data transfer between the mole and proxy so any real world relay could only possibly add to the time delays through their connection. We have two Arduino microcontrollers attached separately to PN532 breakout boards, one is a basic tag reader to verify normal communication, the other is a 'relay safe' version which prevents the communication if a relay is detected. We show that both readers will read a basic tag in authentic fashion consistently and

the 'relay safe' reader will show a warning when trying to read the tag through the relay setup.

Various forms of relay attack setups have been built before, for example, an initial relay attack setup was created and tested using an inexpensive analog transfer of data over a short range radio frequency [21]. This method did not buffer any data to send in packets or take advantage of digital communication mediums which reduced the effective ranges and technology available. This work also mentions the delay introduced in the relay was only about $20 \mu s$, but this was during the initialization and anti-collision phase of communication. The timing done for this project will be at the page communication level and can be read for all sections of communication as we will show.

This enabled us to build and test a Arduino controlled PN532 NFC reader that is hardened against page-level relay attacks. Increasingly, smart phone based NFC applications are becoming popular, and solutions must be found to harden smart phones against NFC relay attacks. In a second phase of this project we will look at the Android operating system to include a timing based relay detector into the kernel. This could be included with any distribution or update of Android. The solution will cause the device to disconnect if a relay is detected so as to completely prevent the communication from occurring and prevent any unwanted loss of data. This could be extended to issue a warning and request additional user authorization instead and so allow the device to transmit its contents on a second attempt. In contrast to the microcontroller based system used in the first phase, any timing based detector in an Android device has to deal with latencies on the device. In order to test the robustness of our relay detector we measured its performance on an Android system with a benchmarking tool to verify operational consistency under variously strained real world scenarios.

These solutions are robust for the tags tested, all Mifare Ultralight A, and consistent in behavior. A worst case example for the Android system would include the occasional false positive which is a much better solution than an occasional false negative. This system can be built into existing readers and secondary authentication could be requested in more secure applications. This would not prevent tags communication which took more time, but would add a level of security to misbehaving tags. If applications were less secure, or a decision was made that a relay attack was unlikely or unimportant to the application the feature could just be ignored or turned off. This approach fits into the RFID/NFC standard with a minimal impact to existing technology. It causes absolutely no changes to all the tags out now and being produced in the future.

3.1 Equipment Setup

For this project we used the following setups to read tags and take timing measurements. First, we used an ACR122u contactless smartcard reader/writer connected to a personal computer (PC) over USB. The computer was a model hp workstation xw9300 running Ubuntu version 12.04 LTS operating system. We installed the software from libnfc [33] and any additional support drivers to get the ACR122u working.

Nexus 4 Android cell phone. This has a PN544 chip by NXP Semiconductor inside it which runs the NFC communication. We wrote application level software to do some analysis and later modified the kernel code from Cyanogenmod [8], an open source development of the Android operating system.

The PN532 breakout board is an NFC controller from Adafruit Industries. It is an integrated contactless control board for NFC communications designed to work as both a tag and initiator [38]. It can be hooked up to either a PC through a

USB connection or to other devices over either an I²C (I-squared-C) or SPI (Serial Peripheral Interface) connection (two different interfaces for short distance wired communication).

Arduino is an open-source electronics platform designed to be easy to use for all kinds of interactive projects [6]. We chose this platform because we could wire our PN532 breakout boards to it, have access to the connections easily if we wanted to attach a logic analyzer and because there is no operating system running on it. There is only a main loop which repeats with a specified delay. This became our controller for the PN532 boards and was used to build our readers and our relay system.

We used a Logic16 analyzer from Saleae [34] for looking at the signals transferred between the Arduino and PN532. This unit was self contained and interfaced with a PC to give readings on the signals between devices. The timing and flow of the algorithms were more easily analyzed using the logic analyzer.

3.1.1 Tag Timing

We started off our time measurements looking at how long it took for one page of data to be sent to the initiator after a read request. This means that after we have a connection between the PCD and PICC we would do a series of read requests to transfer all the data on the tag to the initiator. The initiator will issue a read request from the tag and the tag will respond with the data for the page requested. Figure 3.1 shows an example of this. The reads are repeated for as many sections of data as are requested. The host in the figure can be the PC, the Android operating system or the Arduino and the controller, which is either the ACR122u, the PN532 or the PN544 chip inside the Android phone, depending on which setup is being used. The data reads are further broken down in the tag analysis section.

To do an accurate analysis we needed to take timing measurements on different

devices. To do this we needed to verify what devices were giving us accurate time readings.¹

We went on to implement a system using the PN532 connected to an Arduino. While the PN532 could be connected to a PC over a USB connection we would have just encountered the same issue as before with the ACR122u, therefore programming the Arduino to implement the reading was the best approach. The Arduino only runs a simple control loop so we could control all aspects of what was running on the device. We connected the Arduino to the PN532 over the SPI interface and loaded the libraries put out by Adafruit [2] for the PN532 to run the system. While there were many functions to drive the PN532 from the Arduino included, the library was not complete. We completed portions of it to increase functionality but it could still be expanded further. From this setup with the Arduino we again took timing measurements and we saw a minimum time of approximately 50 *ms*, more than the original time from the Android system. With this surprising result we then looked into the library in detail and found there were waiting time extensions built in that were causing excessive delays.

3.1.2 Analyzing Tag Communication

We hooked up the Logic16 analyzer to the PN532 to look at the actual bytes being sent between it and the Arduino. A careful analysis of the signals revealed several time delays in the sequence which were completely unnecessary. Using the measurements from the logic analyzer and carefully looking over the library code we were able to remove almost 60 *ms* of unneeded delays. We started removing them and testing reads again until we broke the read functionality, then built it back up

¹Previous experiments with an Android based reader yielded minimum times of about 40 *ms*, which was due to the measurements being taken at the application level. We also experimented with using the ACR122u, which gave us latencies of about 19 *ms*. These delays were due to the reader being driven by the USB and the measurements returned were USB polling results.

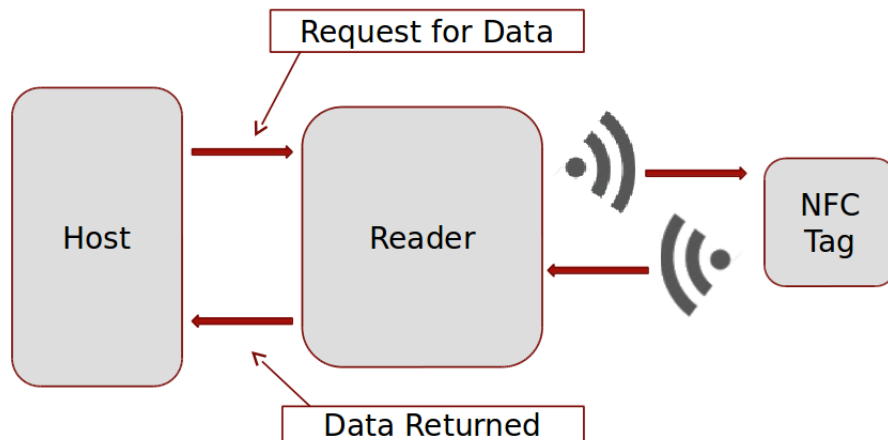


Figure 3.1: Read request from host controller to NFC radio to receive a section of data from the tag (16 bytes at a time - 1 section).

until we had a consistently stable read established. This resulted in a total delay of only 4.7 ms (this was the PN532 to tag communication sequence) and a section read time of 7.294 ms as seen in Figure 3.2. The delay occurs when the Arduino requests the data from the PN532 and that command is transmitted to the tag. It then waits for the 4.7 ms and checks to see if the PN532 is ready to transmit the data back to the Arduino, which means tag communication for that section was complete. If it is not ready to return the data, it waits for another delay period of 4.7 ms . This delay is right on the edge of how long it takes to get the data from the tag, so much so that a small portion of the reads wait for an extra delay which adds another 4.7 ms in this case to the total read of 7.294 ms . An actual solution would increase this wait period to include a safety and guaranteed response time of something closer to 5 ms .

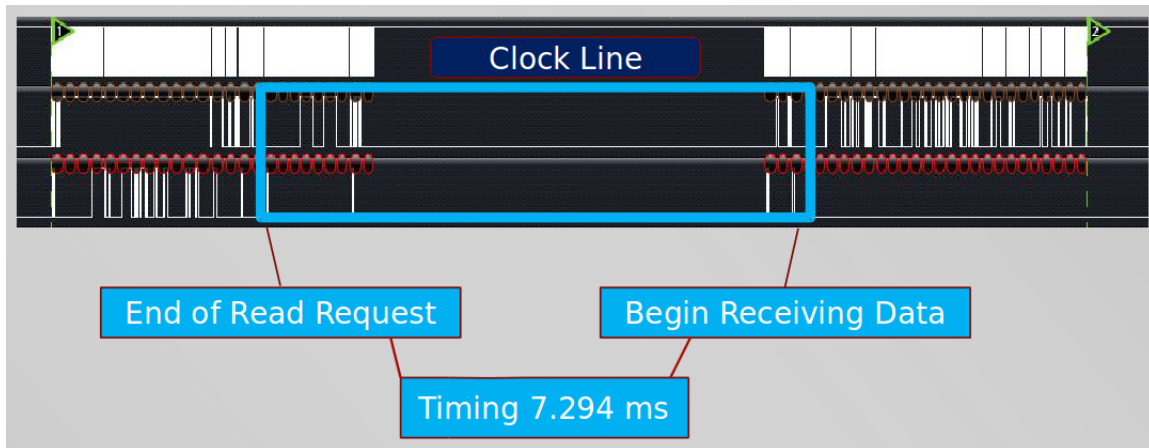


Figure 3.2: Reduced delay on tag time read, timing from logic analyzer.

3.2 Tag Analysis

In order to collect reliable timing data on tag reads we purchased 30 tags from five different vendors to test. These were all Mifare Ultralight type A tags. Because the chips are all currently being produced by NXP Semiconductor, there was no real difference in the chip itself, but the antenna and the medium in which the chips are embedded vary greatly. We purchased paper, plastic, flexible, rigid, square and medium tags with different proportions. We timed communication of data on the tags to verify consistency. Without verification of basic consistency between types of tags a timing analysis would not have been a realistic approach.

The communication of these tags consist of the initialization and anti-collision, then a sequence of data transfers in 4 page sections. A *page* consists of 4 bytes. A tag contains 42 pages of data and can be completely read in 11 data transfers with our device. A data transfer consists of 4 pages at a time which we refer to here as a *section*. This was set up to maintain read compatibility with other types of tags and is standardized for the Mifare Ultralight A tags. The final read will wrap around and return additional data from the beginning of the tag to compensate for

Mifare Ultralight A Tags (NTAG203)

Page/Sector	Blocks				
	0	1	2	3	
0	1 byte	1 byte	1 byte	1 byte	Read occurs in 4 pages (sections) at a time for compatibility issues
1	1 byte	1 byte	1 byte	1 byte	
2	1 byte	1 byte	1 byte	1 byte	
3	1 byte	1 byte	1 byte	1 byte	
4	Note: Read overflows go to the first page (wrap)
5	
6	
7	
8	Total Memory: 168 bytes 11 reads gets 176 bytes, 8 from page 0 again
.	
.	
.	
.	Total r/w memory: 144 bytes
.	
.	
42	

Figure 3.3: Mifare Ultralight A tags page and section setup. One section consists of 4 pages of data, or 16 bytes.

the end pages as 42 divided by 4 does not come out evenly. The bits use either Modified Miller or Manchester encoding for the actual data transfer according to the ISO 14443 specification [26]. Figure 3.3 shows an example of the pages and the sections that make up a data transfer. Each section of data (4 pages) is considered one read and this read is what we are timing. Using an Arduino microcontroller and the PN532 board we took readings on all the tags and compared them looking for large differences. We found deviations of up to 7 μs with basic tag reads over all 30 tags, read 30 times each. The difference between multiple reads can be seen in Figure 3.4. This confirmed that with tags of varying makes the time readings were consistent.

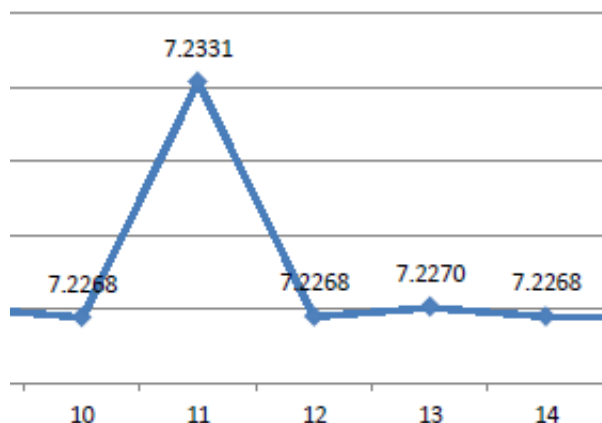


Figure 3.4: Timing of 1 section of a tag read operation, 1 section = 4 pages (x axis). 5 different sections read here. Measurement (y axis) is in milliseconds (*ms*).

3.3 Building a Relay

In order to test the timing of a relay attack we needed a functional relay attack setup. We took the Arduino and hooked it up to two PN532 boards as seen in Figure 3.5. This setup produces the quickest possible page-level data transfer between devices by using the same array for data storage as it is transferred between the mole and proxy. Any setup that implemented an additional communication medium between the mole and proxy, such as a Bluetooth connection, could only hope at best to match the delay we had and at worst add a significant time delay into the relay as we have seen before [28]. The relay system first initializes the mole as a reader and then goes into a loop attempting to connect to a tag. Once a connection is established it checks to see if there is an initializer present. Once those steps occur and an initializer is found, the proxy is initialized as an emulated tag, and the transfer begins. Data is read, and sent, section by section from the authentic tag to the authentic reader. This attack is fast and effective.

The UID of the tag is not emulated in this setup. The PN532 is designed so

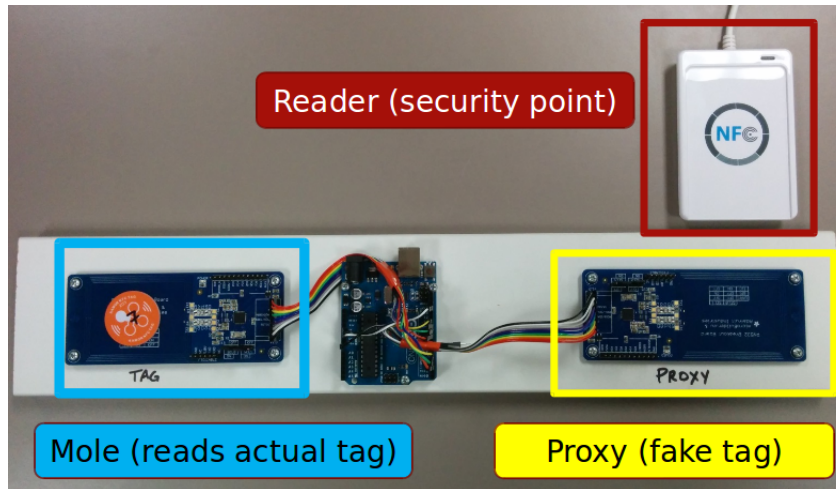


Figure 3.5: Arduino and two PN532 boards used to create our NFC relay system.

that it can not emulate an authentic tag UID, which is 7 bytes long. The emulated tag UID is only 4 bytes from the PN532. This is not a problem with the relay concept as it has already been established that the UID is inherently unsafe in nature, being transparent and able to be emulated by certain devices [7]. This separate initialization on either end adds to the relay speed as the tag is initialized before any communication occurs and never has to transfer that data for any initialization or anti-collision steps. Any of those steps are already taken care of by the mole and the proxy on their respective ends. In other relay setups all initialization can be passed through to either endpoint and encryption standards do not prevent the attack [24].

3.4 Preventing the Relay Attack

Once we had an effective system set up which consistently relayed tags we evaluated the timing of the relay. Hooking the logic analyzer up again to the system and taking readings on all the PN532 boards showed us there was an additional time delay added between the transfers we could take advantage of to prevent these attacks. We wanted a reader that could warn us if the expected delays were too high.

The results of the logic analyzer readings are shown in Figure 3.6. The minimal time for communication we recorded on the relay was 52.765 *ms*. This was a significant difference we could use.

We have some additional delays occurring in the boards, specifically between the authentic reader (security point) and the proxy end of the relay. We did not find the cause for this extra delay but had we eliminated it there remained a large window of time to work with. If we did eliminate the extra delays we would still have a relay attack window of 32.357 *ms*. With these extra delays in our fastest system we can be lenient with where we implement our timing analysis to prevent the attack. Given a 7.294 *ms* delay in an authentic transmission we can look at anything over 10 *ms* as questionable, or if we want to be more lenient we could even double that time to 15 *ms*.

We used the collected timing information to build and test a 'relay safe' reader. For this we used an Arduino attached to a PN532. Loading the same code as we previously did on the original reader we added a timer constraint on each communication block. If any given transmission exceeded 4.7 *ms* we stopped communication, issued failures on the remainder of the sections to be read and noted the attempt. After a predefined number of attempts we blocked the reader from all communication and issued an on screen warning that too many failures had occurred and a relay attack was likely. This demonstrated the capability of the 'relay safe' device to prevent the relay attack on our system. If the user was not malicious and attempted once or failed for another unanticipated reason the next success would reset the relay attempts and no lockout would be issued. If they were maliciously attempting a relay, a warning would alert someone at the reader and cause some form of security to be alerted.

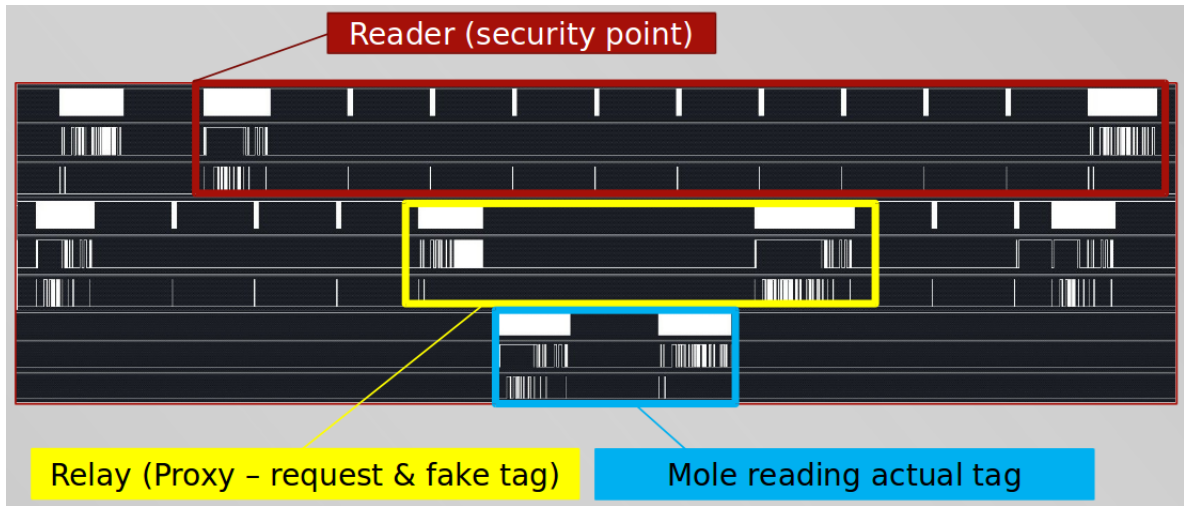


Figure 3.6: Timing analysis of relay attack using logic analyzer. Three different PN532 boards are pictured here, top: security point, middle: proxy on relay, bottom: mole on relay.

3.4.1 Android NFC

After preventing the relay attack through timing with the PN532 we moved back to the Android operating system again. We had previously seen large delays through our application but needed to determine if we could do a lower level timing analysis to prevent an attack in this type of system as well. We downloaded and evaluated code both from Google directly and from Cyanogenmod. Cyanogenmod is an open source community driven project to take the source code for Android from Google and adapt it to be usable on a wide range of devices [8]. This is necessary because the source code for Android put forth by Google only works in the original state for Google devices. To run the Android operating system on other devices there are changes that must frequently be made by the company putting out the phone, such as custom device drivers to run specific hardware not originally added into the source code by Google. Cyanogenmod became a good choice of software to examine

```

/** Internal transceive */
byte[] transceive(byte[] data, boolean raw) throws IOException {
    checkConnected();
    long stime, etime, resultTime;
    try {
        stime = System.nanoTime();
        TransceiveResult result = mTag.getTagService().transceive
            (mTag.getServiceHandle(), data, raw);
        etime = System.nanoTime();
        resultTime = (etime-stime)/1000;
        Log.v(TAG,"Tranceive time: " + resultTime);
        // If time is exceeded, stop communication by closing tag
        // - relay attack prevention
        if (resultTime >= 10000) {
            Log.e(TAG,"NFC Transceive exception - timing limit exceeded!");
            mTag.getTagService().close(mTag.getServiceHandle());
            throw new IOException("I/O timing exception - possible relay attack!");
        }
        if (result == null) {
            throw new IOException("transceive failed");
        } else {
            return result.getResponseOrThrow();
        }
    } catch (RemoteException e) {
        Log.e(TAG, "NFC service dead", e);
        throw new IOException("NFC service died");
    }
}

```

Figure 3.7: Code change in the *BasicTagTechnology.java* file to time the sections.

as changes to this code would be compatible with a much larger range of devices and there exists the possibility of actually adding in the relay prevention code to this kind of project if it proved functional and useful.

We added latency measurements to the function *transceive()* located inside the *BasicTagTechnology.java*. This function sits at the Framework layer, which is implemented just above the Hardware Abstraction Layer (HAL). Figure 3.7 shows the change in the java file and Figure 3.9 shows visually where the timing is implemented within the architecture.

This was implemented in Cyanogenmod version 3.4.0 using Android version 4.4.4 Mako and the kernel was built with the change seen in Figure 3.7. We used 10 *ms* as a timing cut-off for the relay detector, allowing for extra time for robustness. If

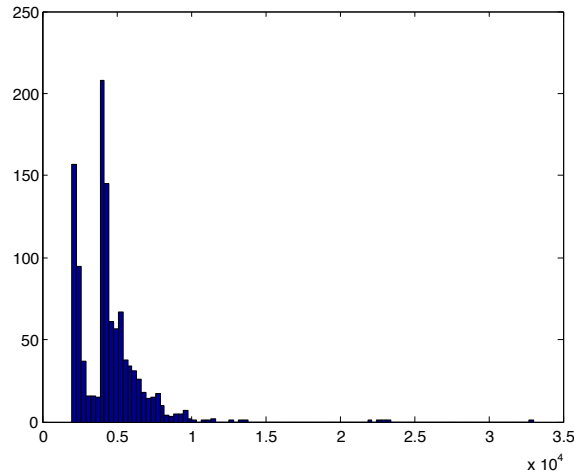


Figure 3.8: Histogram plot of tag read times in Android. The majority of reads are at about 4.5 ms or less. Y axis is sections read over 62 tags, X axis is time for read on each section.

this time was exceeded then the Android operating system would disconnect the tag at that point, not allowing for any further communication to occur. This created a system where the Android would not just provide a warning and still read all the data, but if a warning was issued the data would not be transferred on that attempt. It was important to stop this data transfer in addition to a warning for an acceptable security system. If we provide a warning only and still allow the data to go through then the attack would succeed before any possible intervention.

Taking timing measurements from the Android device at this point resulted in tag reads of about 4.5 ms per section. A histogram plot of one tag read is shown in Figure 3.8. This showed that the timing at this level in the Android was even tighter than the timing measurements received from the custom readers we created. We read over 60 tags to get this time result.

Stressing the system to verify the timing on the Android system was then done.

We loaded the AnTuTu [5] benchmarking software onto the Nexus 4. We ran the suite of benchmark tests on the device and then started taking time measurements again. Reading the tags provided additional failures than it had previously. Out of 62 tag reads originally with no stress system running on the phone we saw at least 13 time readings which would be over the 10 *ms* threshold and give a timing exceeded failure. With 47 reads while running the benchmark for added stress we saw 15 failures and again with 48 reads saw 14 failures from the timing being exceeded. This is only an approximate 10% difference. It should be noted that the benchmark would cut out short of completion due to graphics being initialized and pulling the phone out of the NFC reading application. This should not be a significant issue though as users would most likely not be running graphics programs and using the NFC at the same time.

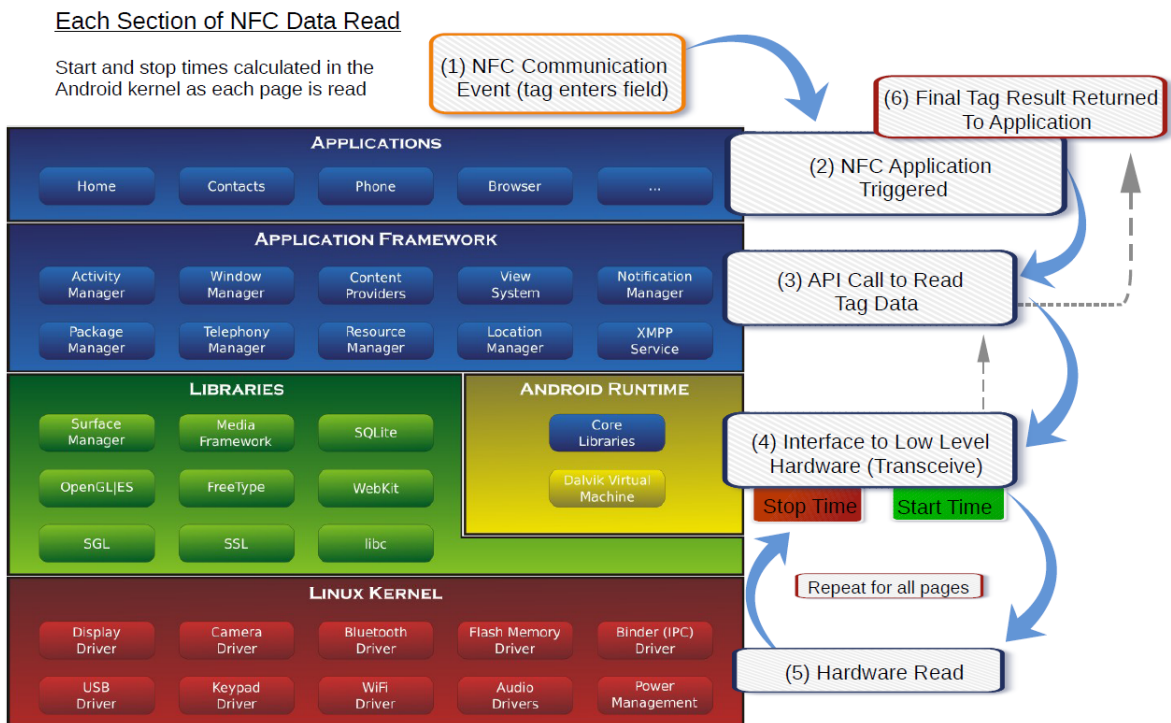


Figure 3.9: Android Architecture [44] with timing shown. Timing starts before HAL layer which sits between the kernel and library layer.

4. EVALUATION

After evaluating all the tags over the relay system with the normal reader setup, one of the Arduino to PN532 setups, we found that they all communicated effectively and consistently through the relay. The second Arduino to PN532 reader was created which had timing constraints preventing communication if the time was exceeded and a possible relay was found. This was our 'relay safe' unit. We verified this 'relay safe' unit effectively and consistently read data from all 30 tags as the first unit would under normal direct tag to reader mode. After verifying this we tested the 'relay safe' unit with all tags over the relay. This unit consistently detected the additional communication time and stopped the read, issuing warnings to begin with then freezing the reader completely after five consecutive relay attempts in a row. At that point, with our algorithm, a reset of the 'relay safe' device was required to continue to function as per our program.

This setup gives us the ability to detect a relay attempt and alert others to the attempt in the process. If only one read attempt is invalid then no alert is given and no additional action is needed. The stop functionality could of course be changed to any number of attempts desired based on the level of security wanted at the reader.

We were not able to test the relay safe reads against a Android because the Android system has a failure to communicate with the relay system we set up. This could be due to several factors. The Nexus 4 has a Broadcom Corporation chip instead of an NXP chip so the make is a little different. The Adafruit site also talks about the PN532 having an inability to communicate with the Android both due to incomplete libraries to implement application level communication and due to its inability to emulate a tag as it has no secure element. The secure element

is an additional piece of hardware in mobile devices to separate the data from the communication element providing an additional internal security device. This is not particularly interesting for the purposes of a relay attack and we have not looked into any secure elements. This also means that the custom relay we set up only reads on the PN532 readers, and not on the Android. Testing the timing over the relay for the Android for this reason did not succeed. There is speculation that the timing constraints at the hardware level are much tighter on the Android system as well. While we can attempt to emulate a Mifare Ultralight A tag and the Nexus 4 seems to detect the tag, there is never any additional information transmitted with it and a consistent error message that the tag was removed from the field before the read was completed. Unfortunately, without additional hardware at this time, we do not believe we can truly emulate a tag detectable by the Android Nexus 4 for the relay setup.

The benchmark for the Android system while running the NFC reads were only one way of stressing the system while measuring tag reads, and perhaps not the best. Because of how the NFC is processed in the Android we can not just create a loop to repeatedly read a tag and then stress the system which would be a more ideal approach. Running the benchmark and then logging all tag reading results while it was running was the approach we took, but this does not give us a good idea exactly what process was being stressed that changed our read timing. It does give us a good idea of how other processes might effect real world performance during reads however.

The results of our timing analysis on the Android Nexus 4 can be seen in Table 4.1 and Table 4.2. As you can see the timing analysis gives fairly close results while stressing the system with the benchmark compared to reading with no additional programs running. A tag will fail to read if a single section exceeds the 10 *ms*

	Entire Tag Read		
Benchmark	Total Reads	Failures	Failure %
AnTuTu	95	29	30.5
None	62	13	21.0

Table 4.1: Complete Tag Reads. Tag failure if one or more section takes over 10 *ms* to read.

window.

In Table 4.2 you can see the timing results of each section during the complete tag read, if a failure occurs the remaining sections are not attempted and a new read is started.

	Average Time (<i>ms</i>) Per Section	Sections Read	Average Time (<i>ms</i>) Per Section	Sections Read
Success	4.263	1306	4.335	1102
Failure	40933	29	16335	14

(a) With AnTuTu running.

(b) No additional programs.

Table 4.2: Timing results per section with and without AnTuTu benchmark running. Average time of a successful or failed read.

It can also be noted that only a portion of the reads which failed during the benchmarks, about 37%, failed on the first few sections of the reads (first 3 sections). The remainder failed later on during reads on that tag. One may get better performance by limiting the timing to the first few pages or even to a section of the tags randomly. Possibly taking a sample section for time readings would accomplish this. The effect on the safety of the relay prevention should be evaluated if this was attempted and we present no predictions on that in this work. The timing analysis done here however is more than adequate for relay attack prevention.

5. CONCLUSION

NFC is currently a rapidly expanding technology. It surrounds us today and will continue to in the near future. We have all seen the effects of malicious parties taking advantage of security vulnerabilities in technology fields, and NFC is no exception to this. The rapid expansion of the technology and lack of solutions to the security vulnerabilities are a concern. Relay attacks are a very real and present threat to this technology, both difficult to detect and to stop.

We show in this work that by timing the communication of NFC tags a section at a time during data transmissions we can detect relay attacks successfully. Accurate timing at the right level is critical in this endeavor. After detection, many different actions can be taken. One such action could be a system that first issues warnings, then completely stops communication.

By looking at the time it takes for a data read on a tag we can preserve the existing technologies by adhering to the ISO specifications and enhance security at the same time. This will not cause major changes to the protocols or the existing devices on the market today. In the worst case this solution would simply warn an end user of the possibility of a relay attack, thereby allowing all current NFC devices to operate under the established protocols and simply alert them of possible malicious activity. Implementing a solution such as this to fit within the current ISO specifications without causing major changes or expense is important and this solution fits that criteria.

This technology benefits many industries and allows for applications in fields which could be extremely advantageous. We expect that the solutions presented here will add both a level of security as well as value to NFC technology.

Future work will include an analysis of other types of tags beyond the Mifare Ultralight A. Additional relay attacks can be created using faster analog systems to relay data at the bit level instead by sections. Providing a relay setup to do further testing against the Android relay prevention code and finding additional constraints with speed will increase our ability to prevent relay attacks.

REFERENCES

- [1] D. Abu-Saymeh, D.E.-D.I. Abou-Tair, and A. Zmily. An application security framework for near field communication. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*, pages 396–403, July 2013.
- [2] Adafruit.com. Adafruit. <https://learn.adafruit.com/adafruit-pn532-rfid-nfc>. Accessed: 2014-5-19.
- [3] N. Alexiou, S. Basagiannis, and S. Petridou. Security analysis of nfc relay attacks using probabilistic model checking. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International*, pages 524–529, Aug 2014.
- [4] A. Alzahrani, A. Alqhtani, H. Elmiligi, F. Gebali, and M.S. Yasein. Nfc security analysis and vulnerabilities in healthcare applications. In *Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on*, pages 302–305, Aug 2013.
- [5] AnTuTu Hong Kong. Antutu.com. <http://www.antutu.com/en/index.shtml>. Accessed: 2015-4-1.
- [6] Arduino. Arduino.cc. <http://www.arduino.cc/>. Accessed: 2014-5-19.
- [7] C. Chen, I. Lin, and C. Yang. Nfc attacks analysis and survey. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2014 Eighth International Conference on*, pages 458–462, July 2014.
- [8] Cyanogenmod. Cyanogenmod. <http://www.cyanogenmod.org/>. Accessed: 2015-01-10.

- [9] ECMA. *ECMA: NFC-SEC, NFCIP-1 Security Services and Protocol Cryptography Standard using ECDH and AES (White Paper)*. ECMA International (European Association for Standardizing Information and Communication Systems), Geneva, Switzerland, dec 2008. 2008.
- [10] ECMA. *ECMA-386: NFC-SEC-01: NFC-SEC Cryptography Standard using ECDH and AES*. ECMA International (European Association for Standardizing Information and Communication Systems), Geneva, Switzerland, jun 2010. 2010.
- [11] ECMA. *ECMA-386: Near Field Communication Wired Interface (NFC-WI)*. ECMA International (European Association for Standardizing Information and Communication Systems), Geneva, Switzerland, jun 2012. 2012.
- [12] ECMA. *ECMA-340: Near Field Communication - Interface and Protocol (NFCIP-1)*. ECMA International (European Association for Standardizing Information and Communication Systems), Geneva, Switzerland, jun 2013. 2013.
- [13] ECMA. *ECMA-352: Near Field Communication Interface and Protocol -2 (NFCIP-2)*. ECMA International (European Association for Standardizing Information and Communication Systems), Geneva, Switzerland, jun 2013. 2013.
- [14] ECMA. *ECMA-385: NFC-SEC: NFCIP-1 Security Services and Protocol*. ECMA International (European Association for Standardizing Information and Communication Systems), Geneva, Switzerland, jun 2013. 2013.
- [15] Ecma-international.org. Ecma international. <http://www.ecma-international.org>.
- [16] A. Falahati and H. Jannati. Application of distance bounding protocols with random challenges over rfid noisy communication systems. In *Wireless Sensor*

- Systems (WSS 2012), IET Conference on*, pages 1–5, June 2012.
- [17] NFC Forum. Nfc forum. <http://www.nfc-forum.org>, 2015. Accessed: 2013-06-01.
- [18] Google.com. Google wallet. <http://www.google.com/wallet>. Accessed: 2015-6-9.
- [19] J.M. Graefe, L. Lashley, M.A.M. Guimaraes, E. Guodabia, A.K. Gupta, D. Henry, and R. Austin. Credit card transaction security. In *Proceedings of the 4th Annual Conference on Information Security Curriculum Development, InfoSecCD '07*, pages 18:1–18:6, New York, NY, USA, 2007. ACM.
- [20] M. Grissinger. Warning! don't miss important computer alerts. *P and T*, 35(7):368+372, 2010.
- [21] G.P. Hancke. A practical relay attack on iso 14443 proximity cards. Technical report, University of Cambridge Computer Laboratory, 2005.
- [22] G.P. Hancke. Practical attacks on proximity identification systems. In *Security and Privacy, 2006 IEEE Symposium on*, pages 6 pp.–333, May 2006.
- [23] G.P. Hancke. Distance-bounding for rfid: Effectiveness of 'terrorist fraud' in the presence of bit errors. In *RFID-Technologies and Applications (RFID-TA), 2012 IEEE International Conference on*, pages 91–96, Nov 2012.
- [24] G.P. Hancke, K.E. Mayes, and K. Markantonakis. Confidence in smart token proximity: Relay attacks revisited. *Computers & Security*, 28(7):615 – 627, 2009.
- [25] E. Haselsteiner and K. Breitfu. Security in near field communication (nfc). strengths and weaknesses. In *In Workshop on RFID Security RFIDSec*, 2006.

- [26] ISO 14443. Identification cards Contactless integrated circuit(s) cards Proximity cards, 2000.
- [27] ISO.org. Iso. <http://www.iso.org/iso/home.htm>.
- [28] W. Issovits and M. Hutter. Weaknesses of the iso/iec 14443 protocol regarding relay attacks. In *RFID-Technologies and Applications (RFID-TA), 2011 IEEE International Conference on*, pages 335–342, Sept 2011.
- [29] R. Jin, X. Du, Z. Deng, K. Zeng, and J. Xu. Practical secret key agreement for full-duplex near field communications. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14*, pages 217–228, New York, NY, USA, 2014. ACM.
- [30] Z. Kfir and A. Wool. Picking virtual pockets using relay attacks on contactless smartcard. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 47–58, Sept 2005.
- [31] T. Korak and M. Hutter. On the power of active relay attacks using custom-made proxies. In *RFID (IEEE RFID), 2014 IEEE International Conference on*, pages 126–133, April 2014.
- [32] libnfc.org. Proxmark.org, a radio frequency identification tool. <http://www.proxmark.org>. Accessed: 2014-11-20.
- [33] libnfc.org. Libnfc:nfc-relay. <http://nfc-tools.org/index.php?title=Libnfc:nfc-relay>, December 2012.
- [34] Saleae LLC. Saleae: The logic analyzer with analog. <https://www.saleae.com/>. Accessed: 2014-5-1.

- [35] D. Nelson, M. Qiao, and A. Carpenter. Security of the near field communication protocol: An overview. *J. Comput. Sci. Coll.*, 29(2):94–104, December 2013.
- [36] nfc.cc. Near field communication and mobile technology provided by professionals. <http://www.nfc.cc>. Accessed: 2015-5-20.
- [37] D.R. Novotny, J.R. Guerrieri, M. Francis, and K. Remley. Hf rfid electromagnetic emissions and performance. In *Electromagnetic Compatibility, 2008. EMC 2008. IEEE International Symposium on*, pages 1–7, Aug 2008.
- [38] NXP. Um0701-02 pn532 user manual rev. 02. http://www.nxp.com/documents/user_manual/141520.pdf. Accessed: 2014-5-19.
- [39] F. Pfeiffer, K. Finkenzeller, and E. Biebl. Theoretical limits of iso/iec 14443 type a rfid eavesdropping attacks. In *Smart Objects, Systems and Technologies (SmartSysTech), Proceedings of 2012 European Conference on*, pages 1–9, June 2012.
- [40] M. Roland, J. Langer, and J. Scharinger. Applying relay attacks to google wallet. In *Near Field Communication (NFC), 2013 5th International Workshop on*, pages 1–6, Feb 2013.
- [41] RyscCorp. Rysccorp: Proxmark3. <http://www.proxmark3.com>. Accessed: 2015-3-15.
- [42] ThinkGeek.com. Thinkgeek.com. <http://www.thinkgeek.com/product/8cdd/>. Accessed: 2015-6-10.
- [43] R. Tiwari. Nfc transaction market (47%) and nfc enabled handsets (55%) 2019 growth forecasts in new global research reports [press release]. <http://www.prnewswire.com/news-releases/nfc-transaction-market-47-and-nfc-enabled-handsets-55-2019-growth-forecasts-in->

new-global-research-reports-285591571.html, dec 2014. Accessed: 2015-6-6.

- [44] Smieh - Anatomy Physiology of an Android. Licensed under CC BY-SA 3.0 via Wikimedia Commons. Android-system-architecture. <http://en.wikipedia.org/wiki/File:Android-System-Architecture.svg>. Accessed: 2015-3-18.
- [45] Z. Wang, Z. Xu, W. Xin, and Z. Chen. Implementation and analysis of a practical nfc relay attack example. In *Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2012 Second International Conference on*, pages 143–146, Dec 2012.
- [46] R. Zhou and G. Xing. nshield: A noninvasive nfc security system for mobile devices. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '14*, pages 95–108, New York, NY, USA, 2014. ACM.

APPENDIX A

VERIFICATION OF TAG TIME MEASUREMENTS

The thirty tags we acquired from five different companies were as significantly different as we could find, that were still all Mifare Ultralight A. All tags were in groups of five. Some were large stickers, square, round, plastic, paper and key chains. Due to constrains in the timing on the testing system we occasionally see an additional 5 *ms* increase on the time resulting in a 12.2 *ms* response instead of the normal 7.2 *ms*. We will address the reason for that difference in Appendix B. The larger response at 12.2 *ms* all had 0.000 *ms* differences between those amounts. While we bought thirty tags for testing, be the time we started taking readings, two of the tags had failed and only twenty eight were included in the results. The two tags that failed were from different places (number 16 and 24), not from the same batch of five.

Table A.1 shows the significant differences in tag timing measurements reading all data one section at a time. All time measurements were taken using the Arduino *micros()* function on an Arduino Uno 16 MHz [6].

Tags	Minimum	Maximum	Difference
1, 3, 5-6, 8-10, 13-15, 18-23, 26-30	7.227	7.233	0.006
2, 4, 7, 11-12, 17, 25	7.227	7.227	0.000

Table A.1: Tag timing using *micros()* function on Arduino. All times are in milliseconds (*ms*).

The data shows that the most significant difference we have is about 6 μs , and

some even smaller. The tag reading times are therefore deterministic to allow for an accurate timing-based relay detector.

APPENDIX B

REDUCING THE TIMING IN ARDUINO CODE

The code on the Arduino was originally taken from Adafruit [2]. When first hooking up the PN532 to the Arduino we found the time for the read was excessive, about 67 *ms* total. We attached the Logic16 analyzer to get a better understanding of where all that extra time was occurring and found many delays spread through the communication. Matching these timings up with the code we tracked down the source of each delay. They were caused by calls to the *delay()* function at various places. These delays over the course of the entire tag communication added up to the large 67 *ms* time we had seen.

In order to get the most accurate and quickest timing analysis we eliminated all of these delays. These delays were in place to give the PN532 a chance to respond to requests from the Arduino and without them everything completed much more quickly. There were several delays we could not avoid, but we did reduce them. By removing them all we broke the communication, so we then built it back up until it worked consistently. The only delays we could not remove completely ended up being a delay waiting for the PN532 to send data after a request for data had gone out to the device. This delay ended up being 4.795 *ms* which allowed almost all sections to communicate. Occasionally, and inconsistently, the PN532 would not be ready to respond when asked by the Arduino, in which case the Arduino would give another delay of the 4.795 *ms* to the PN532 to respond. This caused an infrequent spike in timing measurements by just over that amount. These higher reading always returned the same time (12.251 *ms*) with no differences to the microsecond. These were considered an anomaly due to tightening the constraints so much and not used

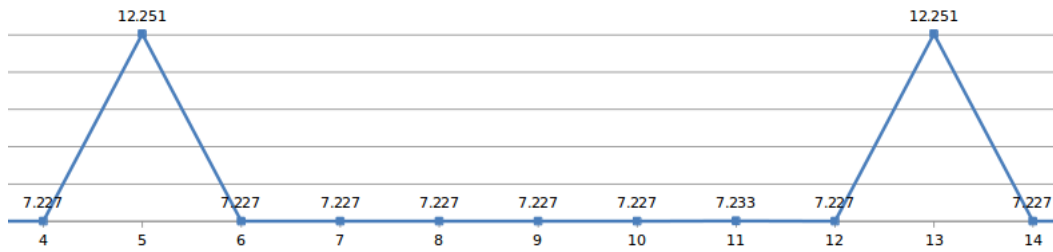


Figure B.1: Graph showing the timing results of sections being read with the extra 5 *ms* jump. X axis is section being read, Y axis is time in milliseconds.

in the analysis of the differences in tag reads. You can see an example of the original graph we created from the timing analysis with the 12.251 *ms* jump in Figure B.1. This is the same graph from earlier where we show the 6 μ s delay jump, but extended to show the additional time difference.