Channel Based Relay Attack Detection Protocol

by

Radi Abubaker

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Masters of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2019

Abstract

A relay attack is a potentially devastating form of a man-in-the-middle attack, that can circumvent any challenge-response authentication protcol. A relay attack also has no known cryptographic solution. This thesis proposes the usage of reciprocal channel state information in a wireless system to detect the presence of a relay attack. Through the usage of an open source channel state information tool, a challenge-response authentication Channel Based Relay Attack Detection Protocol is designed and implemented using IEEE 802.11n (WiFi) in detail. The proposed protocol adapts ideas from solutions to other problems, to create a novel solution to the relay attack problem. Preliminary results are done to show the practicality of using channel state information for randomness extraction. As well, two novel attacks are proposed that could be used to defeat the protocol and other similar protocols. To handle these attacks, two modifications are given that only work with the Channel Based Relay Attack Detection Protocol.

Acknowledgements

I would like to thank my family for supporting me throughout my life and education. I would also like to thank my supervisor, Prof. Guang Gong, both for providing me with the opportunity to experience academia, and for providing me guidance throughout my degree. Thanks also needs to be given to the members of the ComSec Lab at the University of Waterloo for providing help over the past 2 years. Great appreciation is also given to the readers, Prof. Patrick Mitran and Prof. Catherine Gebotys, for volunteering their own time to evalue my thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Since the turn of the twentieth century, wireless communications has increasingly become pervasive in the lives of almost every human. The GSM Association reports that 66% of all people on the planet have a unique mobile phone subscription, with continuing growth projected [1]. Radio Frequency Identification (RFID) is already ubiquitous in many areas of our lives. Beyond established technology, the burgeoning field of the Internet of Things (IoT) will see the addition of low powered computation and wireless communication capabilities in almost all facets of our lives. From thermostats, security cameras, door locks, and even fridges, these colloquially titled smart devices will be communicating primarily by wireless means. Ensuring that these means of communications are secure is paramount to their abilities to function seamlessly in our everyday lives. Without this, confidence in these means of communications for handling delicate information will not be possible. Secure wireless communications should assure confidentiality, authenticity, identity, and accessibility through the use of cryptographic primitives and communication protocols.

Unfortunately, due to the nature of wireless communications being broadcasted on a channel, the vulnerability of these modes of communications makes it difficult to rely on typical cryptographic methods to provide security. One such vulnerability is known as a relay attack. A relay attack is a passive man-in-the-middle attack that immediately transfers messages between legitimate communicating parties via malicious parties. A relay attack can be thought of as a simple range extension or repeater with the intent of breaking authentication without proximity. Typical cryptographic methods to provide authentication and identification are easily overcome as the legitimate parties believe they are talking to one another via unaltered messages. While the methods of attack is innocuous, the repercussions are grand. On September 24 2017 in Solihull, England two thieves were recorded performing a relay attack to unlock a new Mercedes-Benz in the car owner's parking lot. Researchers in Switzerland's ETH have shown that a relay attack can be used to unlock and start many high-end automobiles [2] . Relay attacks can also be utilized to abuse RFID systems, like tap-to-pay payment systems and key-less entry systems.

To deal with this extraordinary form of attack, the paradigms utilized to provide security to typical communications systems cannot be used. When we consider the Open System Interconnection (OSI) model that is used to define communications, we find that most cryptographic solutions appear in the upper layers of communications. Protocols such as Transport Layer Security (TLS), Internet Protocol Security (IPSec), and WiFi Protected Access (WPA), are some of the most widespread cryptographic protocols, and they all exist in the upper layers of the OSI model. The lowest layer in the OSI model is the physical layer and it sees few security features, especially for non-military purposes. The paradigm for the physical layer has been only to handle the transmitting and receiving of upper layer data on to the physical communication channel.

The physical layer typically also has to provide capabilities to deal with channel state effects. Additive white Gaussian noise (AWGN), multipath fading, path-loss, shadowing, and signal interference are all effects that exist on the channel that negatively impact the ability of communicating parties to receive coherent messages at high data rates. While error detection and correction is typically done in the upper layers of communication, the direct combating of these effects takes place on the physical layer. Taking advantage of these statistical characteristics of the channel can potentially provide us with a solution to many of these cryptographic problems that upper layer solutions might not be able to deal with. In 1975, *AD. Wyner* had first shown that under certain conditions, information theoretic security was achievable by taking advantage of channel capacity [4]. The use of channel effects to provide secret key agreement, device authentication, and message encryption have been shown as realistic physical-layer based solutions. Taking advantage of these effects, in combination with typical cryptographic primitives, can be utilized to provide a protocol that can potentially defeat relay attacks.

The Channel Based Relay Attack Detection Protocol is a proposed solution to relay attacks. The protocol utilizes the unique channel state that exists between reciprocal channels to provide authentication against relay attacks. The protocol aims to guarantee that two parties that have pre-shared secret keys are directly communicating between themselves and not via a man-in-the-middle. The protocol is an extension of the typical challenge-response authentication protocols used in many communication security protocols. The channel based relay attack detection protocol allows passive usage of symmetric-key based challenge-response authentication, without the worry of a relay attack. As well, the identification of two new attacks on other physical-layer systems are discussed, with some unique solutions proposed.

The contributions of this thesis are:

1. The design of the Channel Based Relay Attack Detection Protocol that uses measured channel state information to derive a quantized channel. The quantized channel will allow communicating parties to tell if a relay attack is occurring.
2. A design for the preliminary implementation of the Channel Based Relay Attack Detection Protocol on top of the link layer IEEE 802.11 protocol (WiFi),
3. Preliminary analysis of the randomness properties of the Channel Based Relay Attack Detection Protocol,
4. The proposal of a Forged Channel Attack and Analog Relay attack, and the discussion of possible preventions.

This work is organized into six chapters: Chapter 2 covers the background information in wireless communications and other related works in research; Chapter 3 gives an overview of the design; Chapter 4 deals with the set-up of the protocol in software and hardware (Section 4.1), and a preliminary implementation of the channel based relay attack detection protocol (Section 4.2); Chapter 5 looks at preliminary experimental results; Chapter 6 identifies the two new attacks, and their possible solutions; Chapter 7 ends with a discussion of the conclusion and future works. The background material discussed in Chapter 2 begins with a look at the relay attack and how it is able to circumvent upper layer cryptographic methods, and what the repercussions of this are (Section 2.1). An overview is done on the communication channel, including statistical multipath models for channels using real baseband signals, and OFDM systems (Section 2.2). An in-depth look into IEEE 802.11n at the physical layer is done to understand how it can be used for implementing the channel based relay attack detection protocol. Related works in the field of physical layer security, as well as other solutions to the relay attack are reviewed (Section 2.4). Chapter 3 will look at the design for the protocol as an adaptation of the challenge-response authentication protocol. Each of the six major steps in the protocol will be explained (Section 3.1). The chapter will also look at proposed performance metrics for the protocol (Section 3.2). Chapter 4 will take a closer look at the open source tools and libraries used to implement the protocol in a specific instance. This includes: an open source 802.11 CSI Tool, Loss of Radio Connectivity (LORCON), Packet Capture (PCAP), and Libgcrypt (Section 4.1). The chapter will also look at how these packages can be used to implement the protocol (Section 4.2). Chapter 5 focuses on the analysis of the preliminary protocol in a quantitative observation. Some tests from the National Institute of Standards and Technology (NIST) randomness suite are used to generate preliminary randomness results (Section 5.1). In Chapter 6, two potential attacks of different feasibility are discussed, as well as possible solutions. Chapter 7 will end with a conclusion of the thesis, and possible future work for expanding the prevention of relay attacks into other communication protocols and robust experimentation.

Chapter 2

Background and Related Works

2.1 The Relay Attack

To understand the relay attack, we must take a look at the generalized man-in-the-middle attack (MITM). The MITM attack is a form of threat, where an attacker exists in the middle of the communication link between legitimate communicating parties. Typically, the attacker's goal in a MITM attack is to immitate or convince each party that they are the other and intercept messages between them before relaying the message to the intended recipient. MITM attacks can also be used to setup the communications between legitimate parties in an advantageous way to the attacker, that makes the protocol no longer semantically secure or breaking security. We can see a model of the generalized MITM attack in Figure 2.1. The attacker can have two different roles: an active role, known as an active MITM attack, or a passive role, known as a passive MITM attack.

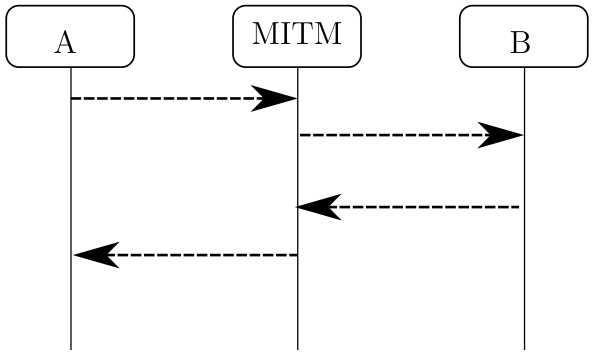


Figure 2.1: A man-in-the-middle attack

In the active scenario the attacker can forge, edit, delay and block messages being sent between the legitimate parties. Active MITM attacks are popular avenues for discovering vulnerabilities in security protocols that make them unsecure. In literature, the lack authentication of cryptographic key agreement is often given as an example of a protocol vulnerable to an active MITM attack. If either communicating party is not authenticated, a MITM could establish fraudulent keys with each party, and forward intercepted messages after decrypting and re-encrypting the messages. Attacks of this type are common on the internet with pages that do not utilize secure socket layer (SSL) or transport layer security (TLS). Even then, without proper certification of a user's identity, attacks can still occur. Typically, active MITM attacks that target cryptographic protocols can be prevented through the proper use of authentication, nonce generation, and the non-reuse of critical components (such as nonces).

In contrast to the active attacker, a passive MITM attacker can simply relays messages between the communicating parties. Before relaying, they can attempt to read the messages, which is referred to as "snooping". If data is encrypted and authenticated, communications remain confidential against passive relay attacks; messages cannot be read, if a proper protocol is used. In a cryptographic context, passive MITM attacks are innocuous.

The relay attack in early literature is a form of passive MITM attack that acts as a signal repeater. The attack's first description in literature was by John Conway in his book *On Number and Games* [6] . In the book Conway describes what has been called the *Chess Grandmaster Problem.* The problem describes a young girl playing simultaneous chess games against legendary chess Grandmasters Bobby Fischer and Boris Spassky. Against Spassky she is playing white and against Fischer she is black. Once Fischer makes the first move, the girl will repeat that move to Spassky and vice-versa. Of course, the girl will guarantee a win against one of the two Grandmasters as they are effectively playing each other. This example of a relay attack has many variants, such as a *Mafia Attack*, where criminals utilize the same sort of relay to fake transactions.

Given the relay attack described, we can generalize to a communication model with parties, tag T and reader R. The attacker can have two modes of operation in implementing the relay attack. The first is defined as a digital relay attack, and the second is defined as an analog relay attack. If the parties are being targeted by a digital relay attack, the attacker utilizes proxies to communicate directly between the parties. These proxies can be considered as a proxy tag PT and proxy reader PR. PR directly communicates with T and PT communicates with R. The two proxies share a link between each other to forward messages they receive. The direct link between the proxies and the legitimate communicating parties utilizes whatever communication methods are established in the protocol being attacked. Before the proxies forward communications between themselves, they would have to demodulate and decode the signals before encoding and remodulating for relaying. Since the signal is being remodulated in the link between the proxies, it is not necessarily the same mode of communication used in the protocol. Furthermore, once the second proxy receives the relay digital information, it would demodulate and decode, before further encoding and remodulating the signal towards the final target. The proxy-to-proxy link utilized by the attacker can effectively increase the range of the protocol to a length much beyond the intended use case. This allows the attacker to vary how the attack is being done. The attack is then called a digital relay attack, because the digital information is what is being directly relayed between each link. We can see the model for this relay attack in Figure 2.2:

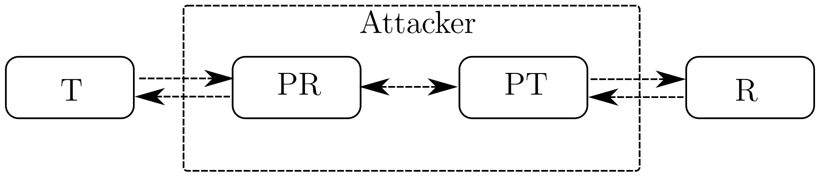


Figure 2.2: Digital relay attack in a two party communication system

In contrast, an analog relay attack is more akin to a radio repeater. A series of analog circuits with antennas or reflectors, are utilized by the attacker to re-transmit the exact received signal towards the legitimate reader or tag. The signal would mostly be kept unaltered; no demoduation and modulation would occur, and only the analog information is relayed. Compared to a digital relay attack, there are not necessarily any proxies being utilized by the attacker. As well, an analog relay attack would mostly be limited by range, since it would happen purely over the wireless channel, and would have to have a chain of relays to get an effect distance. It is possible to quantize the signal and forward it to another device for re-transmission, but there would be a loss in precision. With this in mind, a digital relay attack can comfortably have a proxy link that is over a wired connection, dramatically increasing the potential range. For example, a coordinated pair of attackers could implement a digital relay attack, where the reader and tag are many kilometers away from each other. Regardless of the mode of operation to implement the relay attack, the results are the same. The attacker manages to get the reader and the tag to communicate with one another without the typical expectation of communicating in the instance or environment.

The focus of the relay attack in this paper will be done on the passive authentication system. These systems typically utilize the challenge-response authentication protocol in many real world scenarios. Examples include passive key-less entry system (PKES) and tap payment services. The challenge-response protocol authenticates by showing a proof of knowledge with a verifiable response. This can be done publicly or symmetrically, but involves the utilization of a private key. When both parties already share keys (i.e. symmetric-keys), a response to a random challenge can be done with a tag generated by a message authentication code (MAC). The parties authenticate each other by verifying the MAC on the challenge. For example, if a owner wanted to unlock their car as they got near it, a PKES would have a key fob send a request for the car to generate a challenge. The key fob and the car would both already share a symmetric-key, so the key fob would respond to the challenge with a MAC. If the MAC verifies with the challenge, the car would unlock itself. All this would occur without any input from the owner, for their convenience. This form of PKES is used in many other RFID and higher frequency authentication systems. Symmetric-key challenge-response protocols can also provide mutual authentication as shown in Figure 2.3:

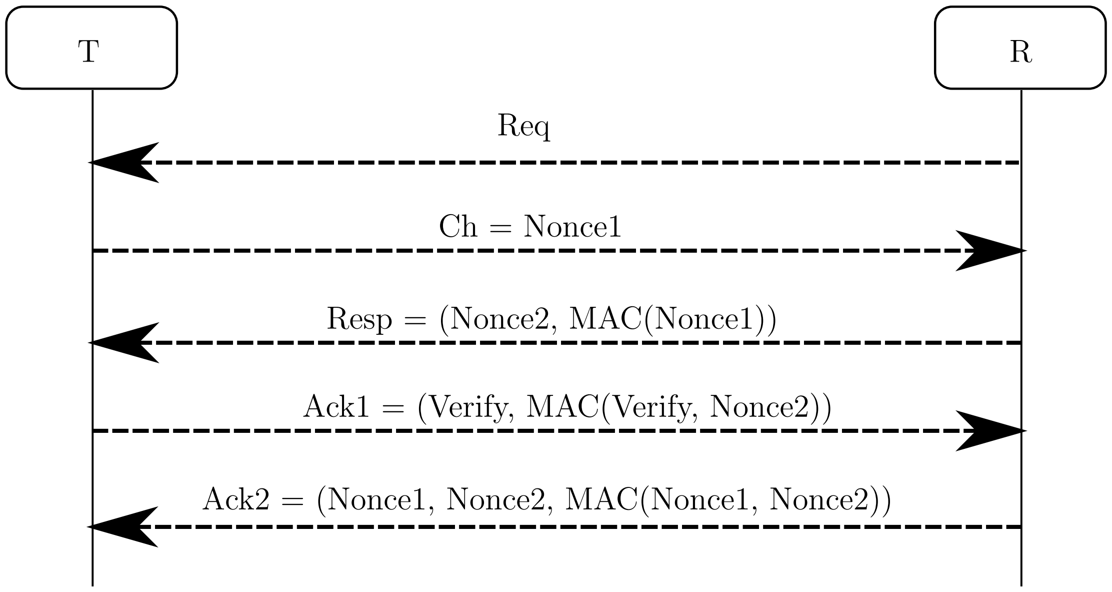

T
R
Req
Ch = Nonce1
Resp = (Nonce2, MAC(Nonce1))
Ack1 = (Verify, MAC(Verify, Nonce2))
Ack2 = (Nonce1, Nonce2, MAC(Nonce1, Nonce2))

Figure 2.3: Mutual symmetric-key Challenge-Response authentication protocol

If we were to inject a MITM performing a digital relay attack in the challenge-response protocol, the protocol would proceed as if T and R were communicating directly with each other. If we consider the example of unlocking a car with a PKES, the car would be unlocking itself by not directly communicating with the key fob, but by communicating to the proxy key fob (i.e. the attacker). We can see the model for the relay attack on the mutual challenge-response protocol in Figure 2.4. This relay attack allows the challenge-response protocol to extend the range to whatever distance the proxy link is.

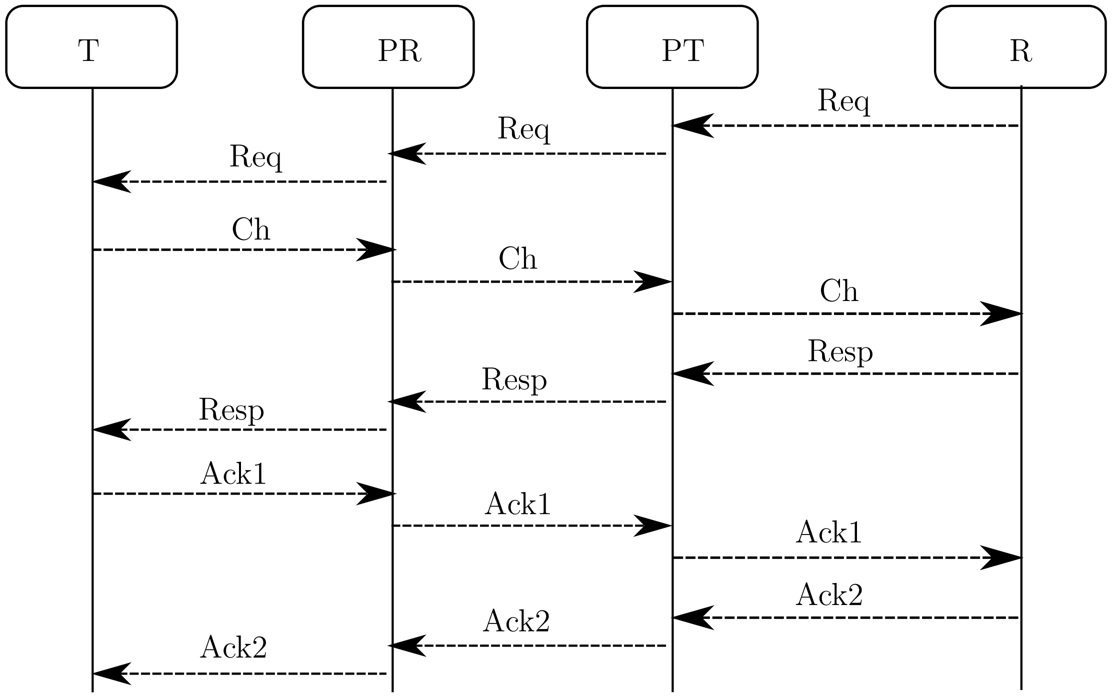


Figure 2.4: Relay attack on mutual symmetric-key Challenge-Response authentication protocol

Regardless of the strength of the cryptographic protocols being used in the challenge-response protocol (i.e. key bit-size, MAC protocol, nonce bit-size), the relay attack will always extend the communication. Whatever challenge is generated will just be relayed between the parties, with a response generated and relayed back. We can imagine a scenario where the legitimate owner of a parked car is roaming in the public. A pair of collaborating attackers acting as the proxies are coordinating an attack on the owner and his car. One attacker stands next to the owner, while the other attacker stands next to the car. The attackers then initiate the authentication protocol between the car and the key in the owner's pocket. Once the attackers finish relaying the exchange, the car will think it has authenticated with the key and will unlock itself to be stolen. This happens without the owner's knowledge.

A limiting factor is that both parties have to expect the protocol to be occurring or be in a mode where the challenge-response occurs automatically (like a PKES). This means the attacker would have to get the proxies to trigger the start of the protocol. As well, being able to insert a MITM to such protocol requires access to the channel, such that the MITM receives the messages before the intended parties. In practice, a relay attack is typically limited to wireless communication channels. Particularly, where the communications used are limited in range and the devices are passively powered. Examples include RFID, near-field communication (NFC), BlueTooth, and ZigBee systems. These systems are especially vulnerable to relay attacks, as what they authenticate is typically tied to the real world (i.e., a cyber-physical system). For example, the success of authentication leads to a door being unlocked, or a physical transaction completing.

2.2 The Wireless Communication Channel

The wireless communications channel characterizes the statistical properties and effects of the physical environment used to transmit and receive messages. This includes phenomenon such as: path loss, shadowing, multi-path effects, inter-symbol interference (ISI), and thermal noise. Real channels are changing in time due to relative movements of receivers, transmitters, and reflectors. Reciprocity can also exists in the channels between the transmitter and receiver, and between the receiver and transmitter. Reciprocity means that transmissions from the transmitter to the receiver and vice-versa, travel through channels that have a high correlation with one another. Reciprocity can exist between different channels depending on time, frequency, and distance. In this section a review of signals and the characterization of a wireless communication channels is given to understand how reciprocity can be utilized to prevent the relay attack, and how to use correlation to improve the performance from other solutions in literature. As a reference, most of the derivations and theory comes from *Wireless Communications*, a textbook by *A. Goldsmith* [3]. More detail on the derivations can be found in the textbook.

2.2.1 Signal Representation

When we discuss the transmitted and received signals, we need to have a proper mathematical representation that allows us to capture all of the possible effects that can occur. In this context, a signal is a function in time with a dependent variable that is some physical quantity, such as voltage or current. The standard in literature is to utilize a low frequency signal to represent higher bandwidth components. This representation is called *baseband equivalent representation*. Signals that occupy a frequency around zero are called baseband signals, while signals that occupy a frequency far from zero are called bandpass signals. In the baseband equivalent representation, a bandpass signal for transmission is represented by a combination of a low frequency signal that carries information and a high frequency carrier signal used to modulate the information signal. The frequency at which a transmission occurs is important factor in the transmission. For example, the frequency dictates the size of the antennas needed for communications, the rate at which information can be transmitted, as well as the effectiveness of the signal to propagate over different mediums and distances.

In the wireless communication system, we represent signals to be transmitted as a vector on an orthonormal basis (signal constellation). The digital bit-stream in the baseband environment is encoded to a vector on this plane. Given the vector, the coefficients are parallelized before being modulated into a bandpass signal for transmission with the orthonormal basis. The receiver performs the inverse operations by demodulating the received signal into a vector and decodes this received vector into a digital bit-stream. We can view this generic communication system in the Figure 2.5:

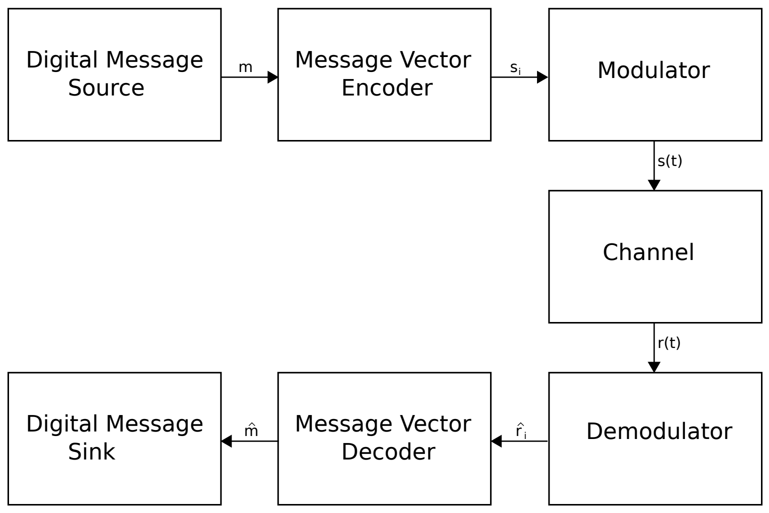


Figure 2.5: Typical Wireless Communication System Block Diagram

How each of these subsystems are implemented depends on the communication protocol being used. This includes the modulation schemes, channel model and message sources.

For the system modeled, we can consider a signal, $x(t)$, between $t \in [0, T_s]$ with bandwidth B_s. The signal is carrying the digital information in an analog form. An antenna functions in the real world, so only real analog signals can be produced and transmitted. The signal is also a baseband signal, and can have a form dictacted by the transmission that represents digital information. We can call $x(t)$ the real baseband signal. To modulate $x(t)$ to some carrier frequency f_c, based on the properties of the Fourier Transform:

$$s(t) = x(t)\cos(2\pi f_c t)$$

where $s(t)$ is the modulated signal for transmission, as shown in Figure 2.5, and is a bandpass signal. In this case the orthonormal set used as the basis is simply $\{\cos(2\pi f_c t)\}$. For IEEE 802.11n, which is the communication scheme of focus in this thesis, the digital modulation techniques utilized are M-ary Phase Shift Keying (MPSK) and M-ary Quadrature Amplitude Modulation (MQAM). For both techniques, the basis used is:

$$\phi = \{\cos(2\pi f_c t), -\sin(2\pi f_c t)\}$$

For MPSK, the coefficients are given by:

$$s_{i1} = A\cos(\theta_i), s_{i2} = A\sin(\theta_i)$$

$$\theta_i = \cos(2\pi(i-1)/M)$$

where $i \in \{\mathbb{Z}|1 \leq i \leq M\}$ and is dictated by the symbol encoding. M is the number of symbols to be represented in the system. For example, in quadrature phase shift keying (QPSK), $M = 4$ and symbol 00 can be coded to $i = 1$. The Coding is typically chosen to minimize the bit error rate.

For MQAM, the coefficients are given by:

$$s_{i1}, s_{i2} = (2i - 1 - L)d$$

where i and M are defined the same as for the MPSK technique, and d is a chosen equidistance between signal constellations. This distance is constrained by the transmission energy of the transmitter. The larger the energy, the bigger the distance.

In both of these digital modulation techniques we can generalize the signal coefficients as the real baseband signals. We can then represent the transmitted signal as:

$$s(t) = x(t)\cos(2\pi f_c t) - y(t)\sin(2\pi f_c t)$$

where $x(t)$ and $y(t)$ are the real baseband signals (or signal constellation coefficients). The signals, $x(t)$ and $y(t)$, are referred to as the in-phase component and quadrature component respectively. We can also represent the transmitted signal in a complex bandpass representation:

$$s(t) = \Re\{[x(t) - jy(t)]e^{j2\pi f_c t}\}$$

This representation makes it easy to capture all of the channel effects that can occur.

All signals also have a frequency response. This allows us to characterize the frequency components of a signal. By utilizing the Fourier Transform and Inverse Fourier Transform, we can convert back and forth between the time and frequency domain. We represent the Fourier Transform on the signal as shown:

$$S(f) = F\{s(t)\}$$

$$s(t) = F^{-1}\{S(f)\}$$

The frequency response of a signal also reveals the bandwidth and power spectral density of a signal. This information is important to quantify how a signal interacts with a channel.

2.2.2 Time-Invariant Channel

With a model for a transmitted signal, we need a reciprocal representation for a received signal. A transmitted signal travels through a channel that has many effects on the signals. The most basic interaction involves thermal noise, which manifests as additive white Gaussian Noise (AWGN) on the channel. More complex effects include multipaths. A multipath channel results in different paths being traveled by the same transmission to the receiver. These components have different angles of arrival and distances traveled. Relative movements can also cause fading due to Doppler's effects and delay paths, resulting in a phase change. These different phase changes result in a destructive superposition of the multipath signals. As well, there are long distance effects, such as path loss of transmission energy and shadowing caused by environmental qualities. When we look at a channel, we can consider it to be a system (filter) in which the transmitted signal is going through. The most basic channel is a linear time-invariant (LTI) channel. An LTI channel has has both the linearity properties and time-invariance mathematical properties.

Basic signals and systems theory tells us that the output of an LTI system is the linear convolution (in time) between the input and the impulse response of the system. The impulse response is the output signal when the impulse signal is input into the system. We represent the impulse response with the signal $h(t)$. We can then represent the output signal by the following operation:

$$r(t) = h(t) * s(t) + z(t)$$

We note that the function $z(t)$ is the AWGN which maintains its Gaussian distribution. We can also perform these operations in the frequency domain to simplify the math, as multiplication is happening instead of convolution:

$$R(f) = S(f)H(f) + Z(f)$$

Since the transmitted signal is a bandpass signal, only the portion of the impulse frequency response around the carrier frequency is considered. The impulse response is the bandpass response. The area of consideration is the bandwidth B of the transmitted signal around the carrier frequency. This impulse response can also be expressed by a baseband component that is filtered around B and modulated to the carrier frequency. This baseband component of the impulse response is convoluted with the baseband component of the transmission. The received signal can be expressed as the modulation of this result:

$$r(t) = \Re\{r_\ell(t)e^{j2\pi f_c t}\} + z(t)$$

Where $r_\ell(t)$ is the resulting equivalent baseband signal of the received transmission.

2.2.3 Time-Variant Channel

A channel that has the time invariance property is not common in real systems. Movements of transmitters, receivers, and reflectors causes the environment of the channel to change with time. This affects the many delay paths a signal travels and produces a Doppler's frequency shift. As mentioned, these effects result in phase changes that cause the constructive or deconstructive superposition of the different multipath signals, and can result in fading in the received signal. Fading means that the received signal is attenuated, to a point where it cannot be reliably decoded. Multipath delays can also result in inter-symbol interference (ISI) if the maximum delay in the multipath component is longer than the symbol time of the original transmission. If symbols are transmitted successively, then a signal delayed by the symbol time would appear when the next symbol also arrives. To represent the multipath effects, the impulse response becomes $h(t,\tau)$. The variable t represents the time at which the response is received, while τ represents the delay in transmission. If a path exists resulting in a delay τ and arrival t, then the impulse was sent at time $t-\tau$. The time of transmission of an impulse is based on τ. To further capture the time variant effects on the received signal we have an expanded expression for it:

$$r(t) = \Re\{\sum_{n=1}^{N} \alpha_n(t) r_\ell(t-\tau_n(t)) e^{j2\pi f_c[t-\tau_n(t)]+\phi_{D_n}(t)}\} + z(t)$$

The summation captures each of the resolvable multipaths for the received signals that superimpose. Unresolveability describes the multipaths that are sampled at a given short interval, such that they are indistinguishable from one another. In this equation, there are $N \in \mathbb{N}$ resolvable paths (for the unresolvable groups), but this may change in time with the addition or removal of reflectors in the relative environment. For each resolvable path there is a varying gain, delay, and Doppler shift. The function $\alpha(t)$ captures the time varying attenuation that occurs due to path loss or other effects. The time varying delay is represented by $\tau_n(t)$, which occurs due to the different lengths traveled by each multipath to the receiver. The final effect captured is the Doppler shift $\phi_{D_n}(t)$, which is a phase shift resulting from the frequency shift caused by Doppler's effect. We note that the phase changes can have a significant effect over a short time-frame, when compared to the effects of path loss or the delays themselves. Other than ISI, a delay only has a large effect because the carrier frequency is multiple magnitudes larger. If the phase is represented in the units of radians, and the carrier frequency is in kilohertz (at the very least), the resulting phase due to the multiplication of the two can be large enough to change the signal. The effects on the channel can be considered to be random variables in time (random processes). Furthermore, under the assumption that each resolvable path is independent from the other resolvable paths, the received signal can be considered as the sum of N independent samples. If N is large enough, then we can invoke the Central Limit Theorem to express the received signal as a Gaussian random process, as noise is also Gaussian distributed. If each path is considered to be independent from one another, then we know that they must also be uncorrelated. This is defined as uncorrelated scattering, which is another assumption we can make on the channel.

An extra effect on the received signal that needs consideration is the differences in physical hardware. Hardware, in wireless communications, includes components such as: clocks, antennas, converters, mixers, modulators, etc. The physical differences in these components can cause changes in timing and frequency synchronization between the transmitter and receiver. These differences add another phase change, that causes a constant effect on each received signal. To derive this phase:

$$r(t) = \Re\{\sum_{n=1}^{N} \alpha_n(t) r_\ell(t - \tau_n(t)) e^{j2\pi(f_c + f_{hdw})[t + t_{hdw} - \tau_n(t)] + \phi_{D_n}(t)}\} + z(t)$$

where t_{hdw} is the hardware difference in the clocks, and f_{hdw} is the frequency difference. As mentioned, the small difference in timing is amplified by the large carrier frequency. In general, we condense these differences into a constant phase term ϕ_{hdw}:

$$r(t) = \Re\{\sum_{n=1}^{N} \alpha_n(t) r_\ell(t - \tau_n(t)) e^{j2\pi f_c[t-\tau_n(t)]+\phi_{D_n}(t)+\phi_{hdw}}\} + z(t) \tag{2.1}$$

We will make the assumption that this phase is relatively constant in time, but varies with the frequency.

For a time-variant impulse response, the received signal can still be evaluated by the convolution with the transmitted signal:

$$r(t) = h(t, \tau) * s(t) + z(t)$$

In the frequency domain, we note that the response also changes with time. Since τ captures the behavior of the impulse response, we can get the frequency response by taking the Fourier Transform of the impulse response in respect to τ:

$$H(f, t) = \int_{-\infty}^{+\infty} h(\tau, t) e^{-j2\pi f\tau} d\tau$$

We can use the frequency response of the time-variant channel and the frequency response of the transmitted signal to express the received signal:

$$R(f) = H(f, t)S(f) + Z(f)$$

The impulse response $h(t, \tau)$ can be expressed as a complex function, with both a phase and a magnitude. Since the received signal and noise are Gaussian random processes, and the transmitted signal is deterministic, the channel impulse response is also Gaussian distributed by the central limit theorem. How the different random processes (noise, phase, amplitude) are distributed depends on the model used to represent the channel and its environment. For example, the phases will have different distributions based on the existence of a line-of-sight (LOS) and the positioning of the reflectors.

2.2.4 Delay Spread and Coherence Bandwidth

The different delays of the many multipath signals allow us to characterize effects in the channel. For example, as mentioned in subsection 2.2.3, the existence of a multipath signal with a long delay can interfere with subsequent transmissions (ISI). To express the effective delays in the channel, the delay spread (T_m) is used. The delay spread is typically defined as the difference in the delay from the first received signal and the last. This can be calculated by taking the first signal as the reference, and comparing it last multipath received. Of course, there may be infinite delayed multipath signals in theory, but if the power of noise is considered the floor of acceptance, then interference below this level can be discarded.

Given that $h(t, \tau)$ can be expressed as a Gaussian process and under the assumption it is wide-sense stationary and the multipaths are independent, we can use its autocorrelation to characterize it:

$$\begin{aligned} A(\tau_1, \tau_2; t_1, t_2) &= A(\tau_1, \tau_2; t, t + \Delta t) \\ &= A(\tau, \tau + \Delta\tau; \Delta t) \\ &= E[h^*(\tau, t) h(\tau + \Delta\tau, t + \Delta t)] \\ &= A(\tau; \Delta t)\delta(\Delta\tau) \\ &\triangleq A(\tau; \Delta t) \end{aligned}$$

The movement from the first line to the second line in the above equation comes from the wide-sense stationary property. The third line to the fourth line comes from uncorrelated scattering. Since different delays are uncorrelated, we only get correlation when $\Delta\tau = 0$. If we further take $\Delta t = 0$ for the autocorrelation, then we get what is referred to as the power delay profile:

$$A(\tau; \Delta t)|_{\Delta t = 0} \triangleq A(\tau)$$

The power delay profile gives us the amount of power at different delays. We can use this function to calculate the statistical delay spread. Typically, the root-mean-squared delay spread is used as an accurate estimation of the delay spread in the channel.

The coherence bandwidth is the bandwidth of the channel, where different frequency components have high correlation with one another. Given that $h(t, \tau)$ has a Gaussian distribution, the frequency response $H(f,t)$ also has a Gaussian distribution. We can then take its autocorrelation to characterize it and the coherence bandwidth:

$$\begin{aligned}
A(f_1,f_2;t_1,t_2) &= E[\int_{-\infty}^{+\infty} h^*(\tau_1,t)e^{j2\pi f_1\tau_1}d\tau_1 \int_{-\infty}^{+\infty} h(\tau_2,t+\Delta t)e^{-j2\pi f_2\tau_2}d\tau_2] \\
&= E[\int_{-\infty}^{+\infty} h^*(\tau,t)h(\tau,t+\Delta t)e^{-j2\pi(\Delta f)\tau}d\tau] \\
&= \int_{-\infty}^{+\infty} A(\tau;\Delta t)e^{-j2\pi\Delta f\tau}d\tau \\
&\triangleq A(\Delta f;\Delta t)
\end{aligned}$$

We go from the first line to the second line due to uncorrelated scattering, which allows us to factor τ from the exponential. The result ends up being the Fourier transform of the autocorrelation of the impulse response. If we take $\Delta t = 0$, then the result is the autocorrelation in respect to the frequency difference Δf. This function allows us to determine when different frequency components decorrelate. We can define the coherence bandwidth (B_c) as the smallest positive root of $A(\Delta f)$. The relationship between the delay spread and the coherence bandwidth comes from the fact that $A(\Delta f)$ is the Fourier transform of $A(\tau)$. If we consider $A(\tau)$ to have low power components beyond the delay spread (T_m), then $B_c \propto \frac{1}{T_m}$.

For transmission signals that have bandwidths larger than the coherence bandwidth, their frequency components outside of the coherence bandwidth become uncorrelated, and can potentially superimpose destructively. This fading is defined as frequency-selective fading. If the signal's bandwidth is within the coherence bandwidth, we define this as flat-fading.

2.2.5 Coherence Time and Uncorrelated Distances

If we notice the autocorrelation in respect to $\Delta\tau$ and Δt, we can derive the correlation in terms of the difference in observation time by setting $\Delta\tau = 0$:

$$A(\Delta\tau,\Delta t)|_{\Delta\tau=0} = A(\Delta t)$$

Similar to the coherence bandwidth, we define the coherence time (T_c) to be the lowest positive root of $A(\Delta t)$. The coherence time of a time-variant channel is the amount of time that has to pass before the channel becomes uncorrelated. At a high level, the coherence time can be thought of as the rate at which the channel is changing. Any observation of the channel is only representative for the coherence time. This means that a transmission in time must be completed within the coherence time to ensure there is no fading. If fading caused by a changing channel occurs, we define the channel as a fast-fading channel. Similarly, if the coherence time is large relative to the transmission time, it does not affect fading. The channel is considered to be a slow-fading channel in this circumstance.

Given the difference in time between channel correlation, we also want to establish a distance for which the channel becomes uncorrelated. Let us consider two receivers seperated by some distance d. The two receivers receive the signals $r_1(t)$ and $r_2(t)$ respectively. We define $r_1(t)$ to be the arrival at the first receiver, and $r_2(t)$ the arrival at the other one. As well, we make the assumption that the two receivers are in the same environment; the scattering for each multipath comes from the same reflectors. If the receivers were not in the same environment, then the number of multipaths and delays would be uncorrelated. Another assumption is that the receivers exist in a dense scattering envrionment. This means that they are equally likely to receive a multipath from every possible angle of arrival $(0, 2\pi]$. Under these assumptions we can evaluate the cross-correlation of between the two received signals. In *Goldsmith's* book [3], the relation to the uncorrelated distance is a Bessel function of the 0^{th} order. This distribution tells us that after a distance of several wavelengths the channels are fully decorrelated.

2.2.6 Orthogonal Frequency Division Multiplexing

As shown in subsection 2.2.4, the multipath channel has a delay spread, which if large enough can cause frequency-selective fading and ISI. To defeat these effects, the transmitted symbols must have a symbol duration (T_s) that is significantly greater than the delay spread ($T_s >> T_m$). This can also be interpreted as the bandwidth of the signal being significantly smaller than the coherence bandwidth ($B << B_c$). If we consider a single frequency being utilized, by increasing the symbol duration of the transmission (decreasing bandwidth), we are simultaneously reducing the rate of the transmission (by Shannon-Hartley Theorem). Orthogonal Frequency Division Multiplexing (OFDM), intends to defeat this phenomenon by parallelizing a single stream transmission, into many smaller transmissions that will ideally utilize the same full bandwidth when combined. Each of these parallel frequencies must be orthogonal from one another to prevent interference. The orthogonality also allows the subcarriers to overlap to achieve high spectral efficiency.

For a channel with a set coherence bandwidth (B_c), we want to take a single frequency transmission with a set bandwidth (B), and split its bandwidth into N parallel transmissions on different "subcarrier" frequencies, as opposed to a single carrier frequency, such that each of the N transmissions has a bandwidth $<< B_c$. We set the subcarrier bandwidth to be $B_N = \frac{B}{N}$, such that $B_N << B_c$. To implement OFDM in practical systems, the Inverse Fast Fourier Transform (IFFT) algorithm is used to minimize the hardware requirements for transmitting, and the Fast Fourier Transform (FFT) is used for receiving. For transmitting, a modulated symbol stream is parallelized into the N substreams. A set of N of these symbols are used as the input to the IFFT to generate N subcarrier transmissions.

2.2.7 Channel Reciprocity

In subsection 2.2.4 and subsection 2.2.5, we defined three aspects in which the channel becomes uncorrelated: changes in frequency, time, and distance. If we consider a transmitter and receiver that are communicating, under the assumption that the channel is wide-sense stationary and has independent scattering, we have high correlation under the following simultaneous conditions:

1. The respective carrier frequencies and corresponding transmission bandwidth are within the same coherence bandwidth of the channel,

2. The round-trip (including signal processing time, transmission rate, and time-of-flight) is within the coherence time of the channel,

3. The individual movements of the transmitter, receiver and reflectors in the effective environment is less than the wavelength of the carrier frequency.

All current practical wireless communication systems are half-duplex; communications only occur from a single device over a given frequency and time instance. Under the above conditions, the half-duplex channel will appear reciprocal (highly correlated) between the transmissions of both parties. Specifically, the impulse response, H_{12} (seen from the transmitter to receiver) and response H_{21} (seen from receiver to transmitter), are relatively equal under the conditions. Furthermore, for any other device that exists in the wiretap channel, if it does not satisfy these correlation conditions between one or both of the communicating parties, it will have an impulse response that is uncorrelated with the others. An example of a reciprocal exchange in a half-duplex wireless system can be seen in Figure 2.6:

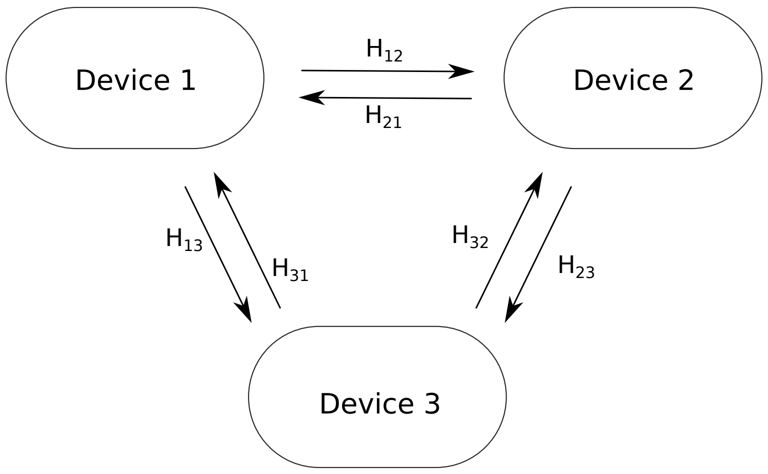


Figure 2.6: A 3-Device Half-Duplex System with a Reciprocal Exchange

If we can make the assumption that the exchanges between the devices are occuring within the correlation parameters, and that the devices are seperated from one another by at least a few wavelengths, then we find the following relation between the channel states H_{ij}:

$$H_{12} = H_{21}$$

$$H_{13} = H_{31}$$

$$H_{23} = H_{32}$$

$$H_{12} \neq H_{13} \neq H_{23}$$

When an attacker exists on the wireless wiretap channel and is attempting to relay messages, as long as the devices are seperated by a multiple of the wavelength, the channel measurements between each hop will be different. This is similar to the observation seen in Figure 2.6. This phenomenon is the motivation for detecting relay attacks.

2.3 IEEE 802.11n

The Institution of Electrical and Electronic Engineers (IEEE) first released the 802.11 protocol in 1997 to take advantage of the newly deregulated 2.4 GHz frequency band. Their aim was to create a standard that would allow fast, reliable, and ubiquitous access for devices trying to communicate on a wireless area network (WAN). The implementation of the protocol on devices is colloquially known as Wireless Fidelity (WiFi), and has become widespread over the last decade, appearing in billions of consumer devices. Since the original standard was released, several standard amendments have been released to improve performance, accessibility, flexibility, and other performance criterion. Technologies such as modulation schemes, frequency bands, antenna arrays, and multiplexing have all been modified in newer 802.11 amendments to help improve the performance and usage of the protocol in different scenarios. Anecdotally, the most utilized 802.11 standard is the 802.11n amendment released in 2009. It introduced key technologies into the 802.11 standard such as multiple-input multiple-output (MIMO) diversity, the utilization of both the 2.4 GHz and 5 Ghz frequency bands, orthogonal frequency division multiplexing (OFDM), frame aggregation, and beam-forming. IEEE 802.11n allows user devices to achieve high throughput (HT) transmissions to meet growing consumer demands. If we consider the International Organization for Standardization (ISO) five layer networking model in Figure 2.7:

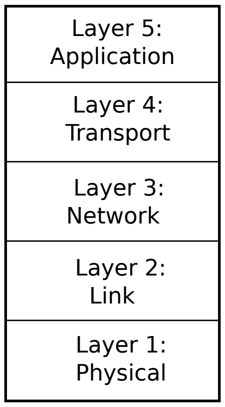


Figure 2.7: Five Layer Networking Model

IEEE 802.11 exists across the link and physical layers. In this section an overview of the link and physical layers for IEEE 802.11n are given. Detail is also given on the OFDM implementation in 802.11n, and how channel estimation is utilized with it.

2.3.1 Link Layer

The link layer is typically thought to be split into two sublayers: medium access control (MAC) and logical link control (LLC) sublayers. The MAC layer intends to deal with the problems of data encapsulation and media access management [34]. Data encapsulation includes tasks such as: data frame delimitation (framing), synchronization, device addressing, and error detection capabilities. Media access management includes the tasks of medium allocation for the transmitter, and contention resolution if a collision (multiple transmitters at the same time) occurs in the half-duplex channel. The functionalities of the MAC layer can be managed by many different standards, such as the IEEE 802.11 standard. The LLC primarily acts as the interface from the upper network layer to the MAC layer. When data from the upper layers of communications are ready to be transmitted across a channel to the next hop device, the LLC sublayer provides the data to the MAC layer for encapsulation and transmission to meet the requirements of the standard. Similarly, when data is received on the MAC layer, the LLC transfers the data to the network layer [35]. A figure of this arrangement can be seen below:

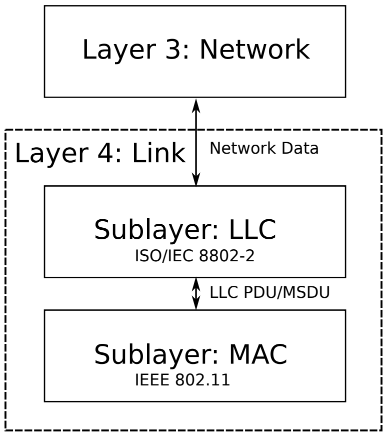


Figure 2.8: Link Sublayers

As shown in Figure 2.8, the newest LLC standard established by the IEEE is ISO/IEC 8802-2 (International Electrotechnical Commission). ISO/IEC 8802-2 has a protocol data unit (PDU) that encapsulates the upper layer data. The LLC PDU adds a header which includes a service access point (SAP) field from the source and destination. The SAP communicates the protocol that the upper layers of the source and destination utilize. An example of such a protocols is the Internet Protocol (IP). The LLC PDU is further encapsulated into a MAC protocol data unit (MPDU), where the LLC PDU is becomes referred to as the MAC Service Protocol Data Unit (MSDU). For IEEE 802.11, the MSDU is encapsulate by the header and a frame checking sequence (FCS), which is automatically added based on the preceding bytes. The format for the IEEE 802.11n header is in Figure 2.9:

Frame Control	Duration\ID	Address 1	Address 2	Address 3	Sequence Ctrl	Address 4	QoS Ctrl	HT Ctrl
2 Bytes	2 Bytes	6 Bytes	6 Bytes	6 Bytes	2 Bytes	6 Bytes	2 Bytes	4 Bytes

Figure 2.9: IEEE 802.11n MAC Frame Header Format

The Duration and ID field either has an identifier for the station on a WAN, a duration based on the frame subtype, or a fixed indicator value. The addresses are all long term identifiers for a station, known as the MAC address. Address 1 and Address 2 indicate the transmitter and receiver respectively in the wireless communication. Address 3 indicates the AP (if one exists in the WAN), otherwise it is the same address as the transmitter. Address 4 is used if the MPDU is going to be translated into another standard, such as IEEE 802.3 (Ethernet), to follow the next hop communication. The Sequence Control field identifies the sequence number of an MPDU (12 bits) and identifies the fragment (4 bits). The quality of service (QoS) field provides control information when a QoS frame with a data subtype is transmitted. The High Throughput (HT) Control field is utilized in certain control, QoS, and management frames to provide control information for the HT functionality. The Frame Control provides information on the type of frame being transferred. Frame control has the following expanded fields:

Protcol Version	Type	Subtype	To DS	From DS	More Fragmentations	Retry	Power	More Data	Protected	Ordered
2 Bits	2 Bits	4 Bits	1 Bit	1 Bit	1 Bit	1 Bit	1 Bit	1 Bit	1 Bit	1 Bit

Figure 2.10: IEEE 802.11 Frame Control Format

The protocol version identifies the version of IEEE 802.11 being used. By default it is set to '00', since there is only one protocol version in the standard at the moment. The type identifies one of four MPDU types: control (01), management (00), data (10), and QoS (11). The subtype identifies several different subtypes of the MPDU for each of the main types. For example, for type data, subtype '0000' indicates a pure data frame. A detailed table of all the possible subtypes can be found in this CISCO guide [7]. To Distribution System (DS) and From DS indicate if a station is playing the role of an access point (AP) or a client. If they are a DS (From DS = 0), then the station plays a role as an AP. If neither devices communicating are an AP (To DS = 0, From DS = 0), the stations will operate in an adhoc network. Similarly, if both devices are an AP (To DS = 1, From DS = 1), then the stations will operate in a mesh network. The more fragmentation fields indicates if the MSDU contains a fragmentment packet from the network layer. The retry field indicates if the current PDU is a retransmission. The Power field communicates to the receiver if the transmitter is changing to a low powered state after the transmission occurs. If a station indicates changes into a low powered state after a transmission, another station communicating with it may use the More Data field to prevent the station from entering a low powered state. The Protected field indicates that the MSDU is encrypted and authenticated with the security scheme established in the system. The Ordered field indicates if a data frame must be processed in order or arrival [8].

2.3.2 Physical Layer

The physical layer contains two sublayers that handle its functionalities: the Physical Layer Convergence Procedure (PLCP) and Physical Medium Dependent (PMD) sublayers [37]. The PLCP is a translational sublayer similar to the LLC in subsection 2.3.1, where it must manage the MPDU, through encapsulation and extraction. The PMD sublayer deals with the coding, modulation, and transceiving of the data frames. Once an MPDU is ready for transmission on a wireless channel, it becomes a PLCP Service Data Unit (PSDU) and is moved on to the PLCP from the link layer. Similarly, when a wireless frame is received, it is sent by the PMD to PLCP to be interpreted and parsed into an PSDU. The PLCP prepares the PSDU for transmission on the PMD, as established by the communication standard (e.g. IEEE 802.11) [35]. A model of the layers can be seen in the Figure 2.11

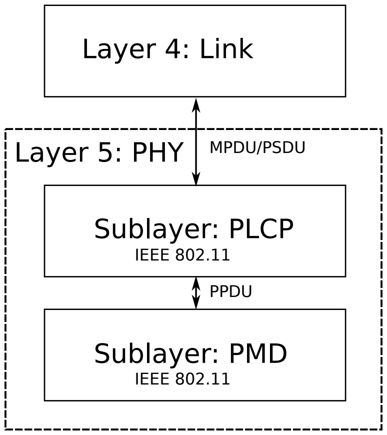


Figure 2.11: Physical Sublayers

For IEEE 802.11n, the PLCP encapsulates the PSDU by adding a preamble and header to it. The result is a PLCP Protocol Data Unit (PPDU), which is sent to the PMD. The purpose of the preamble is intended to provide time and frequency synchronicity, and channel estimation for equalization. The header provides information about the format and transmission information of the frame. This includes information such as: how many antennas are being utilized, the modulation scheme, whether or not HT is occuring, how large the guard interval (GI) is, the expected rate of transmission, and the bandwidth utilized for each transmission. IEEE 802.11n was designed to be backwards compatible between the older version standards, which allows for three possible PLCP modes for creating a PPDU. The modes are: Non-HT PPDU, HT-mixed format PPDU, and HT-greenfield format PPDU [35]. Below, Figure 2.12 and Table 2.1 (from the *IEEE 802.11-2012 Standard* [9]) show the format for the PLCP modes and the subsequent data:

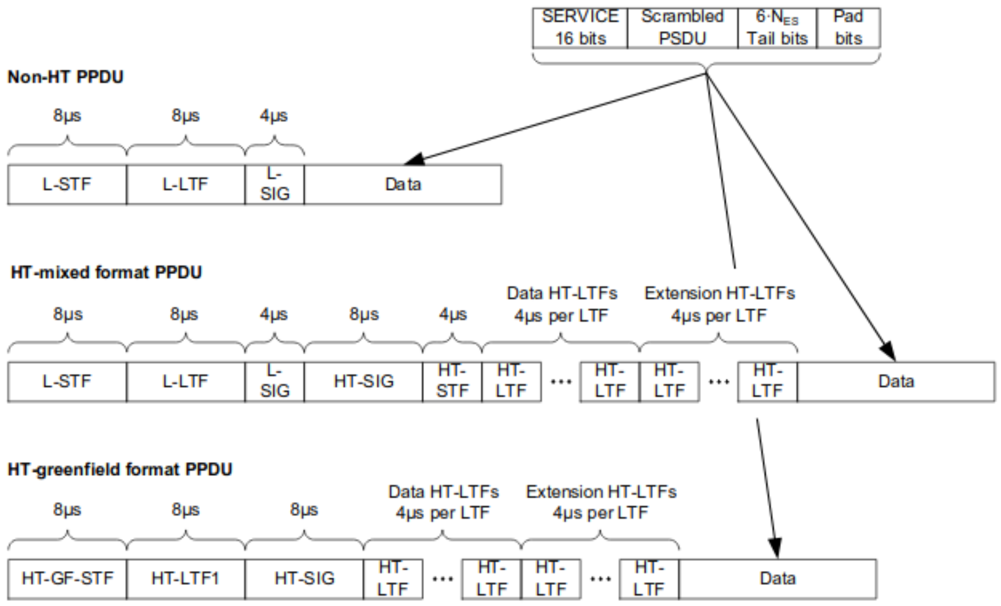
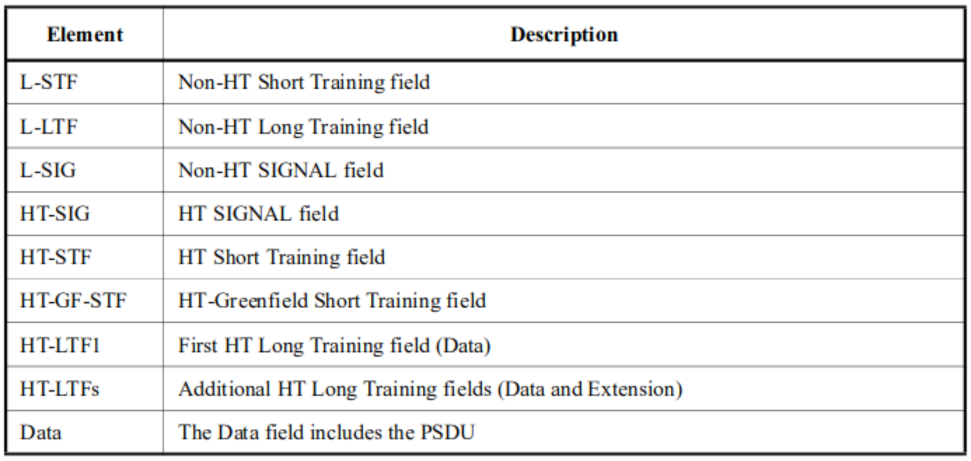


Figure 2.12: Different Formats for the PLCP

Table 2.1: Elements in PLCP



Element	Description
L-STF	Non-HT Short Training field
L-LTF	Non-HT Long Training field
L-SIG	Non-HT SIGNAL field
HT-SIG	HT SIGNAL field
HT-STF	HT Short Training field
HT-GF-STF	HT-Greenfield Short Training field
HT-LTF1	First HT Long Training field (Data)
HT-LTFs	Additional HT Long Training fields (Data and Extension)
Data	The Data field includes the PSDU

For the proposed implementation of the relay attack detection, HT-greenfield format is utilized for the PLCP format. The transmission is read in big-endianness, where HT-STF is transmitted and received first. In the PPDU, the preamble consists of HT-STF and the HT-LTFs. The header is the HT-SIG. HT-STF is used for the intial timing of the transmission, coarse-grained timing and frequency synchronization, and automatic gain control. The HT-STF is transmitted using 10 tones over 2 OFDM symbols, with only 16 subcarriers. HT-LTF is used to provide fine-grained timing and frequency synchronization, and it is used to estimate channel state information (CSI) for the forthcoming data. The HT-LTF1 is mandatory, but the extra HT-LTFs can be used to improve the fine-grained synchronization. The HT-LTF1 is twice as long as the HT-LTF, and transmits 2 symbols over all the subcarriers data is transmitted on. Each HT-LTF subcarrier will provide a CSI for each of the subchannels. Due to the coherence bandwidth, not all the subcarriers will be correlated, so individual estimations are needed from the CSI results. The HT-SIG provides detailed information for the receiver so they know how to frame the incoming data. The format for the 2 HT-SIG symbols from the *IEEE 802.11-2012 Standard* is shown in Figure 2.13:

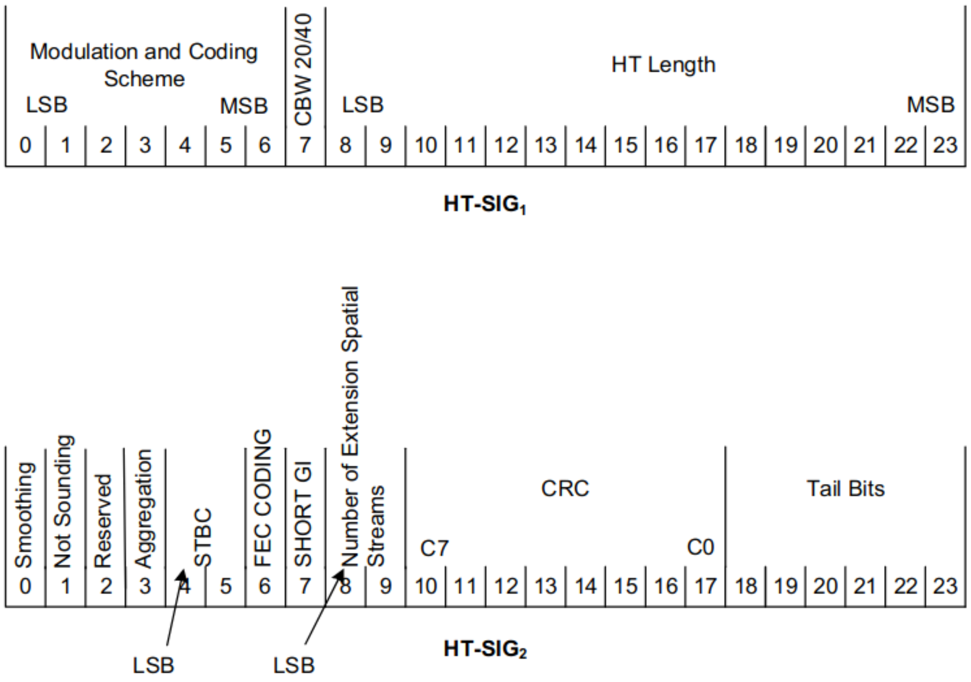


Figure 2.13: Format for HT-SIG in HT-Greenfield PLCP Format

HT-SIG is modulated using the lowest rate BPSK (for resilience) and OFDM with all 52 subcarriers. The Modulation and Coding Scheme (MCS) is specifcally used to signal to the receiver how the data is going to be modulated and coded. Depending on the HT MCS index, the bandwidth of the channel, and the guard interval duration, different transmission rates are possible. For example, if the MCS index was set to '1000000' and the was set to 800 ns with a total bandwidth of 20 MHz, the transmission speed would be 13 Mb/s with BPSK. As mentioned in subsection 2.2.1, the options range from BPSK to 64-QAM.

The PMD consists of the physical hardware necessary to transmit and receive the PPDU. IEEE 802.11n in North America (Canada & USA) allows 11 carrier (numbered 1 through 11) around the 2.4 GHz frequency: 2.412, 2.417, 2.422, 2.427, 2.432, 2.437, 2.442, 2.447, 2.452, 2.457, 2.462 GHz. There is also a second set of channels around the 5 GHz frequency, but the usage is heavily regulated due to priority for government and commericial radar. The bandwidth allocated for the transmission is 20 MHz or 40 MHz at both frequencies and for all modulation and coding schemes. IEEE 802.11n also utilizes OFDM with all its transmissions, where the modulation and coding is dictated by the MCS from the PPDU. With a 20 MHz bandwidth, there are a total of 64 OFDM subcarriers, with the following distribution: 8 subcarriers are used for the cyclic prefix and guard intervals, 4 subcarriers are utilized for further pilot signals, and 52 subcarriers are utilized for the data. With a 40 MHz bandwidth, there are 128 subcarriers, with the following distribution: 14 subcarriers are used for the cyclic prefix and guard intervals, 6 subcarriers are utilized for further pilot signals, and 108 subcarriers are utilized for data. IEEE 802.11n also allows up to 4 antennas to achieve diversity. The cyclic prefix can occupy either 0.8 μs of the 4 μs OFDM symbol duration, or 0.4 μs of the 3.6 μs OFDM symbol duration. The view of the transmitter in the PMD can be observed in the Figure 2.14:

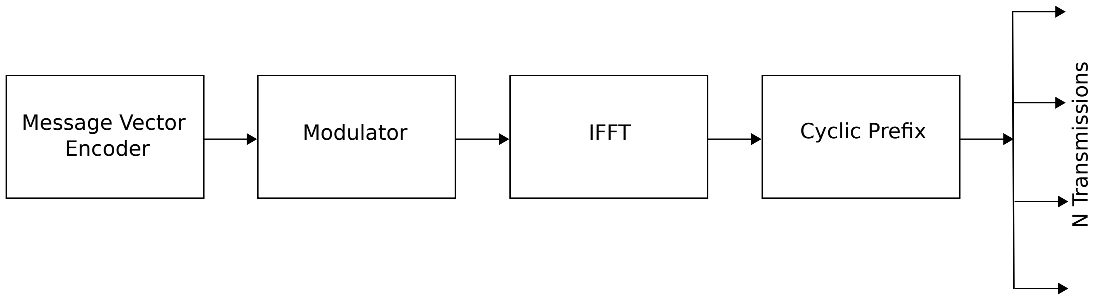


Figure 2.14: PMD Transmission module

We can model the receiver to function in a reciprocal manner to the transmitter. The functionality of the PLCP and PMD have to follow the standard of IEEE 802.11ln, but the implementation (hardware and software) can be done at the discretion of the engineer. For example, Intel and Qualcomm may choose to implement the management of the PSDU and PPDU between the different sublayers and subcomponents in their own way.

IEEE 802.11 utilizes carrier sensing multiple access system with collision avoidance (CSMA/CA) to provide half-duplex multiple access to the wireless channel. CSMA/CA is the standard protocol for all variations of the IEEE 802.11 standard. At a high level, the protocol has devices wait to transmit until the WAN appears silent (i.e. no transmissions are occurring). This is known as carrier sensing. If it appears silent for a certain amount of time, the waiting device will transmit. Since wireless communications is half-duplex, the transmitting device cannot tell if a collision occurs (multiple transmissions), so it waits for an acknowledgement from its target. If no acknowledgement is received after a time, the device assumes their was a collision and backs off for a random time, before trying the whole process again. This is known as collision avoidance. If the WAN has a high amount of devices trying to communicate, there may be a large delay before a transmission can occur, as all the devices compete for access. This can affect the ability to have a reciprocal exchange that is correlated, as the delay can be larger than the coherence time.

2.3.3 Channel Estimation

As mentioned in subsection 2.3.2, the CSI is estimated from the HT-LTF in the preamble preceding a PPDU. At the 20 MHz band, HT-LTF utilizes all 52 subcarriers that the data is also transmitted on, so channel estimation can be done for all the data on the different subcarriers. Depending on how the estimation is done, a complex measurement that captures the impulse response is found.

2.4 Related Works

As mentioned in section 2.1, the first formal proposal of the relay attack in literature was in John Conways' *On Numbers and Games* [6] in 1976. From this point many papers have looked at creating practical relay attacks in many different environments.

One of the major proposed solution for detecting the relay attack in wireless systems was by David Chaum and Stefan Brand in their 1993 paper *Distance-bounding protocols* [5]. Their paper focused on mafia-attacks, which are a form of relay attack, but their solution extends to all forms of relay attacks. Distance-bounding measures the delay of the signals being transmitted to get a coarse measurement of the end-to-end and round-trip distances. If the distance is within some expected upper bound, the measurement is accepted. One of the limiting factors with distance-bounding is that it is latency limited.

If the measurements are based on using the round-trip time (T) and the speed of light ($C = 3 \times 10^8$ m/s) to estimate the distance, any small difference in synchronization or processing time (latency) can add some small delay (τ). If this delay is in the range of nanoseconds or more, then the added distance can be significant ($d_{err} = \tau \times C$). For short distance communications (PKES, payment systems, etc.), any small latency can cause there to be false-positive, as the d_{err} can added several metres to the round-trip time (i.e. $T < \tau$). Specifically, The addition of this error in the distance can cause the round-trip time to surpass the upper-bound, making this system think a relay attack is extending the distance. In more modern literature, different distance-bounding protocols have been implemented for different devices, such as RFID [10]. Unfortunately, the latency issue will always be a problem [11], and most proposed solutions lack practical implementations. Similar "Time-of-Flight" solutions exist. For example, measuring the RF signal strength was proposed in one paper [12], but the solution in this paper was under the assumption that the response to the challenge can be directly measured at the reader (which might not always be the case). The use of time-of-flight, based solutions also does not deal with attacks that are within the threshold of acceptance.

In general, solutions to the relay attack attmept to take advantage or introduce an *unrelayable channel*. Proposed in the paper *Multichannel protocols to prevent relay attacks* [13], the theoretical unrelayable channel has the properties of weak unclonability, strong unclonability, unsimulability, and untransportability. The unclonability should make it prohibitively difficult to copy the source. Unsimulability should make it prohibitively difficult to copy the response. Untransportability should make it prohibitively difficult to transfer the unrelayable information to another location. This unrelayable channel should be used to exchange unique information. The result of the exchange should make it so a failed response is detected as a relay attack. Distance-bounding is such an example of information exchanged on an unrelayable channel. It is impossible for the relays to alter the physical distance needed for the response to travel for verification. Since the response has to come from the legitimate parties, it should be impossible for the relays to clone or simulate the response. Since the transportation is limited by the speed of light, even if transportability was possible, it would not improve any performance, so the channel is untransportable as well. Of course, the issue with distance-bounding is in its practicality, not its design.

To utilize this unrelayable concept, we look at the physical layer. A seminal work in the field of physical layer security was done in the paper *On the effectiveness of secret key extraction from wireless signal strength in real environments*, in 2009 by *Suman Jana et al* [15]. The paper discusses the utilization of the received signal strength information (RSSI) between reciprocal pilot signals to establish a unique secret key. Due to the reciprocal channel (discussed in subsection 2.2.7), two parties could extract a key from the RSSI measurements. The paper introduced a new secret key extraction protocol and looked at other major protocols by *Aono et al.* [17] , *Tope et al.* [18] ,*Mathur et. al* [16] , and *Azimi-Sadjadi et al.*[19]. The paper explains the idea of a Lossy and Lossless exchange of information through channel information, and the trade off between entropy and bit generation rate. The paper's major contribution was through the quantization protocol called Adaptive Secret Bit Generation (ASBG), which several other papers would later adapt. Other physical layer solutions attempt to utilize reciprocal information such as the phase. One paper, *Cooperative Secret Key Generation from Phase Estimation in Narrowband Fading Channels* from 2011 [21], suggested the use of the reciprocal phase change (due to doppler effects and delay) in a narrowband rayleigh channel to extract a secret key. Since the phase in this channel is theoretically uniformly distributed over $[-\pi, \pi]$, this would give a theoretically perfect distribution for generating a key. Unfortunately, the solution has the same timing problems that exist with distance-bounding that make it impractical. Synchronicity and hardware difference make it impractical to implement such a system as unrelayable information. Finally, a seminal paper proposed the utilization of channel state information (CSI) as reciprocal information for key extraction. A *Fast and Practical Secret Key Extraction by Exploiting Channel Response* by *H. Liu et al.*, proposed the usage of channel state information in the IEEE 802.11n standard to drastically improve the performance of the key extraction [21]. The addition of many OFDM subcarriers and MIMO antennas, allows many uncorrelated pilots to be utilized for high thoughput key generation. They implemented their protocol with an Intel network interface card (NIC), and compared it to the other RSSI key extraction protocols, with vast improvements in performances. The paper also introduces Channel Gain Complement (CGC), which is method to remove non-reciprocal components in a transmission to increase correlation. This protocol is utilized in the protocol proposed in this thesis. Several papers adapted this protocol, such as in *KEEP: Fast secret key extraction protocol for D2D communication* [22]. Fundamentally, all the key extractions protocols (RSSI, phase, and CSI) utilize the unrelayable reciprocal channel, which can be used to identify relay attacks with a challenge-response protocol.

A Mobile Adhoc Network (MANET) must have all its nodes periodically update the graph of the network to find the shortest distance to other nodes. If the weight is based off of a distance measured from RSSI, then a relay attack could trick a node in to making a fraudulent graph of the network. This form of relay attack is reffered to as a wormhole attack. Solutions for wormhole attacks are equivalent to solutions for relay attacks. One such solution to the wormhole attack was explored in the paper *Preventing Wormhole Attacks Using Physical Layer Authentication* [25]. Further development to generalize a solution was done in the paper *Preventing Relay Attacks and Providing Perfect Forward Secrecy using PHYSEC on 8-bit μC* in 2018 [26]. The paper looks at creating a practical solution to relay attacks (and wormhole attacks). Similar to the RSSI secret key generation that dominates physical layer security, this paper proposes the usages of the reciprocal channel as an unrelayable channel. RSSI is exhanged on the unrelayable channel, and is used to check if a relay attack is occuring. The short paper did not discuss the protocol in detail, but did do a practical implementation in IEEE 802.15.4 (ZigBee) to show the practicality of the solution for solving relay attacks. Another solution added noise to a channel to keep track of the amount of hops occuring in the transmission [14]. This solution is not feasible for wireless communications since it required full-duplex communications.

Chapter 3

The Usage of Channel State Information To Detect Relay Attacks

The relay attack has been shown in section 2.1 to be a simple and potentially devastating attack on many cyber-physical systems that rely on challenge-response authentication. Being able to detect the occurrence of a MITM can be used to identify a relay attack (or other forms of MITM attacks), when a relay is not expected in the system. In this chapter, a protocol is proposed that uses the reciprocity of channel state information to detect a digital relay attack during a modified channel-response authentication protocol. Specifically, a full description of the protocol is discussed, as well as the motivations in literature for the aspects of the protocol.

3.1 The Protocol

The usage of a digital relay attack is most prevalent in circumventing challenge-response based protocols. Particularly because the hardware requirements are very low. For example, many relay attacks can be implemented on smartphones, with only upper layer implementations. For this reason a modified one-way challenge-response authentication protocol is used as the base in the relay attack detection protocol. In this protocol a symmetric-key based system is used to provide authentication, similar to the mutual-authentication protocol described in Figure 2.3. These keys are assumed to be installed on the communicating devices prior to the execution of the authentication protocol. We can observe the general steps in the relay detection protocol in the table below:

Table 3.1: Relay Attack Detection Protocol

Step	Description
1. Initiation Request:	A tag (T), that wants access to a system, makes a request to authoritative reader.
2. Request Acknowledgement:	The reader acknowledges the tag's request by responding with an explicit ACK, and a challenge (nonce).
3. Pilot Exchange:	The tag begins an equal exchange of pilot signals between itself and reader.
4. Channel Gain Complement:	The tag encrypts and transmits its CSI from the pilot to the reader to allow the reader to adjust for discrepencies.
5. Authentication:	The tag transmits a MAC on the challenge and of its quantized CSI.
6. Verification:	The reader verifies the tag's response. If the MAC and the quantized CSI match, then the tag is authorized, otherwise it is rejected.

The protocol and the messages can be further visualized in Figure 3.1:

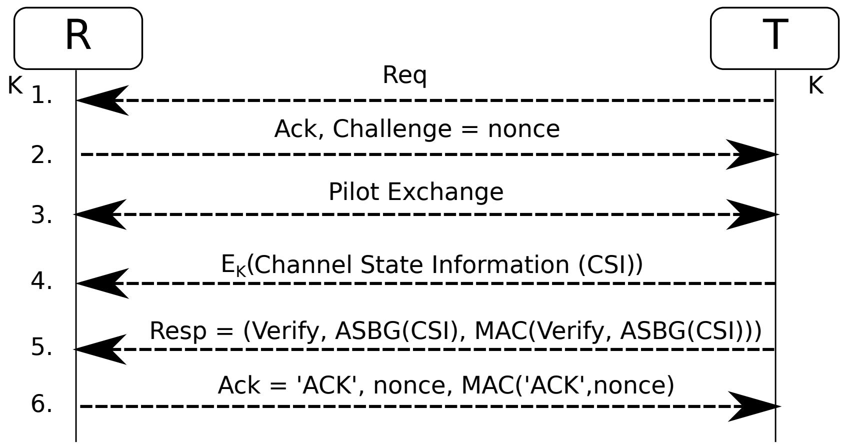

R
T
K
K
1.
Req
2.
Ack, Challenge = nonce
3.
Pilot Exchange
4.
E_K(Channel State Information (CSI))
5.
Resp = (Verify, ASBG(CSI), MAC(Verify, ASBG(CSI)))
6.
Ack = 'ACK', nonce, MAC('ACK',nonce)

Figure 3.1: Relay Attack Detection Protocol

3.1.1 Initiation Request

For a PKES, the *Initiation Request* is a passive communication dependent on the standard being used. For example, in a Bluetooth or WiFi system, the request may be established by constantly listening for broadcasts from the reader. When a tag hears a broadcast with an identifier, it can make a request. Other standards such as passive RFID/NFC, will wait for the tag to be powered by the reader to initiate. Once the request is received by the reader, a 128-bit nonce is generated for usage as the challenge. The nonce must be stored in memory on the reader for authentication. An acknowledgement and the nonce are sent to the tag to continue the authentication protocol.

3.1.2 Request Acknowledgment

When the acknowledgement is received by the tag, it is verified before the nonce is stored in memory for future authentication. If the nonce is not long enough or the acknowledgement has not been received for a *time-out period* (ToP), the tag resets the protocol.

3.1.3 Pilot Exchange

The tag begins a reciprocal exchange with the reader with pilot messages. A single exchange occurs for fixed pilot time up to ToP, at a pilot exchange rate R_p in units of $pilots/second$. When a pilot message is received it is stored in memory, with a time-stamp of when it was received. Each received pilot message is used to calculate an estimation of the CSI in the reciprocal channel. The channel estimation is considered to be the complex gain and phase represented by h_p. Each CSI measurement can be quantized into a single bit, which is utilized to detect the presence of a MITM.

3.1.4 Channel Gain Complement

The phase difference, ϕ_{hdw}, described in subsection 2.2.2 and Equation 2.1, expresses the difference that is caused by frequency and timing synchronization. The reciprocity in the pilot signals is a requirement for the authentication between the reader and tag. While the channel components can be reciprocal, the hardware phase may result in discrepencies between the measured channel transfer functions. To help compensate for these differences, a method known as Channel Gain Complement (CGC) is used to compensate. The method was first designed in *Fast and Practical Secret Key Extraction by Exploiting Channel Response* [21], to eliminate channel differences for secret-key generation. The implementation follows the description of the protocol provided in the paper. To perform CGC, the tag transmits its encrypted CSI measurements to the reader. The messages are encrypted so a MITM does not have the ability to modify the measurements to influence the CGC process. For a single pilot measurement, we can observe the frequency response of both the reader and the tag respectively:

$$\begin{aligned} h_{p,R} &= h_p + n_R + h_{hdw,R} \\ h_{p,T} &= h_p + n_T + h_{hdw,T} \end{aligned} \tag{3.1}$$

where the assumption is that h_p is the reciprocal complex channel effect if the pilots are exchanged within the same coherence time. The addition of random AWGN, n_R and n_T, is different for each measured response. As well, there is the physical hardware term, which manifests in an additive term $h_{hdw,R}$ and $h_{hdw,T}$ for the reader and tag respectively. We should note that in this method, they made an assumption that there is an additive hardware difference. Typically, many of the hardware changes affect the timing or frequency, which would add a phase change. A phase change is multiplicative change in the transmitted signal, but the CGC method assumes there is also an additive hardware effect. The additive effects are what is removed.

If the two measurements are close enough in time to each other, then their quantization should result in a bit symbol that is close to equal. The CGC method utilizes the mean of the difference between the shared measurements to estimate $h_{hdw,R}$. Another assumption is that for a given exchange in the protocol, the additive hardware differences are constant over the many instances of communication $h_{hdw} = \rho_{hdw,R}$. For N pairs of reciprocal exchange measurements we define the mean difference:

$$\begin{aligned}\mu_{diff} &= \frac{1}{N}\sum_{i=1}^{N} h^i_{p,R} - h^i_{p,T} \\ &= \frac{1}{N}\sum_{i=1}^{N} h^i_p + n^i_R + h_{hdw,R} - (h^i_p + n^i_T + h_{hdw,T}) \\ &= \frac{1}{N}\sum_{i=1}^{N} (n^i_R - n^i_T) + (\rho_{hdw,R} - \rho_{hdw,T})\end{aligned}$$

The noise is AWGN with some gaussian distribution $\sim N(\mu_{noise}, \sigma^2)$. The difference, $h_{p,R} - h_{p,T}$, has a statistical mean:

$$\begin{aligned}\mu &= \mu_{noise} - \mu_{noise} + \rho_{p,R} - \rho_{p,T} \\ &= \rho_{p,R} - \rho_{p,T}\end{aligned}$$

The Weak Law of Large Numbers and Chebyshev's Inequality tell us that $\mu_{diff} \to \mu$ as $N \to \infty$. The encrypted CSI measurements sent to the reader in the *CGC* stage are used with its own reciprocal CSI measurements to calculate the average difference μ_{diff}. Once the reader has μ_{diff}, it can subtract the measurement from all its measurements:

$$\begin{aligned}h'_{p,R}(t) &= h_{p,R} - \mu_{diff} \\ &= h_p + n_R + \rho_{p,R} - \rho_{p,R} + \rho_{p,T} \\ &= h_p + n_R + \rho_{p,T}\end{aligned} \tag{3.2}$$

Compared to the measurement from the tag, $h_{p,T}$, the reader's complemented measurement, $h'_{p,R}$, only differs in the noise, though the distribution is the same.

After the reader transmits the last pilot signal, it will wait for ToP before it resets the authentication.

3.1.5 Authentication

Following the transmission of the tag's CSI measurements, the tag generates the response to the challenge, and its quantized CSI measurements. With the symmetric-key, the MAC used can be a Hash-MAC (HMAC), or any other MAC primitive, as long as the reader follows the same standard.

To quantize the CSI measurements, a method known as Adaptive Secret Bit Generation (ASBG) is used. A form of this method was proposed by *Mathur et al.* for quantization [20], but their method did not account for long-term fading effects. The ASBG by name and final design, was created by *Suman Jana et al* [19]. The protocol vastly improved the bit generation rate, without compromising the entropy, compared to the other RSSI key generation methods at the time. ASBG requires four parameters: the block-size m, symbol size n, quantizer threshold modifier α, and the acceptance threshold d. These parameters would be established by the reader and tag prior to the occurrence of quantization. Given a set of N measurements in time, the measurements are grouped by chronological order into blocks of size m. For each block, the mean and variance is calculated on the magnitudes of the CSI measurements. The mean and variance for each block is used to calculate the upper and lower adaptive thresholds, $q+$ and $q-$ respectively:

$$q+ = mean + \alpha\sqrt{variance}$$
$$q- = mean - \alpha\sqrt{variance}$$

where $\alpha \geq 0$, and is chosen experimentally to maximize performance. Furthermore, to increase the bit-generation rate, n can be chosen as any integer:

$$2 \leq n \leq \lfloor \log_2(max(h^i) - min(h^i)) \rfloor$$

where the minimum and maximum are chosen from within the measurements in a block. Each region above and below the thresholds is further separated in to 2^{n-1} quantization bins. Measurements are encoded based on the quantization bin they fall within. Each bin is encoded into a n-bit symbol based on Gray-coding, to minimize the error. If a measurement falls within the region between $q+$ and $q-$, it is discarded. The quantization of the CSI measurements results in a bit-stream with a maximum size of $N \times n$. To add on to this protocol, we further increase entropy with a deterministic method. The symmetric-key is hashed with the nonce to generate an output sequence. This sequence is grouped into indices, which are use to randomly select k bits from the CSI generated bit stream. The k-bits are used as authentication against the relay attack with the challenge-response. Since n can be chosen to increase the amount of bits extracted from a single measurement, we would expect the overall entropy to decrease. By separating the measurements into blocks, long-term and slow fading effects (such as path-loss and shadowing) are better accounted for, which provides an increase in entropy.

It is important to recognize that the generated response does not need to be kept a secret. Since the response is only transmitted once with a nonce and the MAC, it is unique, and only useful for the single instance of verification. Replay attacks would be ineffective due to the nonce, and the response would be rejected if it was modified, due to the MAC.

3.1.6 Verification

To verify the response from the tag, the reader performs ASBG on all its modified CSI measurements, $h'_{p,R}(t)$. The reader must also perform ASBG on the tag's values received in the CGC phase, and must drop the measurements that the tag dropped. Since the reader cannot drop measurements the tag did not drop, if a measurement would otherwise fall within the threshold of $q+$ and $q-$, the reader selects the measurement to be encoded to a 0 or 1 with equal distribution. This must be done to ensure that they both have the same output length k. To compare the results, the reader checks if the hamming distance between the reader's quantized measurements and the tag's response are less than or equal to d. If they are, then the response is accepted by the reader to have come from the tag, otherwise it is rejected. Due to channel reciprocity and correlation, if a MITM was performing a digital relay attack, then the CSI measurements would be different between the tag, MITM, and the reader. We can visualize this exchange in Figure 3.2:

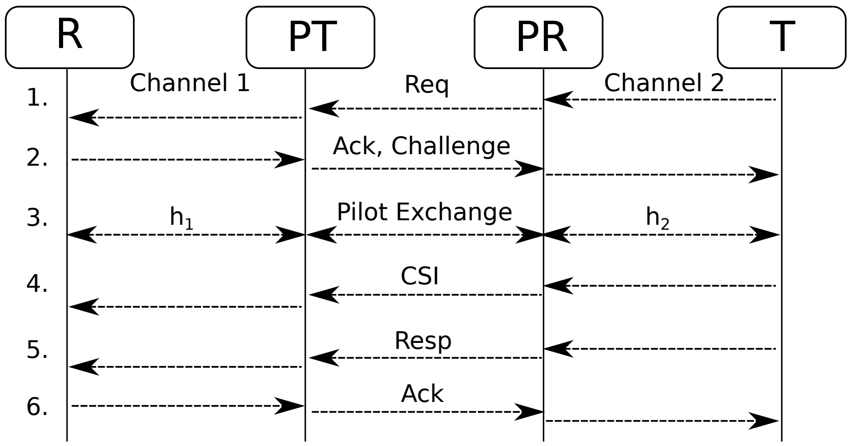


Figure 3.2: Digital Relay Attack against Protocol

If the channels between the proxies and the legitimate parties are uncorrelated, then the CSI measurements in each channel will be different from one another. Under a digital relay attack, the forwarding between the proxies would not carry over the channel effects in the proxy link, resulting in an unrelayable measurement from the channel at the reader and tag.

While N needs to be large for CGC result to be precise, the amount of bits needed to be quantized and compared does not need to be large. Since a unique MAC is generated for the challenge and the quantized CSI, a MITM cannot alter the response. If the uncorrelated results are truly random (i.e. seemingly uniform), then the attacker would require roughly $\sqrt{2^{2k}} = 2^k$ attempts (by the birthday problem), for the two CSI measurements to match. For any challenge-response system, the reader could have a threshold of maximum attempts to try to provide authentication. For example, if a key attempts and fails to unlock a car after 1000 tries, the car can set off its alarm, or send an alert to the owner that an unauthorized access is being attempted. Thus, k can be in the low double digit bits (e.g. 10 bits).

The benefit of the usage of CSI for relay attack detection, is that a high exchange rate of pilots can occur to allow for a large N for CGC estimation in a short period of time. The coherence time acts as a minimum distance to extract a k. Since k is small in size, only a few coherence times are necessary to generate the output. In retrospect, N is large (i.e. many occurrences in a coherence bandwidths), because it is only needed for CGC.

Once the reader has verified the response, it generates an acknowledgement to let the tag know it has been authenticated. The acknowledgement can also be provided through the allowance of access to the system (e.g. a car, door, payment system, etc.)

3.2 Performance Metrics

In the area of physical layer key-generation, the typical performance metrics utilized are the bit-generation rate, randomness, and mismatch rate. The key-generation rate quantifies how fast the system and protocol are able to generate random bits. The randomness of the generated bit stream can be quantified via a pseudorandomness test. The mismatch rate quantifies how likely legitimate communicating parties are to have different results. For key-exchange protocols, a mismatch would result in the devices having different keys, thus unable to communicate. For the relay attack detection, a mismatch would result in a false-positive detection of an attacker.

3.2.1 Key-Generation Rate

The key-generation rate in the protocol, R_k (in units of *bits/second*), has a minimum performance:

$$R_k = \frac{1}{1 + 2N \times ToP} \frac{R_p \times pilot\_size}{2k} \tag{3.3}$$

In this calculation, the processing time is considered to be neglible for quantization. In the worst case, each of the N reciprocal exchanges that occur can take a maximum of ToP seconds each. Furthermore, each pilot consists of *pilot_size* amount of bits, but only k of the bits of the output stream are used, and half the rate is utilized for transfering duplicates in the pilot exchange.

3.2.2 Randomness

The basic necessity for the protocol to prevail, is that the quantized output is uniformly distributed. As mentioned in subsection 3.1.6, if it is uniformly distributed, then it is unlikely that when a relay attack is occurring, that the reader and tag will have the same output. If there is a bias for a certain result, then the system may have a false-negative; the devices authenticates, despite the existence of a MITM. To evaluate the randomness properties, it is necessary that the system passes basic pseudorandomness test, such as: entropy, 0-1 distribution (frequency), run length, etc.

3.2.3 Mismatch Performance

As mentioned in subsection 3.1.4, a mismatch may occur due to the difference in the noise for a pair of measurements, and due to the phase differences caused by hardware (only additive differences are removed with CGC). In Equation 3.2, the noise of the modified CSI measurement at the reader comes from n_R, while the noise at the tag is n_T. The mismatch performance is given by the probability that a mismatch (false-positive) occurs for a single measurement. The probability of a mismatch can be approximated to the difference of a Rayleigh (non-LOS) or Rician (LOS) channel with the subtraction of a noise component. For the particular implementation in this thesis, we are able to make this approximation due to the fact that OFDM turns a wideband signal into many narrowband signals. The narrowband model allows us to categorize the channel effects into a Rayleigh or Rician distribution.

Chapter 4

Implementation of The Relay Attack Detection Protocol

Compared to many of the proposed solutions for the relay attack, this chapter will describe an easy and open implementation. The implementation is done using open source tools, packages, drivers, and cheap consumer products that can be utilized in any modern computer. The main communication standard utilized for the challenge-response protocol is IEEE 802.11n. The implementation helps provide a proof-of-concept that the protocol is feasible to implement. In this chapter a guide is given to set-up the implementation, followed by an overview the implementation.

4.1 Set-up Guide

This section focuses on setting up the tools, packages, and other software necessary to recreate this instance of the protocol implementation.

4.1.1 Operating System

Based on testing and per the recommendations of the creators of the *Linux 802.11n CSI Tool*, the most up to date operating system (OS) that can be used is Ubuntu 14.04.1 LTS. A link to the old Ubuntu releases that contain the OS can be found in [30]. The OS cannot be ran on a virtual machine, but can be dual booted. The OS should be updated before installation. This can be done by running the follow command in the terminal:

```
sudo apt-get install update
sudo apt-get upgrade
```

The update will change the kernel, but there should not be any problem with any of the software packages used. The rest of the installation should happen within this OS. It is recommended that the build-tools be installed prior to the loss of internet connectivity in the other steps.

4.1.2 Network Interface Card

The 802.11n CSI Tool used can only be installed with the *Intel 5300* NIC. You cannot use the tool with any other NIC from Intel. This includes newer models such as *Intel 6300*, *Intel 7260*, *Intel 8260*, *Intel 9260*, etc. While the NIC was released in 2012, it should still work in any modern desktop or laptop. As of April 2019, the NIC can still be purchased through the American or Canadian Amazon website [31].

The *Intel 5300* NIC uses a half-sized mini-peripheral component interconnect express (mini PCIe) interface. Most laptops have a mini-PCIe slot that can be used. If a laptop does not have this slot or if it is inaccessible, then the laptop cannot be used with the CSI tool used here. Other open source tools exist, which may be capatible. For a desktop that does not have a mini-PCIe slot, a PCIe $\times 1$ to mini-PCIe adapter can be used. It is also recommended that if an adapter is used, it has a cut-out to fit an antenna. A mini-PCIe to USB adapter will not work due to the scheduling and low bandwidth. The NIC requires an antenna to have a decent range. For the implementation a single omni-directional 5 dBi rubber ducky antennas was used for each NIC (no MIMO). The connection between the

NIC and antennas use a copper 50 ohm impedance U.FL to female SMA interface. Both of these components can be found at any electronics store. Due to the CGC method described in subsection 3.1.4, the possible performance losses due to having different antennas should be slightly reduced.

4.1.3 Linux CSI Tool

The Linux 802.11n CSI Tool is a combination of a custom firmware for the *Intel 5300* NIC, as well as custom Linux drivers for iwlwifi. The project was created with the collaboration of a group of researchers, *Daniel Halpern et. al*, with a team of engineers from Intel [25]. The tool provides an easily accessible method to extract the CSI estimations used for equalization during WiFi communications. Typically, the only data that is accessible is upper layer data (Link and above), with no option to access the PPDU header that contains the pilot results. The tool provides access to this information that is typically unaccessible. Detailed instructions to install the tool can be found through the team's guide [25], but is also detailed below to keep the thesis self-contained. All of these instructions are in the Ubuntu command terminal:

Installation Instructions:

1. Download the Tool from the repository:

   ```
   sudo git clone https://github.com/dhalperi/\
   linux-80211n-csitool.git
   ```

2. Navigate to the repository in the terminal.
3. Install the build-tools, development headers, and git-core:

   ```
   sudo apt-get install gcc make linux-headers-$(uname -r) git-core
   ```

4. Obtain the CSI Tool's source tree for the drivers and check out the repository for your upstream kernel version:

```
CSITOOL_Kernel_TAG=csitool-$(uname -r | cut -d . -f 1-2)
git clone https://github.com/dhalperi/linux-80211n-csitool.git
cd linux-80211n-csitool
git checkout ${CSITOOL_Kernel_TAG}
```

5. Establish the kernel tag for the version of Ubuntu. If the previous instructions were followed, then it should be *Ubuntu-3.13.0-164.214*. If you are using a different OS, then the kernel version be found by running the following in the terminal:

```
 sudo /proc/version_signature
```

6. Once the kernel version is found, the tag value can be found through online searches. Once found, set the variable:

```
UBUNTU_Kernel_TAG= #Kernel_TAG
```

7. Merge the Linux kernel versions:

```
. /etc/lsb-release
git remote add ubuntu git://Kernel.ubuntu.com/ubuntu/\
ubuntu-${DISTRIB_CODENAME}.git
git pull --no-edit ubuntu ${UBUNTU_Kernel_TAG}
```

8. Build the modified driver for the existing kernel:

```
make -C /lib/modules/$(uname -r)/build\
M=$(pwd)/drivers/net/wireless/iwlwifi modules
```

9. Install the modified drivers into the updates directory:

```
sudo make -C /lib/modules/$(uname -r)/build M=$(pwd)\
/drivers/net/wireless/iwlwifi INSTALL_MOD_DIR=updates \
    modules_install
sudo depmod
cd ..
```

10. Obtain supplementary material. This includes programs and scripts to test the tool:

```
git clone https://github.com/dhalperi/\
linux-80211n-csitool-supplementary.git
```

11. Relocate the existing Intel firmware:

    ```
    for file in /lib/firmware/iwlwifi-5000-*.ucode;\
     do sudo mv $file $file.orig; done
    ```

12. Install the firmware:

    ```
    sudo cp linux-80211n-csitool-supplementary/firmware/\
    iwlwifi-5000-2.ucode.sigcomm2010 /lib/firmware/
    sudo ln -s iwlwifi-5000-2.ucode.sigcomm2010 /\
    lib/firmware/iwlwifi-5000-2.ucode
    ```

13. Build the logging tool:

    ```
    make -C linux-80211n-csitool-supplementary/netlink
    ```

After the firmware and drivers have been installed, whenever the tool is required to log CSI measurements (e.g., during the relay attack detection), the tool must be initiated for 2-way communications:

Tool Initiation:

1. Unload the device drivers:

    ```
    sudo modprobe -r iwldvm iwlwifi mac80211
    ```

    Once the custom firmware is installed, there will no longer be any support for authentication and encryption (WPA2, WEP, etc.), so connections to most access points (APs) will not be possible (and not recommended). If internet connectivity is needed on the device, then it is recommended to use a ethernet connection, not another WiFi NIC. The ethernet connection can be done using a USB adapter if an adapter is lacking on newer laptops.

2. Load the drivers with the logging enabled, and force the transmission mode:

```
sudo modprobe iwlwifi debug=0x40000 connector_log=0x1
echo 0x4101 | sudo tee /sys/Kernel/debug/ieee80211/phy0/iwlwifi\
/iwldvm/debug/monitor_tx_rate
```

Note: 0x4101 sets the HT-SIG symbols (Figure 2.13) in the PLCP. This parameter follows the format shown in Table 4.1:

Table 4.1: HT-SIG Control Parameters

Bit Position	Control Description
B16:	Use the third antenna for transmission
B15:	Use the second antenna for transmission
B14:	Use the first antenna for transmission
B13:	Sets the guard internal; ‘0’ = 0.8 μs, ‘1’ = 0.4 μs
B12:	Duplicate transmission on two 20 MHz channels
B11:	Select the total bandwidth for transmission; ‘0’ = 20 MHz, ‘1’ = 40 MHz
B10:	Preamble selection ‘0’ = Legacy, ‘1’ = Greenfield
B9:	Modulation Type; ‘0’ = OFDM, ‘1’ = CCK
B8:	Transmission Rates; ‘0’ = Legacy, ‘1’ = HT
B7: B6: B5:	Set to 0
B4: B3:	Diversity Selection; ‘00’ = SISO, ‘01’ = MIMO-2, ‘11’ = MIMO-3
B2: B1: B0:	Used for MCS selection

3. Once the drivers are loaded, the NIC needs to put in monitor mode to read arbitrary broadcasted frames:

```
ifconfig wlp4s0 down
iwconfig wlp4s0 mode monitor
ifconfig wlp4s0 up
iw dev wlp4s0 set channel CHANNEL HT20
```

Note: *wlp4s0* is the device ID, and may vary from machine. To find the device ID use the *ifconfig* command. The parameter CHANNEL is the channel in which experimenting is being done. Set this to the channel to be used. Acceptable channels will vary from country as mentioned in subsection 2.3.2. Any other parameter for the NIC, including power mode can be set via *iwconfig*. These parameters should be consistent between communicating parties in this implementation.

4. Once the CSI tool is setup, you can begin logging any received MPDUs to dat file:

```
sudo linux-80211n-csitool-supplementary/netlink/\
log_to_file csi.dat
```

Any logged data will be written to the csi.dat file in the computers user folder.

The only MPDUs that can be logged are 802.11n data units sent in HT mode at the channel being monitored. As well, the MPDUs must have the following MAC addresses to be logged:

```
Address 1 = 00:16:ea:12:34:56
Address 2 = 00:16:ea:12:34:56
```

where addresses 3 and 4 can be any arbitrary address.

Since the CSI tool is working on IEEE 802.11n, the tool saves 30 out of 52 of the CSI estimates from the HT-LTF from a received PPDU at the 20 MHz bandwidth. The subcarriers that are chosen come from the CSI Report Field defined in the IEEE 802.11n standard [23]. Specifically, the subcarriers are: {28,26,24,22,20,18,16,14,12,10,8,6,4,2,1,1,3,5,7,9,11, 13,15,17,19, 21,23,25,27,2}. For the protocol used a single subcarrier is extracted to emulate a non-OFDM system.

4.1.4 Loss of Radio Connectivity

Loss Of Radio CONectivity (LORCON), is an open source network analyzing package. It can be used for the injection of custom MPDU on to a wireless channel. It has built-in capatability with many of the major NIC drivers that are on Linux such as Intel (iwlwifi), Qualcomm Atheros (ath9k), and others. It is possible that other tools and packages (such as Scapy [26]) function with the CSI Tool, but LORCON was chosen due to the recommendations by the creators of the CSI tool. The LORCON application programming interface (API) is in the C-programming language, but a rudimentary implementation exists in the Python programming language. The version of LORCON used can be pulled from the following repository:

```
sudo git clone https://github.com/dhalperi/lorcon-old.git
```

The package can be built with the following commands:

```
./configure
make build
sudo make
```

The format of the LORCON MPDU follows the basic IEEE 802.11n in subsection 2.3.1. Particularly, the MAC Frame Header Format in Figure 2.9. The packet format is defined by a C structure that matches the MPDU. When instantiating the structure, each field represents each of the fields in the header portion of the MPDU.

4.1.5 Library Packet Capture

LORCON provides the capability to inject forged frames, but does not give the option to capture frames. To capture frames from different mediums (ethernet, WiFi), the package Library Packet Capture (libpcap) is used. The API for pcap is also in the C-programming language, so it is a convenient compliment to LORCON. The version of libpcap used can be pulled from the Ubuntu default packages through the terminal:

```
sudo apt-get install libpcap-dev
```

Due to the low-level implementation of libpcap, it is able to filter out received frames at the kernel level through the use of a bandpass filter (BPF). For example, all packets that do not match a given address (e.g. Address 1 = 00:16:ea:12:34:56), can be filtered before being passed up to the OS, saving valuable resources. Since timing is important for reciprocal exchanges, libpcap is an ideal tool for capturing compared to other options due to these efficiencies.

4.1.6 Library Gcrypt

All the cryptographic primitives used in the implementation are from the Library Gcrypt (libgrypt) package [32]. Libgcrypt is reputable C-library, that is most recognized for its use in GNU Privacy Guard (GPG). The primitives used in the implementation include: MAC, hashing, symmetric-key encryption, and pseudorandom generation. Specifically, for the implementation of the relay attack detection, SHA256 is used for hashing and for the HMAC, while the ChaCha20 streamcipher with an 128-bit key is used for the symmetric-key encryption. A detailed reference manual for all the primitives and how to use them is provided [27].

The libgcrypt package also requires the Library GPG Error (libgpg-error [33]) package to handle errors. To install libgpg-error run the following commands in the libgpg-error directory:

```
./configure --prefix=/usr &&
make
sudo make install &&
sudo install -v -m644 -D README /usr/share/doc/libgpg-error-1.36/README
```

Then to install libgcrypt run the following commands in the libgcrypt director:

```
./configure --prefix=/usr &&
make
```

```
sudo make install &&
sudo install -v -dm755   /usr/share/doc/libgcrypt-1.8.4 &&
sudo install -v -m644    README doc/{README.apichanges,fips*,libgcrypt*} \
                    /usr/share/doc/libgcrypt-1.8.4
```

4.2 Preliminary Protocol Implementation

As mentioned, the protocol is implemented utilizing IEEE 802.11n (WiFi), due to its ease of use and accessibility compared to many other wireless communication standards. While WiFi may not be the standard that is typically used for challenge-response PKES, it still has some practicality. Other standards were not used due to the lack of access to CSI, without customizing the standards. This would require significant work compared with WiFi, which has relatively easy access to its CSI. Furthermore, the frequency channels that WiFi operates in overlap with channels utilized by standards such as Bluetooth, which is also used for PKES [24]. This lends credence to WiFi being a decent platform for testing the feasibility of the protocol, as the results are generated in similar conditions to more appropriate standards (i.e., Bluetooth). In this section an overview is given of the implementation of the channel based relay attack detection protocol using WiFi, in both C and MATLAB programming languages.

We can consider the protocol described in Figure 3.1 as the baseline for the implementation. All of the WiFi communications occur on the 64^{th} channel, which has a carrier frequency of 5.320 GHz and a bandwidth of 20 Mhz (5.310 - 5.330 GHz). This frequency was chosen to minimize collisions with other users in the WAN. The transmission speed for the reader and the tag is set to 13 Mbps using QPSK. Both the reader and the tag have a pre-installed 128-bit symmetric-key. All the data for the communications are stored in the MSDU portion of the MPDU instead of an upper layer frame. The protocol can be effectively implemented in three components: The pilot exchange phase, CGC phase, and verification component.

4.2.1 Pilot Exchange

This portion of the protcol is primarily implemented in C using the LORCON, libgcrypt, and libpcap packages. For the protocol to begin, the reader uses libpcap to constantly listen to the channel for a request, with the option for filtering for the MAC address of the tag. The assumption is that the tag will automatically broadcast a request periodically, by injecting frames into the channel using LORCON. Once the reader receives the request over the WiFi channel it will generate and store an 128-bit nonce using libgcrypt, before transmitting it over the channel with an acknowledgement using LORCON. When the tag receives the acknowledgement and the challenge, it will begin the CSI exchange with the reader. When the exchange occurs, the data portion of the MPDU is left blank to minimize the transmission size, which ultimately decreases the time within an exchange. For each pilot $R_p = 36$ bytes, which comes from the MAC frame header (36 Bytes). Though, we should also consider the preamble in the PPDU. In total, the exchange period must occur for atleast 2 seconds, to allow for enough coherence times to occur. The coherence time for an indoor system is typically around 100 ms, which would give roughly 20 coherence times over a 2 second period. This should allow for up to 20 highly uncorrelated measurements. Given this period, the minimum number of packets exchanged to occur to be:

$$\begin{aligned} N &= \frac{13e6 \times 2}{36 \times 8 \times 2} \\ &\approx 4500 \end{aligned} \tag{4.1}$$

It is important to note that due the exponential back-off nature of the CSMA protocol in IEEE 802.11n, that it could take longer than 2 seconds to have a 4500 exchanges. In contrast, a strict 2 second exchange may not have 4500 exchanges. In the worst case each exchange takes up to $2 \times ToP$, which can be significantly larger than 2 seconds. Ideally, the usage of the 5 GHz band should reduce the chance of a collision. Collisions can cause the exchange to exceed the ToP, and the premature ending of the protocol. It also worth noting that different environments could have different coherence times (larger or smaller), and this is a physical limitation.

Since the CSI tool is able to leverage IEEE 802.11n to extract 30 measurements per pilot ($N \times 30$), the extra measurements could be used to increase the key-generation rate by up to a theoretical 30 times. All the results are stored in the csi.dat output file. Due to the coherence time, some of these measurements (and thus quantized bits) can be correlated with one another. To minimize this loss in entropy, the random selection of k-bits described in subsection 3.1.5 should decrease the amount of measurements that correlate in the output. Similarly, if all subcarrier measurements were used, some would correlate within a coherence bandwidth, so a similar random selection would have to be used.

4.2.2 Channel Gain Compliment

Once the exchange is complete, the tag will encrypt its csi.dat records using chacha20 and the symmetric-key, before transmitting to the reader. The reader will then perform CGC in a MATLAB/Octave environment use this CSI.

4.2.3 Verification

Following the transmission of its CSI measurements, the tag performs ASBG in MATLAB/Octave, and writes the output to a binary file that can be read by the C packages. The parameters for ASBG and the ToP can be chosen to meet performance requirements. The resultant quantized information is then used to transmit over the channel with the challenge-response. The HMAC is performed using libgcrypt. When the reader receives the response, it performs ASBG on its modified CSI measurements and also writes the output to a binary file. The reader will use libgcrypt to verify the authenticity before providing the tag with an acknowledgement.

Chapter 5

Preliminary Experimental Results

In this chapter a brief set of experiments and their results are discussed on the preliminary implementation. As a proof of concept design, these experiments provide evidence that this protocol has the potential to detect relay attacks. The experiments will look at some of the pseudorandomness properties from the NIST testing suite. For the preliminary experimentation, two environments are tested under two modes of operations. The environments are a LOS and NLOS setting. The modes of operation are with a stationary and mobile transceiver.

5.1 NIST Randomness Testing

When using a pseudorandom sequence generator for cryptographic purposes, NIST requires a set of 15 tests that must be passed for the sequence to be considered pseudorandom. These tests are documented in detail in the *SP 800-22* test suite document [35]. Other randomness tests exist, such as Golomb's randomness postulates, but the NIST test is considered to be the de-facto test for professional and research purposes. To run the test, an open source GitHub project by *S. Ang* was utilized [36]. The only edit to the code was setting the pattern length (M) to 2 for the entropy test, since only 100 bit sequences are tested.

5.1.1 Experiment

The quantized measurements of the channel are put under evaluation by the randomness test. Only 5 out of the 15 tests were put under experiment. The reason for this is that many of the experiments require a sequence of $\geq 10,000,000$. Due to the coherence time of the channel, the rate to get randomness is extremely limited. In a typical wireless channel, we can expect a coherence time in the hundreds of milliseconds, thus the amount of time necessary to get a 10 Mb sequence would take a significant amount of transmission time at a minimum. For example, With a $T_c = 100\ ms$ and with each measurement needing to occur after a coherence time:

$$T \geq 10000000 \times 100$$

Where T is in units of milliseconds and is the amount of time required to generate $10,000,000$ bits. The minimum amount of time would be 12 *days*. With 5 different experiments it would have taken nearly 7 weeks to get results, while locking up the devices used for the experiments. If the experiment needed to ran again, even more time would be needed.

The experiment was done in a walled work space. The LOS test had the devices several meters away from each other, while the NLOS test was roughly the same distance, but with a wall in between. To simulate a moving channel, the antennas were oscillated back and forth at roughly $\frac{2\lambda}{second}$. For the NLOS results, the ASBG parameters are: $block_size = 10$ and $\alpha = 0.55$. No keys or nonces are used for the experiments to make the bit selections, instead 100 random indexes (bits) are selected to emulate the key and nonce.

5.1.2 Results

The results of the LOS and NLOS experiments can be seen in the tables below:
We notice that all of tests pass, except for the DFT test for the NLOS stable reader test. These preliminary results are promising for showcasing the ability of the protocol at extracting random information from the channel.

Table 5.1: Line-of-Sight NIST Randomness P-values

Test	Tag Stable	Tag Move	Reader Stable	Reader Move
1. Mono-bit:	0.841	0.162	0.110	0.841
2. Runs:	0.686	0.104	0.501	0.160
3. Entropy:	0.200	0.124	0.466	0.655
4. CFS:	0.629	0.144	0.071	0.959
5. CRS:	0.629	0.144	0.071	0.959

Table 5.2: Non-Line-of-Sight NIST Randomness P-values

Test	Tag Stable	Tag Move	Reader Stable	Reader Move
1. Mono-bit:	0.424	0.028	0.317	0.689
2. Runs:	0.459	0.353	0.614	0.701
3. Entropy:	0.704	0.022	0.115	0.737
4. CFS:	0.541	0.025	0.387	0.722
5. CRS:	0.541	0.025	0.387	0.722

Chapter 6

Proposed Attacks and Potential Solutions

In this chapter two active MITM attacks will be introduced. These attacks are not just effective against the proposed relay attack detection protocol, but they should theoretically be able to effect all of the channel based physical-layer protocols observed [18, 19, 20, 21, 22]. The first attack abuses the ordering of the pilot exchanges to filter the channel. The second attack uses an analog relay to creating a single channel, from the two linear channels, despite the existence of a MITM. Fundamentally, both attacks attempt to prevent the channel from maintaining its unrelayability as described in section 2.4. This happens due to the nature of the weakness of the transportability property in a wireless channel.

Two modifications to the relay attack detection protocol are also proposed in this chapter, to prevent each of the attacks respectively.

6.1 Forged Channel Attack

6.1.1 The Attack

In the paper by *Suman Jana et al* [19], an attack that attempts to influences the physical channel is proposed. The example given is an object that can prevent LOS (or cause LOS) in a controlled manner. When LOS is lost the signal strength would decrease, indicating a '0', and when it is gained it would result in a '1'. This object would be controlled by the attacker so a reliable estimate on the quantization can be made. Similar forged channel attacks are proposed for the RSS based relay attack detection [26]. Due to the effect of phase, this phenomenon is not as prominent when looking at the CSI when compared to the RSS.

In the forged channel attack proposed, instead a filter (or an amplifier) is used to influence the channel to either the reader or the tag. When we think about the procedure in most of these physical-layer protocols, the MITM will always receive the pilot signals before the intended party. For example, in a digital relay attack, the first pilot signal that leaves the tag is received at the proxy tag before being forwarded. Since pilot signals are known, the proxy tag could find the channel, H_1, between it and the tag. Similarly, when the reader responds, the proxy tag can discover the channel H_2. Now, at this point, the attacker has both channel states, before both of the legitimate parties. The attacker could simply filter the pilot signal, such that the channel derived by the tag looks like H_2, instead of H_1. As mentioned, this attack is possible due to the a posteriori knowledge that the attacker gains of both channels, that comes from the a priori knowledge of the pilots. A representation of this attack can be seen Figure 6.1:

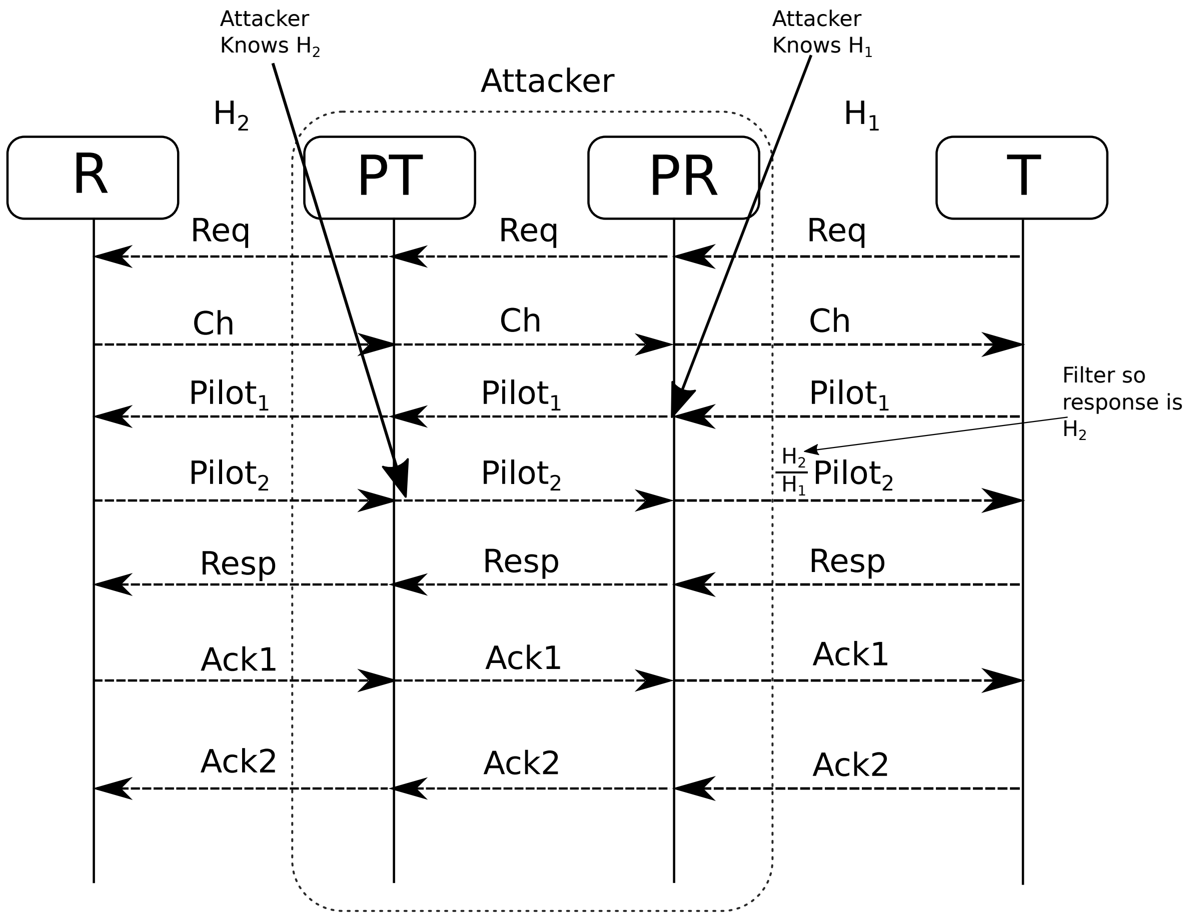


Figure 6.1: Forged Channel Attack in a Digital Relay

From the perspective of the tag, the channel it sees is now H_2, despite the real channel being H_1. The ramifications of the attack is that the quantization now looks the same, despite the existence of the MITM. This would effectively nullify the ability to use CSI to detect the MITM.

This forged channel attack can also be extended to protocols that use RSS. A similar manner to the method described can be used. Once the attacker makes an RSS measurement for each channel, it can simply boost or reduce the signal strength so both channels match.

The attack can also be extended to discover the key for physical-layer based key exchanges. Normally, the existence of a relay would cause the legitimate parties to have different keys. Through the usage of the forged channel attack, both parties would end up having the same key, and this key would be known to the attacker.

6.1.2 Proposed Solution

To prevent the attack, the a priori knowledge must be removed. Typically, a pilot signal, like the HT-LTF and HT-STF used in IEEE 802.11n, is publicly available information. A possible solution is to make the pilot signal a random sequence, and only reveal what the original pilot values are after the exchange is complete. Both the tag and the reader would generate their own random sequence for the pilots that only they would know. By preventing the a priori knowledge of the pilot, the attacker will not be able to estimate the channel, thus being unable to filter the channel. Until the pilots are revealed, the communication parties could store the pilot measurements for future analysis.

This proposed solution is not possible for an RSS based system since there is no pilot, and thus no a priori knowledge. This makes the solution for the RSS detection proposed in other works potentially vulnerable to a forged channel attack [26].

6.2 Linear Analog Relay Attack

6.2.1 The Attack

The focus of the proposed solution in the preceding work was on a digital relay attack. As mentioned in section 2.1, the analog version of the relay attack simply forwards the received signal. In retrospect a digital relay attack demodulates and re-modulates the information. In a analog relay attack we can consider a transmitted signal as through a cascading of linear channels. A representation of this attack can be seen in Figure 6.2:

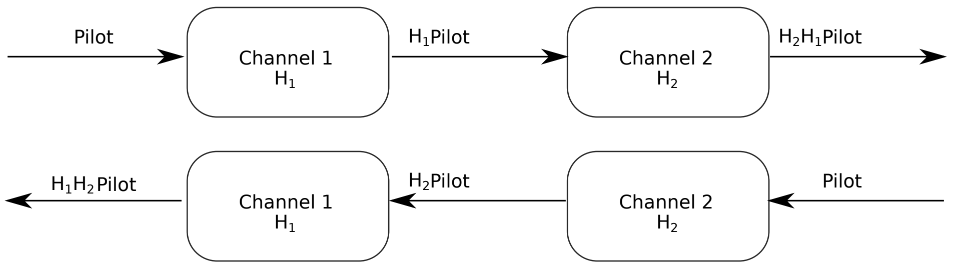


Figure 6.2: Analog Relay Attack

Despite the existence of a MITM, the channel in both direction looks the same due to the symmetry property of linear systems. Under this circumstance the other proposed solutions would not be effective. The attack is not as robust as the digital relay attack; both the required hardware and the signal processing create a high barrier of entry for an attacker.

6.2.2 Proposed Solution

A proposed solution to this problem requires significant signal processing capabilities of at least one legitimate communication party (e.g. the reader or the tag). The solution builds off the mathematics used in the CGC method described in subsection 3.1.4, that was proposed *H Liu et al.* [22]. From Equation 3.2, we may also consider the existence of an attacker:

$$h_{R,E} = h_{p_{R,E}} + n_R + h_{hdw,R}$$
$$h_{E,T} = h_{p_{E,T}}(h_{p_{R,E}} + n_R + h_{hdw,R}) + n_E + h_{hdw,E}$$

$$h_{T,E} = h_{p_{T,E}} + n_T + h_{hdw,T}$$
$$h_{E,R} = h_{p_{E,R}}(h_{p_{T,E}} + n_T + h_{hdw,T}) + n_E + h_{hdw,E}$$

where R, T, and E represent the reader, tag, and attacker respectively. The subscripts represent the direction in which the communications are occurring. Similar to the CGC method, we make the assumption that hardware differences exist, and that some portion of the are a constant additive component. We also make the assumption that there is reciprocity; the same subscripts give an equivalent result. By taking the difference of these channel measurements through CGC:

$$\begin{aligned} h_{E,T} - h_{E,R} &= h_{p_{E,T}} h_{p_{R,E}} + h_{p_{E,T}} h_{hdw,R} + h_{p_{E,T}} n_R + n_E + h_{hdw,E} \\ &\quad - (h_{p_{E,R}} h_{p_{T,E}} + h_{p_{E,R}} h_{hdw,T} + h_{p_{E,R}} n_T + n_E + h_{hdw,E}) \\ &= h_{p_{E,T}} h_{hdw,R} + h_{p_{E,T}} n_R - h_{p_{E,R}} h_{hdw,T} - h_{p_{E,R}} n_T \end{aligned}$$

If we consider that each component is independent, then variance of this result:

$$\begin{aligned} Var(h_{E,T} - h_{E,R}) &= Var(h_{p_{E,T}} h_{hdw,R}) + Var(h_{p_{E,T}} n_R) \\ &\quad - Var(h_{p_{E,R}} h_{hdw,T}) - Var(h_{p_{E,R}} n_T) \end{aligned}$$

We note that if $h_{p_{E,R}} = h_{p_{E,T}}$, there is no attacker and the variance will result in 0 as the number of measurements increases. If there is an MITM in the system, then $h_{p_{E,R}} \neq h_{p_{E,T}}$, and the variance will be some none 0 results.

The limiting factor in this solution is that $h_{hdw,R}, h_{hdw,T}, n_r$, and n_T may have small distributions (i.e., mean and variance), which can make the measurements insignificant. This theoretical solution would require significant signal processing capabilities.

Chapter 7

Conclusions and Future Works

In this thesis, significant work was done in designing and providing the preliminary implementation for a practical solution against the relay attack. While several other solutions have been proposed, many of them are impractical or are vulnerable to attacks introduced in this thesis [10, 11, 14, 20]. Through the usage of open source packages and utilities, an implementation of the protocol was proposed. This implementation shows the practicality of the protocol. Preliminary experiments were also done in these works. The results show that the channel and the quantization method designed, allow for seemingly enough randomness to create an effective block with the channel based relay attack detection protocol.

In regards to future works, the main focus would be on expanding the experimental results and the scope of the implementation for the protocol. There are many different environments and use cases that have not been tested that are of interest. Such environments include crowded areas, open fields, large in-door structures, and an-echoic environments. As well, realistic scenarios, such as with cars and other PKES settings, would help show the practicality of the solution. Modes of operations for consideration include different relative movements of the transceivers, reflectors, and attackers. As a part of these experiments, a more expansive model would also be desired. This includes creating an attacker under the proxy model of the digital relay attack, and having the attacker see if they can defeat the detection protocol. Results on false positive rate would also need to be gathered, which is a crucial statistic for measuring the practical capabilities of the protocol and its implementation. Based on this feedback, it is possible that the protocol and the proposed implementation may need some design changes to better improve performance.

More work also needs to be done to improve the coded implementation in this thesis. The code exists over MATLAB and C as described. This splits the functionality between multiple programs. Ideally a single seamless application would be used. As other aspects of the project are expanded upon and refined, it is also likely the code will change dramatically as well.

The relay attack detection protocol in these works was implemented in IEEE 802.11n, which is not the ideal wireless standard for challenge-response authentication. As mentioned, Bluetooth, NFC, RFID, ZigBee, and other communication standards, would be more apt for preventing a relay attack. Unfortunately, the ability to gather channel state information from these standards is not as straightforward as IEEE 802.11. Most of these standards do not have built in channel estimation capabilities, so custom implemented protocols and devices would have to be utilized. This would require extensive work with software defined radios, or other design capable hardware. If this idea was to be commercialized into an actual product this would be a requirement. Understandably, this work would be of a much larger scope than in these brief designs.

As a separate project, the relay attacks introduced in these papers should be expanded upon and implemented. Experiments on utilizing the attacks on other physical-layer cryptographic protocols can be explored. As well, the solutions proposed should be implemented and tested in conjunction with the proposed attacks, to determine their viability. Both the attacks and their proposed solutions are unique, and could present a new avenue in physical-layer research. It is possible that further work could elaborate on the attacks and possible solutions that are more rigorous than the ones described.

References

[1] GSM Association (2018),*The Mobile Economy 2018*. Accessed on: Apr. 17,2019. [Online]. Available: https://www.gsma.com/mobileeconomy/wp-content/uploads/2018/05/The-Mobile-Economy-2018.pdf

[2] A. Francillon, B. Danev, S. Capkun, Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars, *Cryptology ePrint Archive*, Report 2010/332, 2010.

[3] A. Goldsmith, "Statistical Multipath Channel Models" in *Wireless Communications*, Cambridge University Press, Aug. 2005, ch. 3. Accessed on: Apr. 17, 2019. [Online]. Available: http://web.cs.ucdavis.edu/~liu/289I/Material/book-goldsmith.pdf

[4] A.D. Wyner, "The Wire-Tap Channel", *The Bell System Technical Journal*, vol, 54 , Issue: 8 , 1975

[5] S. Brands, D. Chaum, "Distance-bounding Protocols", *Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology, EUROCRYPT '93*, Lofthus, Norway, 1993.

[6] J.H. Conway, *On Numbers and Games*, CRC Press, Dec. 2000

[7] N. Darchis, "802.11 frames : A starter guide to learn wireless sniffer traces", CISCO, Mar. 3, 2019. Accessed on: May 21, 2019. [Online]. Available: https://community.cisco.com/t5/wireless-mobility-documents/802-11-frames-a-starter-guide-to-learn-wireless-sniffer-traces/ta-p/3110019

[8] R. Nayanajith, “CWAP MAC Header : Frame Control”, MRN-CCIEW, Sep. 27, 2014. Accessed on: May 21, 2019. [Online]. Available: `https://mrncciew.com/2014/09/27/cwap-mac-header-frame-control/`

[9] “IEEE Standard for Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”, 2012.

[10] G.P. Hancke, M.G. Kuhn, “An RFID Distance Bounding Protocol” , *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM’05)*, Athens, Greece, 2005.

[11] Clulow J., Hancke G.P., Kuhn M.G., Moore T., “ So Near and Yet So Far: Distance-Bounding Attacks in Wireless Networks”, *Security and Privacy in Ad-Hoc and Sensor Networks. ESAS 2006*, LNCS vol 4357. Springer, Berlin, Heidelberg, 2006.

[12] Kim, G. , Lee, K. , Kim, S. and Kim, J, “Vehicle Relay Attack Avoidance Methods Using RF Signal Strength”, *Communications and Network*, 5, 573-577 , 2013.

[13] Stajano F., Wong FL., Christianson B. “Multichannel Protocols to Prevent Relay Attacks” *Financial Cryptography and Data Security 2010. LNCS* vol 6052. Springer, Berlin, Heidelberg, 2010.

[14] Choudary O. “Make Noise and Whisper: A Solution to Relay Attacks (Transcript of Discussion)” *Security Protocols XIX. Security Protocols 2011. LNCS*, vol 7114. Springer, Berlin, Heidelberg, 2011.

[15] S. Jana, S. N. Premnath, M. Clark, S. K. Kasera, N. Patwari, S. V. Krishnamurthy, “On the Effectiveness of Secret Key Extraction from Wireless Signal Strength in Real Environments” *MobiCom ’09* , Beijing, China, 2009.

[16] S. Mathur, W. Trappe, N. B. Mandayam, C. Ye, and A. Reznik, “Radio-telepathy: extracting a secret key from anunauthenticated wireless channel” *MobiCom ’08*, San Francisco, California, 2008.

[17] T. Aono, K. Higuchi, T. Ohira, B. Komiyama, and H. Sasaoka, "Wireless secret key generation exploiting reactance-domain scalar response of multipath fading channels " *IEEE Transactions on Antennas and Propagation*, 53(11): 37763784,. 2005.

[18] M. A. Tope and J. C. McEachen, "Unconditionally secure communications over fading channels", *2001 MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force*, McLean, Virginia, USA, 2001.

[19] B. Azimi-Sadjadi, A. Kiayias, A. Mercado, and B. Yener, "Robust key generation from signal envelopes in wireless networks" *InCCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, Alexandria, Virginia, 2007, pp 401410.

S. Jain, J. S. Baras, "Preventing wormhole attacks using physical layer authentication", *WCNC 2012*, Shanghai, China, 2012.

[20] C. T. Zenger, M. Pietersz, C. Paar, "Preventing Relay Attacks and Providing Perfect Forward Secrecy using PHYSEC on 8-bit uC", *IEEE ICC 2016*, Kuala Lumpur, Malaysia, 2016.

[21] H. Liu, Y. Wang, J. Yang, Y. Chen, "Fast and practical secret key extraction by exploiting channel response", *2013 Proceedings IEEE INFOCOM*, Turin, Italy, 2013.

[22] W. Xi, X.-Y. Li, C. Qian, J. Han, S. Tang, J. Zhao, K. Zhao, KEEP: "Fast secret key extraction protocol for D2D communication", *IWQoS*, Hong Kong, China, 2014.

[23] *IEEE Standard for Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput*, 2009.

[24] SM Devices, *Bluetooth Keyless Entry - Keyless Go*, SM Devices. Accessed on: May 21, 2019. [Online]. Available: `http://www.smdevs.com/keylessgo.html`

[25] D. Halpern, W. Hu, A. Sheth, D. Wetherall, "Tool Release: Gathering 802.11n Traces with Channel State Information", *ACM SIGCOMM CCR 2011*, Volume 41: pp 1-53, Jan. 2011.

[26] Scapy, *Packet crafting for Python2 and Python3*, Scapy Project. Accessed on: May 21, 2019. [Online]. Available: `https://scapy.net/`

[27] W. Koch, M. Schulte, The Libgcrypt Reference Manual Ver 1.8.2, GnuPG, Nov. 23, 2018. Accessed on: Apr. 2019. [Online]. Available:`https://www.gnupg.org/documentation/manuals/gcrypt.pdf`

[28] *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, NIST Special Publication 800-22 Rev. 1a, Apr. 2010, April 2010. [Online]. Available: `https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf`

[29] S. Ang, *Python implementation of NIST's A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, GitHub, Jab. 22, 2018. Accessed on: May 5, 2019. Available: `https://github.com/stevenang/randomness_testsuite`

[30] Ubuntu, *Ubuntu 14.04.2 LTS (Trusty Tahr)*, Canonical Ltd, Mar. 7, 2019. Accessed on: May 21, 2019. [Online]. Available:`http://old-releases.ubuntu.com/releases/14.04.1/`

[31] Amazon Canada, *Intel WiFi Link 5300 Wireless LAN Half Size Mini PCI-E*, Amazon Canada. Accessed on: Nov. 21, 2019. [Online]. Available: `https://www.amazon.ca/Intel-Wireless-450Mbps-533AN_HMW-Draft-N1/dp/B0099FLLFK/ref=sr_1_fkmrnull_2?keywords=intel+5300&qid=1555297971&s=gateway&sr=8-2-fkmrnull`

[32] The BLFS Development Team, *libgcrypt-1.8.4*, Beyond Linux From Scratch, May 21 2019. Accessed on: May. 21, 2019. [Online]. Available: `http://www.linuxfromscratch.org/blfs/view/svn/general/libgcrypt.html`

[33] The BLFS Development Team, *libgpg-error-1.36*, Beyond Linux From Scratch, May 21 2019. Accessed on: May. 21, 2019. [Online]. Available:`http://www.linuxfromscratch.org/blfs/view/svn/general/libgpg-error.html`

[34] IEEE Standards Association, *The Structure and Coding of Logical Link Control (LLC) Addresses: A Tutorial Guide*, IEEE Standards Association. Accessed on: April 17, 2019. [Online]. Available: `https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/tutorials/llc.pdf`

[35] Tectronix, *Wi-Fi: Overview of the 802.11 Physical Layer and Transmitter Measurements: WLAN Testing and Analysis (IEEE 802.11)*, Tectronix MDO4000, Mar. 13, 2017. Accessed on: May 12, 2019. [Online]. Available: `https://www.tek.com/document/primer/wi-fi-overview-80211-physical-layer-and-transmitter-measurements`

[36] "*OSI model 802.11 - Data-Link Layer*", Aug. 27, 2017. Accessed on: May 22, 2019. Available: `http://www.netprojnetworks.com/osi-model-802-11/`

[37] M. Gast, "Physical Layer Overvie", in *802.11 Wireless Networks: The Definitive Guide*. O'Reilly, April 2002, ch. 9, pp.158.