

Novel Channel Based Relay Attack Detection Protocols in the Physical-Layer

by

Radi Abubaker

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2024

© Radi Abubaker 2024

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Lifeng Lai Professor, ECE, University of California, Davis
--------------------	---

Supervisor:	Guang Gong Professor, ECE, University of Waterloo
-------------	--

Internal Member:	Patrick Mitran Professor, ECE, University of Waterloo
------------------	--

Internal Member:	Mark Aagaard Professor, ECE, University of Waterloo
------------------	--

Internal-External Member:	Alfred Menezes Professor, C&O, University of Waterloo
---------------------------	--

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Radi Abubaker was the sole author of Chapters 1, 2, 3, 7, and 10, which were written under the supervision of Prof. Guang Gong for this thesis, but were not written for any publication or have not yet been reviewed and edited for future submission.

This thesis consists of work from two papers accepted or published and three manuscripts with the intent on publication. Exceptions to sole authorship of material are as follows:

Chapters 4 and 6 were written and designed by Radi Abubaker, with support, editing, and supervision by Prof. Guang Gong. The software implementation and derivations were done solely by Radi Abubaker, under the supervision of Prof. Guang Gong. These chapters were combined into a single pre-print paper with the target of being submitted for future publication, after revisions.

Citations:

R. Abubaker and G. Gong, “Relay Attack Detection using OFDM Channel-Fingerprinting,” TechRxiv, 2024.

Chapter 5 was written and designed by Radi Abubaker under the supervision of Prof. Guang Gong, except for the key-bit security portion in section 5.2.1, which was written by Prof. Guang Gong. Both Radi Abubaker and Prof. Guang Gong reviewed and edited the others work, with feedback from Prof. Guang Gong used to improve the security of the protocol under active relay attacks. This chapter was written for the target of future publication, after revisions.

Chapter 8 was written and designed by Radi Abubaker, with support, editing, and supervision by Prof. Guang Gong. The software and hardware implementations were done solely by Radi Abubaker. This chapter comes from a paper accepted to a conference, but currently unpublished.

Citations:

R. Abubaker and G. Gong, “Evaluating Channel Based Relay Attack Detection in the near field,” IEEE RFID-TA 2024

In chapter 9, sections 9.4.1 and the introduction in section 9.4 was written for previous publication by Radi Abubaker and Prof. Guang Gong. In particular, the introduction was written by Radi Abubaker, while the Milewski construction was written by Prof. Guang Gong. Both edited and reviewed the work by the other. Section 9.4.2 was mostly written by Prof. Guang Gong, in particular the theorems, lemmas, and their associated proofs were done by Prof. Guang Gong, with review from Radi Abubaker. Section 9.4.3 was

written by Radi Abubaker under the supervision and review by Prof. Guang Gong. The software and hardware implementation for this section was done solely by Radi Abubaker. The rest of chapter 9 was written solely by Radi Abubaker, with the supervision of Prof. Guang Gong for this thesis.

Citations:

R. Abubaker and G. Gong, “Frequency distance sequences for packet detection in physical-layer security,” *Designs, Codes and Cryptography*, 2024, doi:10.1007/s10623-024-01475-x

Abstract

A relay attack is a set of simple, but powerful attacks that can be used to circumvent entity authentication protocols. Through the forwarding of legitimate and real-time messages, relay attacks can cause unexpected authentications to occur, despite the use of modern cryptographic methods, such as digital signatures and message authentication codes. Relay attacks are primarily used by attackers to fraudulently authenticate cyber-physical systems, leading to undesirable responses in the real-world. For example, the relay attack can be used to start vehicles, activate payment systems, and illegally access secured areas. Current solutions are not widely tested and come with trade-offs, which motivates the use of the wireless channel of the physical-layer of communications in wireless systems as a novel prospect for detecting relay attacks.

This thesis details the theory, motivation, design, and implementation of four physical-layer protocols that can individually detect either a decode-and-forward or an amplify-and-forward relay attack, occurring in a wireless challenge-response entity authentication protocol. Each protocol utilizes different aspects of the wireless channel to perform the detection. The first proposed protocol leverages channel reciprocity to detect a decode-and-forward relay attack. The second proposed protocol generalizes detection to non-reciprocal channels to detect decode-and-forward relay attacks under stronger adversarial conditions. The third proposed protocol utilizes the change in distribution of a relayed channel as a feature to detect amplify-and-forward relay attacks. The fourth proposed protocol utilizes a deep neural network to generalize feature selection to detect an amplify-and-forward relay attack. Each protocol builds on work from prior physical-layer research, but introduces novel ideas to handle attacks that have not been well explored.

To evaluate the performance of these protocols, this thesis performs Monte-Carlo simulations, hardware implementations on software defined radios, and theoretical performance evaluations. These results show practically that physical-layer based relay attack detection using the wireless channel is capable of robustly detecting relay attacks, while simultaneously providing ubiquitous high-throughput data communications, which other potential solutions lack.

The thesis is organized into nine chapters. The first three chapters detail the background and motivations of the relay attack. The next five chapters focuses on the protocol designs and their theoretical performance evaluations. The last chapter details the hardware implementation on embedded software-defined radios and the novel processes required to implement the protocols.

Acknowledgements

I would like to thank my family for supporting me over the years while I completed by degrees. Without their understanding and support I would not have been able to achieve what I have done. I would also like to give my sincere gratitude to my supervisor Prof. Guang Gong for giving guidance and support over the last 8 years in academia. The experience and opportunity to learn in this enriching environment would not have been possible without her support. Thanks is also given to my colleagues in the Communication Security Lab at the University of Waterloo, who have provided me moral support. Great appreciation is also given to the committee members, Prof. Patrick Mitran, Prof. Alfred Menezes, and Prof. Mark Aagaard, who have graciously volunteered their time over the previous years to help me achieve my PhD milestones. Thanks must also be given to Prof. Slim Boumaiza and Prof. Catherine Gebotys, who were previously involved with my PhD milestones, and the external examiner Prof. Lifeng Lai, who has volunteered to be a part of the committee.

Table of Contents

Examining Committee Membership	ii
Author's Declaration	iii
Statement of Contributions	iv
Abstract	vi
Acknowledgements	vii
List of Tables	xiii
List of Figures	xv
1 Introduction	1
2 Background	9
2.1 Challenge-Response Protocol	11
2.2 The Relay Attack	14
2.3 Wireless Communications	21
2.3.1 Basic Communications	21
2.3.2 Fading Channel Effects in far field Communications	23
2.3.3 Orthogonal Frequency Division Multiplexing	29

2.4	Channel Reciprocity	32
2.4.1	Channel Non-Reciprocity	33
2.5	Software Defined Radios	34
3	Literature Review	36
3.1	Relay Attack	36
3.2	Distance-Bounding	38
3.3	Contextualization	43
3.4	Physical-Layer Security	45
3.5	Motivation	52
4	One-Way Decode-and-Forward Relay Detection Protocol under Passive Attacks using Channel Reciprocity	54
4.1	Protocol Details	55
4.1.1	Step 1: Initiation	55
4.1.2	Step 2: Channel Probing through a Pilot Exchange	56
4.1.3	Step 3: Channel Feedback	57
4.1.4	Step 4: Decode-and-Forward Relay Detection using the Cross-Correlation Evaluation	58
4.1.5	Step 5: Challenge-Response Authentication	61
4.2	Feedback Size and Complexity	62
4.3	Simulation Experimentation	63
4.4	Active Relay Attack	66
5	One-Way Decode-and-Forward Relay Detection Protocol under Active and Passive Attacks	68
5.1	Protocol Details	69
5.1.1	Step 1: Initiation	73
5.1.2	Step 2: Nonce Transmission	73

5.1.3	Step 3: Channel Feedback	75
5.1.4	Step 4: Relay Challenge	75
5.1.5	Step 5: Prover Generation	77
5.1.6	Step 6: Decode-and-Forward Relay Detection with Authentication .	77
5.2	Feedback Size and Complexity	78
5.2.1	Filter Generation and Security	78
5.2.2	Simulation Experimentation	79
5.2.3	Hardware Experimentation	83
6	One-Way Amplify-and-Forward Relay Detection Protocol under Passive Attacks using Channel Distribution	88
6.1	Protocol Details	89
6.1.1	Step 1: Initiation	91
6.1.2	Step 2: Channel Probing through a Pilot Exchange	91
6.1.3	Step 3: Channel Feedback	92
6.1.4	Step 4: Amplify-and-Forward Relay Detection	93
6.1.5	Step 5: Challenge-Response Authentication	99
6.2	Feedback Size and Complexity	99
6.3	Simulation Experimentation	100
7	One-Way Amplify-and-Forward Relay Detection Protocol under Active and Passive Attacks using a Deep Neural Network	103
7.1	Protocol Details	104
7.1.1	Step 1: Initiation	106
7.1.2	Step 2: Nonce Transmission	106
7.1.3	Step 3: Channel Feedback	106
7.1.4	Step 4: Amplify-and-Forward Relay Attack Detection using a DNN	107
7.2	Simulation Experimentation	108

8	Near Field Relay Attack Detection	111
8.1	Near Field Channel Model	112
8.2	Modified One-Way Decode-and-Forward Relay Detection Protocol for Pas- sive Near Field Detection	114
8.2.1	Step 1: Initiation	115
8.2.2	Step 2: Nonce Transmission	116
8.2.3	Step 3: Channel Feedback	116
8.2.4	Step 4: Relay Challenge	117
8.2.5	Step 5: Prover Generation	118
8.2.6	Step 6: Decode-and-Forward Relay Detection with Authentication .	118
8.2.7	Security Analysis	119
8.3	Hardware Experimentation	119
8.3.1	False-Positive Experiment	120
8.3.2	False-Negative Experiment	121
9	Hardware Implementation	123
9.1	Motivation	123
9.2	Pluto SDR	124
9.2.1	Industrial Input/Ouput	126
9.2.2	Communication Setup	131
9.3	OFDM System	134
9.4	Pilot Selection: Construction of Frequency Distance Sequence Sets	136
9.4.1	Pilot Sequence Properties	137
9.4.2	Representation and correlation of frequency distance sequence sets .	140
9.4.3	Pilot detection sequence application	148
9.4.4	Channel Feedback	150
9.5	Timing Synchronization	151
9.5.1	Coarse Timing Synchronization	156

9.5.2	Fine Timing Synchronization	168
9.6	Carrier Frequency Offset Elimination	171
9.7	Secure Waterfilling and Adaptive Modulation	174
9.8	Relay Attacks	185
9.8.1	Amplify-and-Forward Relay Attack	187
9.8.2	Relay Waterfilling	188
10	Conclusion and Future Work	189
10.1	Conclusion	189
10.2	Future Work	191
	References	193

List of Tables

2.1	Steps of the Decode-and-Forward Relay Attack on a Challenge-Response Authentication Protocol	18
4.1	The Steps and their Descriptions of the Protocol	56
4.2	DF Simulation parameters	64
5.1	Steps of the Channel Based Relay Attack Detection Protocol	71
5.2	DF Simulation parameters for Protocol 2 with OFDM-QPSK transmissions	82
5.3	Hardware experiments parameters for Protocol 2 with OFDM transmissions	84
6.1	Phases and their Descriptions in Protocol 3	90
6.2	AF Simulation parameters [128]	100
7.1	Steps of the Channel Based Relay Attack Detection Protocol	105
8.1	Steps of the Channel Based Relay Attack Detection Protocol [158]	116
9.1	Major processing components in the Analog Devices ADALM-Pluto SDR .	125
9.2	Specifications of radio transceiver on the ADALM-Pluto SDR [122]	126
9.3	Major processing components in the Analog Devices ADALM-Pluto SDR .	131
9.4	Correlation properties of (ℓ, q) -FDSS	147
9.5	Seed sequences	148
9.6	PAPR of the interpolated sequences	149

9.7	Steps of the timing synchronization algorithm	156
9.8	Steps of the coarse timing synchronization algorithm	160
9.9	Experiment parameters for the SDRs	166
9.10	Steps of the timing synchronization algorithm	169
9.11	Steps of the Waterfilling Protocol	179

List of Figures

1.1	Open System Interconnection main layers of communications	3
1.2	Thieves stealing cars using the relay attack [146, 147, 148, 149, 150, 151] .	4
2.1	Uni-directional symmetric-key challenge-response entity authentication protocol	12
2.2	Uni-directional public-key challenge-response entity authentication protocol	13
2.3	Mafia-fraud relay attack on a challenge-response authentication protocol [69]	16
2.4	Decode-and-forward mafia-fraud relay attack	17
2.5	Decode-and-forward mafia-fraud step 2 in detail	18
2.6	Amplify-and-forward mafia-fraud relay attack through a single relay	19
2.7	Active attacker in a decode-and-forward mafia-fraud relay attack through a single relay	21
2.8	Symbols before and after multiplexing	30
2.9	OFDM Transmitter and Receiver	31
2.10	A 3-Device Half-Duplex System with a Reciprocal Exchange [69]	33
2.11	Typical layout of a Software Defined Radio	34
4.1	Challenge-Response Authentication	62
4.2	false-positive rate for detecting DF relay attacks versus CNR	65
4.3	false-negative rate for detecting DF relay attacks versus CNR	66
5.1	Protocol with no relaying	70

5.2	Protocol with with a decode-and-forward relay	71
5.3	Graphical summary of the steps of the proposed protocol	72
5.4	Step 2: with (bottom) and without (top) a decode-and-forward relay attack	74
5.5	Step 4: with (bottom) and without (top) a decode-and-forward relay attack	76
5.6	FPR versus SNR simulation results in a NLOS channel with small and significant frequency selectivity	80
5.7	FPR versus SNR simulation results SNR in a LOS channel with small and significant frequency selectivity	81
5.8	FNR versus SNR simulation results in a NLOS channel with small and significant frequency selectivity	82
5.9	FNR versus SNR simulation results in a LOS channel with small and significant frequency selectivity	83
5.10	FPR versus SNR hardware results in a LOS channel with small and significant frequency selectivity [128]	86
5.11	FNR versus SNR hardware results in a LOS channel with small and significant frequency selectivity [128]	87
6.1	General One-way Challenge-Response Authentication Protocol [128]	91
6.2	K amplify-and-forward relay nodes with K uncorrelated channels, resulting in a final channel that is the multiplication of each CFR	94
6.3	The mapping of correlated time-frequency channel to an uncorrelated channel through decimation [128]	98
6.4	False-positive rate for detecting amplify-and-forward relay attack versus CNR [128]	101
6.5	false-negative rate for detecting amplify-and-forward relay attack versus CNR	102
7.1	Basic visualization of the neural network used in Protocol 4	107
7.2	False-Positive rate for detecting amplify-and-forward relay attack versus CNR	109
7.3	False-Negative rate for detecting amplify-and-forward relay attack versus CNR	110
8.1	Example of near field challenge-response authentication with a HF RFID credit card and a payment terminal	112

8.2	Message exchange of the Channel Based Relay Attack Detection Protocol [158]	115
8.3	step 4: Without a decode-and-forward relay attack (top) and under one (bottom) [158]	117
8.4	Experiment setup [158]	120
8.5	False-Positive Rate versus SNR at various distances between the prover and verifier [158]	121
8.6	False-Negative Rate versus SNR in the near field and far field under a relay attack [158]	122
9.1	Pluto SDR structure with IIO	127
9.2	Pluto SDR structure with IIO	128
9.3	The throughput measured through a TCP/IP packet exchange over IIO on a Pluto SDR and USB 2.0 on a Linux remote computer	128
9.4	Example of the receiver buffer queue on the Pluto SDR	129
9.5	Example of the receiver buffer queue on the Pluto SDR [123]	130
9.6	The throughput measured through a TCP/IP packet exchange over IIO on a Libre SDR and Gigabit Ethernet on a Linux remote computer	132
9.7	SDR setup to control and power remotely	133
9.8	SDR setup with physical devices	133
9.9	OFDM Transmitter Chain	134
9.10	OFDM Receiver Chain	135
9.11	OFDM Transmission Structure	135
9.12	Aperiodic Autocorrelation Functions of the FDSS interleaved sequence with a Milewski Construction with different seeds (1) FZC (2) BPSK-GCP (3) QPSK-GCP [123]	149
9.13	Intra-sampling timing error when sampling an analog signal from an original discrete signal	151
9.14	STO offset comparing 4 QPSK symbols received on neighbouring subcarriers (20 QPSK total) in a single OFDM symbols between 2 Pluto SDRs in LOS	153

9.15	State Machine for the timing synchronization algorithm utilized on the embedded SDR for protocol implementation	155
9.16	General synchronization process and which property of the signal is important for the process [123]	158
9.17	Running time in μs versus sample size of a single buffer in the queue with a sampling rate $f_s = 20$ MSPS, decimation rate = 30 [123]	159
9.18	Example of a received transmission on a Libre SDR, where the portion of the received signal could be in 2 blocks	163
9.19	False-Negative Rates versus SNR for different FDSS interleaved sequences using various seed sequences: (1) FZC (2) BPSK-GCP (3) QPSK-GCP (4) OFDM QPSK [123]	165
9.20	Analog Devices ADALM-Pluto SDRs connected with an SMA cable [123] .	167
9.21	FDSS interleaved sequence comparison from coarse synchronization experiment on SDRs [123]	168
9.22	CFO comparing 4 QPSK symbols received on 4 subcarriers over 5 different OFDM symbols between 2 Pluto SDRs in LOS	173
9.23	Neighbouring Moose's estimate on neighbouring pilots	174
9.24	QPSK symbols in time after removal of CFO in first 5 symbols	174
9.25	Example of how power levels are allocated for $M = 4$ subcarriers and an initial $CNR_{\min} = 10$ dB	182
9.26	Example of relay interference between legitimate communication parties in a room	186

Chapter 1

Introduction

As hardware has become cheaper and more capable over time, wireless communications have proliferated in our daily lives. Movements like the Internet-of-Things (IoT) look to place low powered wireless devices all over our homes, public spaces, and industries under the doctrines of convenience, efficiency, and economic development. High throughput wireless communications are already ubiquitous in our lives, allowing for massive media consumption. IEEE 802.11 (WiFi) exists in almost every home in Canada, while Long Term Evolution (LTE) and 5th Generation (5G) cellular services are just as common, with 6th Generation (6G) services already on the horizon. Beyond this, technologies like Bluetooth Low Energy (BLE), Radio Frequency Identification (RFID) and its relative, near field communications (NFC), have already been highly adopted by consumers all over the world as communication systems used for authentication. These technologies have provided new conveniences in areas like payment services, that now only require tapping or walking to automatically authenticate. Consumers are also currently transitioning to payments through their mobile devices through Apple Pay and Google Pay using NFC. RFID is also prominent for controlling physical accessibility, particularly for private environments like businesses, labs, and homes. Even automobiles have been increasingly adopting wireless communications through the usage of vehicular networks, location services, and cellular systems. Vehicles also use various wireless systems, like BLE, to control accessibility. One such method is known as Passive Keyless Entry Systems (PKES), which locks or unlocks vehicles when the owner walks away or towards it, respectively.

With all this information, ranging from menial to critically important, it is crucial that all types of consumers are confident and certain in the security of these wirelessly communicating devices. By the nature of electromagnetic waves, wireless communications occurs on a broadcast channel, for which anyone can in principle listen in on. This

allows for collecting, manipulating, jamming, or relaying of all the signals being transmitted. The wireless communications devices that are ubiquitous in our lives utilize mass produced components and standardized software, like network interface cards (NICs) and open source firmware, which can easily be modified. Determined individuals can use and share software to interact with the broadcasted information, for which nefarious actors can utilize, without any specific knowledge of their own. To protect against these potential adversaries, security is generally provided through doctrines of confidentiality, integrity, authentication, accessibility, and non-repudiation. Mathematical and hardware advancements have provided many of these requirements over the last several decades, allowing the broadcasted information to stay secure in these potentially adversarial environments. Symmetric and public key encryption provide confidentiality. Digital signatures and message authentication codes provide integrity, authenticity, and non-repudiation (for signatures). Accessibility has been harder to manage, but generally revolves around resource contention and management (power, bandwidth, time, etc.) between malicious and legitimate parties. When considering the five layer Open System Interconnection (OSI) model of communications seen in Figure 1.1, almost all of the security requirements mentioned are provided digitally through the upper layers of communications. Only jamming, which denies accessibility, is managed at the physical-layer, which also happens to be where most wireless communication systems are implemented. Upper layers, above the physical-layer, attempt to strip physical phenomenon in a wireless system, such as delay, noise, fading, Doppler shifts, and other effects. At best, physical features are used as a source of randomness (seeds) or stability (biometric authentication), which are still only used by the upper layers of communication for cryptographic primitives. The physical-layer has generally been neglected for practical usages in providing most of the security requirements. Over the last 30 years, as wireless communications proliferated, research work has expanded the theoretical usages of security at the physical-layer for key generation, encryption, and authentication, but there has not been much push in industry due to the perceived lack of necessity over current methodologies and the lack of detailed security analysis, especially when compared to the vast works in the established cryptography field.

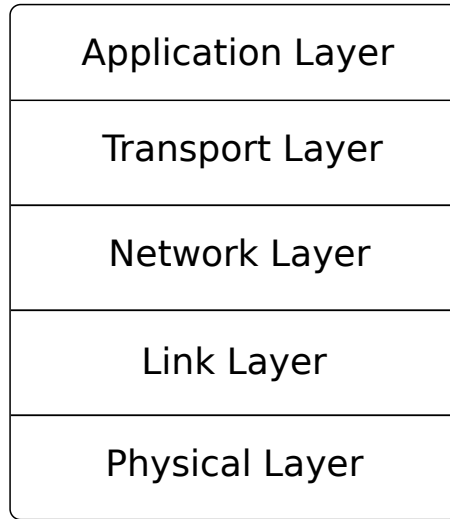


Figure 1.1: Open System Interconnection main layers of communications

Over the recent years, an attack colloquially known as the mafia-fraud relay attack has increasingly become a problem. At the most basic level, the relay attack is a forwarding of communications between two legitimate parties by a man-in-the-middle (MitM), in which the attacker is the one doing the forwarding. A mafia-fraud relay attack is a specific type of relay attack, in which neither legitimate party is aware that the attack is occurring. The goal of the attack is to cause communications to occur, when they otherwise should not be. Usually, this is used as a range extension, where a system designed for communications at one distance is increased beyond what would be expected, through the attacker's relay. The relay attack works best with wireless systems, as communications can be done without a direct physical connection, but it can still occur with wired systems. One of the most concerning examples of a relay attack is with RFID payment systems. These systems typically only work in the ranges of less than 1 m, but a relay attack could extend this to be dozens of metres or even kilometres, depending on how the attack is performed, and what the communication standard is. Because the relaying can happen in verbatim, from the perspective of the legitimate parties they may as well be talking directly to each other. While this typically should not be a problem for most communications, as the doctrines of security are maintained, it becomes an issue when dealing with entity authentication protocols that rely on wireless communications for interacting with cyber-physical systems. With the example of the RFID payment, an attacker could use a credit card to complete a payment, without the owner's consent or knowledge to illegally purchase a product inside a store. Similar to wireless payment systems, relay attacks can be used to get around other

wireless challenge-response authentication security systems including doors, safes, and cars. Relay attacks can also be performed on location services, tricking a system into thinking the devices are somewhere they are not. Any situation where the authentication works passively (without user input) through wireless communications is particularly vulnerable to a relay attack. Relay attacks can even be used to provide plausible deniability. For example, if a phone indicates one location, but a relay attack occurs to provide payment in another location, the attacker could claim that the payment was fraudulent to get a refund, despite actually purchasing the product.

The mafia-fraud relay attack is already being used by criminals to steal vehicles from their owners. Vehicles utilizing PKES to authenticate access have become common as a convenience for owners. These systems operate automatically to unlock or lock as the owner approaches or leaves, respectively, which allows for relay attacks to occur. While it is difficult to put a specific price on the damages, in 2023 at least 32 cars were reported stolen in Toronto every day alone, with the relay attack being cited as one of the primary methods used by thieves to break in to and start the vehicles [90, 91, 92]. Major insurance companies even warn owners to protect their vehicles against the relay attack through non-technological methods (e.g., steering wheel locks) [93]. A. Francillion *et al.* have shown that at least 10 different vehicle models from 8 different manufacturers are vulnerable such a relay attack, allowing unauthorized unlocking and starting of the vehicles [10]. A collage of thieves stealing cars using relay attacks can be seen in Figure 1.2.

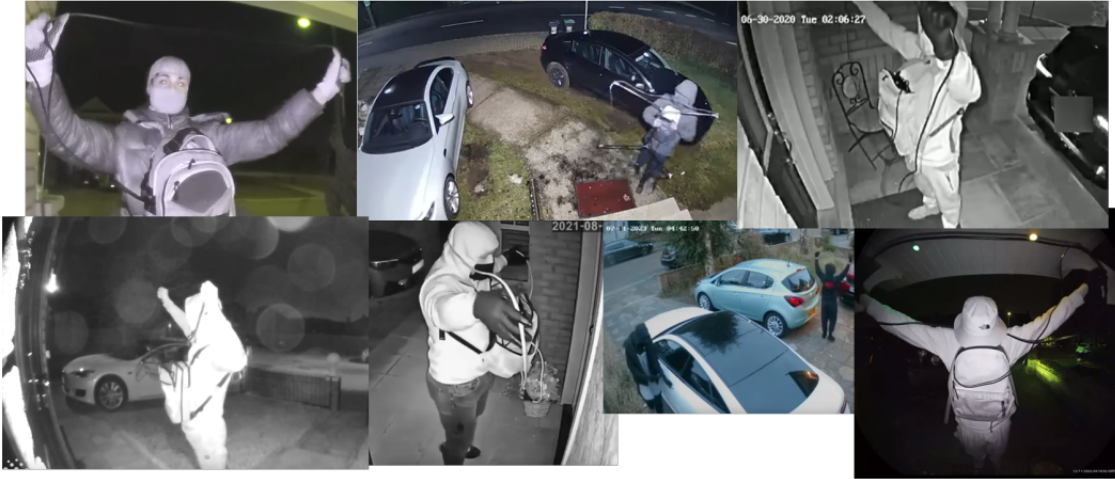


Figure 1.2: Thieves stealing cars using the relay attack [146, 147, 148, 149, 150, 151]

The typical methods that provide authentication at the upper-layers of the OSI model

fail to handle the relay attack. All these methods do is verify mathematical equations, which on their own cannot discern a notion of distance or legitimate physical device. As long as these protocols are presented with the correct verifiable knowledge, authentication occurs. The legitimate parties with private and public keys are the ones generating the messages and communicating with each other, albeit not in the situation expected. There is nothing unbecoming of these communications from the perspective of information. One easy countermeasure to the relay attack is to simply ensure that users have to verify themselves through some physical means, before any authentication occurs. This is exactly what Apple and Google use for their NFC payment systems. User must unlock their phones before the transaction can occur. Some other simple solutions include keeping devices in Faraday cages, to limit their abilities to communicate. One of the major trade-offs with these solutions is the loss in convenience, as the user must directly intervene before any interaction can occur. Furthermore, the practicality is limited, as not all systems can be interacted with. For example, low-powered IoT and RFID devices may not have any input options. Other devices may be physically inaccessible in remote locations, may operate autonomously, or are embedded inside another device. Regardless, the expectation that user input is limited or impossible is common for wireless devices.

In this thesis, the physical-layer is identified and proposed as a solution for detecting mafia-fraud relay attacks, through the observation of the wireless channel. At the moment, the physical-layer does not stand out for most security requirements, but it provides a unique potential to deal with the relay attack that cannot be dealt with at the higher levels of communication. Physical phenomenon like time delays caused by traveling electromagnetic waves, stochastic effects such as noise, multipaths, Dopplers effect, and environmental conditions, can be utilized to identify not only the communication channel, but the state of the channel that exist between nodes. By identifying features of interest and how these features in a received signal change under a relay attack, it is possible to determine if a relay attack is occurring. When considering the different capabilities of the attacker, there are several types of relay attacks, including: decode-and-forward, amplify-and-forward, passive, and active relay attacks. Furthermore, a wireless relay attack can happen in the near field or far field. An ideal solution to the relay attack that utilizes the physical-layer should be able to detect when any of these type of attacks are occurring in the different conditions, while providing challenge-response authentication.

In this thesis, four novel protocols that perform symmetric-key challenge-response entity authentication, while detecting mafia-fraud relay attacks using the physical-layer, are proposed in detail. Each protocol is individually capable of detecting either a decode-and-forward or amplify-and-forward relay attacks, but not both simultaneously. By combining these protocols in series, both decode-and-forward and amplify-and-forward relay attacks

will be able to be detected. The first proposed protocol leverages channel reciprocity to detect a decode-and-forward relay attack under a passive attacker, through the consideration of the cross-correlation of the estimated channel. If the legitimate devices are talking directly to each other the result will be above an identified threshold, otherwise it will be below. The second proposed protocol generalizes detection to non-reciprocal channels to detect decode-and-forward relay attacks under stronger adversarial conditions, allowing for the detection of passive and active relay attacks. The protocol leverages a pre-shared symmetric-key to provide secure feedback to allow for this non-reciprocal detection. The third proposed protocol leverages the change in distribution of a relayed channel as a feature to detect amplify-and-forward relay attacks, under passive conditions. From the central limit theorem, a single wireless channel can be modeled as a Gaussian random process, which changes as the amplification and relay occurs. The protocol uses a hypothesis test on normality to make a determination for amplify-and-forward detection. The fourth proposed protocol leverages the same ideas (i.e., that features of the channel change as they are amplified and forwarded), but generalizes detection through the utilization of a deep neural network. The neural network focuses on a simple binary classification problem of the input channel estimation. By utilizing the secret key, the protocol is also capable of working under active attacker conditions. Furthermore, by combining the selection of the first two protocols with the second two, in series, it is possible to detect both decode-and-forward and amplify-and-forward relay attacks.

At the base of each of the proposed protocols is the same orthogonal frequency division multiplexing (OFDM) communication system, which provides data communication and channel estimation through various pilot symbol exchanges. Through the pilot exchange, each party is able to estimate the wireless channel, resulting in an estimate of the channel frequency response and channel impulse response. Using these estimates, each protocol is able to perform a form of relay detection.

To evaluate the practicality and performance of the protocols proposed, simulations and hardware experimentations are done, with a focus on using Monte-Carlo results to find the false-positive and false-negative rates, when performing authentication under direct communications and under a relay attack, respectively. Experimentation is done under various channel conditions to gather these results and explore the practical implications of the protocol implementations. For example, considerations such the throughput, number of exchanged messages, channel conditions, in which the protocols are effective is explored. Detailed design and analysis is done on the hardware implementation, with a focus on the novel protocols and algorithms on embedded software-defined radios.

The specific contributions of the thesis are the following:

1. An overview of the different forms of relay attacks and the general approaches to deal with them,
2. Proposal of four novel channel based relay detections protocols for detecting when relay attacks occur considering under decode-and-forward, amplify-and-forward, passive, and active relay attacks,
3. Experimentation that shows the viability of channel based relay detection for near field relay detection,
4. Detailing of the first open-source hardware implementation of amplify-and-forward relay attacks and decode-and-forward relay attacks on software-defined radio,
5. The first hardware implementation of relay attack detection on software-defined protocols using the wireless channel,
6. Software simulations of the proposed protocols in various channel conditions,
7. Experimental evaluations of the proposed relay attacks, showing how they perform in software and hardware,
8. Theoretical evaluations of the proposed protocols, showing how they provide security, and how the security may change under different channel conditions.

The thesis is organized into 10 chapters, including the Introduction in Chapter 1 and ending with the Conclusion in Chapter 10. Chapter 2 details the general background theory in communications (e.g., OFDM, software-defined radios, etc.) and the mathematical notations that are prominent throughout the thesis. Chapter 2 also defines the relay attack problem in detail, including the amplify-and-forward, decode-and-forward, passive, and active attacks, which form the basis of the problems addressed in this thesis. Chapter 3 is a literature review of related works in research. The focus of the literature survey will be on the consequences of the relay attack, the current solutions for the relay attack, and the motivations for the proposed solutions that utilize the physical-layer, and particularly the wireless channel. Chapters 4, 5, 6, and 7 focus on each of the four novel protocols, respectively. The chapters detail the steps in each of the protocols, how security is provided, and any complexity to be analyzed. Chapter 8 looks at using the channel based relay detection in the near field. Chapter 9 focuses on the implementation of the protocols and an OFDM system on an embedded software-defined radio. A significant focus will be given towards detailing the general hardware implementation required for the OFDM communication platform on the software-defined radios. This chapter only has the specific

implementation of the second and fourth protocols. Chapter 10 is a conclusion of the work presented in the thesis, including specific future work and general ideas that could be explored.

Chapter 2

Background

This chapter details the background materials and theories that are fundamental to the protocols being proposed to detect relay attacks. It also establishes the notation that will be used throughout the thesis. These theories include a broad overview of the relay attack, particularly all the situations and types that can occur, and why the distinction matters. This will lead to the problem statement, which this thesis will look to solve. A quick overview of symmetric-key methods for challenge-response entity authentication protocols is also discussed, which is the main method for cyber-physical authentications, in which the relay attacks are most consequential. An overview of signals and wireless communications is also given, so the ideas later presented are clear.

Some basic formatting, functions, and signal processing representation are defined that will be used and referenced throughout the thesis:

- $\mathbb{C}$, the complex field, $\mathbb{R}$, the real number field, and $\mathbb{C}^{n \times m}$, the set consisting all $n \times m$ matrices over $\mathbb{C}$, and $\mathbb{C}^n$ a set of all vectors of dimension n over $\mathbb{C}$,
- For an n by m matrix, say $\mathbf{H}$, $\mathbf{H}^T$ represents its transpose and $\mathbf{H}^H$ is the Hermitian,
- Capital bolded letters are used to represent matrices and bolded lowercase letters are used to represent vectors,
- Given a vector $\mathbf{x} \in \mathbb{C}^N$, this vector can be interpreted as a column vector with elements given by the form $\mathbf{x} = (x[0], x[1], \dots, x[N-1])$, where each element $x[n] \in \mathbb{C}$,
- By default, every vector can be assumed to be a complex vector (i.e., contains complex elements),

- A set of N column vectors, each of size $M \times 1$ can be concatenated to be a $M \times N$ matrix by $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{M-1}]$, where $\mathbf{x}_n = (x_n[0], x_n[1], \dots, x_n[N-1])$,
- Given a matrix $\mathbf{X}$, consisting of vectors $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{N-1}]$, each element in the matrix can be represented as

$$X = \begin{pmatrix} x_0[0] & x_1[0] & \cdots & x_{M-1}[0] \\ x_0[1] & x_1[1] & \cdots & x_{M-1}[1] \\ \vdots & \vdots & \ddots & \vdots \\ x_0[N-1] & x_1[N-1] & \cdots & x_{M-1}[N-1] \end{pmatrix}$$

- If a symbol contains N samples over a sampling period of T_s , the resulting transmitted and received symbols can be represented by the vectors $\mathbf{s} = (s[0], s[1], \dots, s[N-1])$ and $\mathbf{r} = (r[0], r[1], \dots, r[N-1])$ respectively, where each element in the index is a sample and the symbol duration is NT_s seconds,
- $D\{\cdot\}, \mathbb{C}^N \rightarrow \mathbb{C}^{N \times N}$, diagonalizes a vector into a diagonal matrix, where the diagonal of the matrix is the vector and the upper and lower triangles are all 0,
- Given 2 vectors of the same size N , $\mathbf{x} = (x[0], x[1], \dots, x[N-1])$ and $\mathbf{y} = (y[0], y[1], \dots, y[N-1])$, the element wise multiplication of the vector can be given by

$$D\{\mathbf{x}\}\mathbf{y} = (x[0]y[0], x[1]y[1], \dots, x[N-1]y[N-1]),$$

- $[\cdot, \cdot], < \mathbb{C}^N, \mathbb{C}^N > \rightarrow \mathbb{C}^{N \times 2}$, combines two vectors as columns in a matrix,
- $\cdot\|\cdot, < \mathbb{C}^{N/2}, \mathbb{C}^{N/2} > \rightarrow \mathbb{C}^N$, concatenates two vectors into a single vector,
- $F\{\cdot\}, \mathbb{C}^N \rightarrow \mathbb{C}^N$, is the Discrete Fourier Transform (DFT),
- $F^{-1}\{\cdot\}, \mathbb{C}^N \rightarrow \mathbb{C}^N$, is the Inverse Discrete Fourier Transform (IDFT),
- By default, assume all vectors are the frequency domain unless otherwise specified,
- Filters will be represented as exclusively as FIR filter's frequency response $\mathbf{h}$,
- All wireless channels will be represented as a filter, where the channels are assumed to be shorter than the input signal lengths, with trailing zeros to infinite. This allows the wireless channel filters to be approximated as FIR filters, where the cyclic-convolution and convolution are equivalent,

- Filtering the signal $\mathbf{x}$ with $\mathbf{h}$ will be represented by the element wise multiplication $\mathbf{y} = \mathbf{D}\{\mathbf{h}\}\mathbf{x}$.

2.1 Challenge-Response Protocol

Entity authentication allows one or more entities to assure another party that they match the identities claimed. The authentication can provide device authentication, proving that the entities must be some specific devices, or it can be user authentication, where they prove they are some entity, regardless of the device being used for the communications. To cryptographically provide entity authentication, a challenge-response protocol is used [95]. The challenge-response works by having a verifier produce a random value (nonce) that acts as a challenge. The prover generates a response to this challenge, that a verifier can confirm could only be generated by the prover and not some other party trying to pretend to be the prover, thereby verifying their identity. The response can either be in the form of a message authentication code (MAC) or a digital signature, which would be generated by a symmetric-key or public and private key pair, respectively. Verification can occur in both directions (bi or single direction), where both or one of the communicating party proves or verifies the identities of the other. In both cases, it is assumed that any symmetric or public key is properly tied to the prover. If the challenge is generated from a large and uniform random source, then the verifier can be confident that the prover is who they say they are, as it would be unlikely that an attacker could generate or find prior response that satisfies the challenge, as long as the MAC or digital signature processes selected are cryptographically secure. The first likely (though not proven) usage of the challenge-response authentication protocol was for identifying warplanes in World War 2 [97]. Since the planes were travelling at increasingly high speeds and radar systems were able to identify planes at further and further distances, authentication through voice was no longer viable to prevent friendly-fire incidences. Identification, friend or foe (IFF) systems had to be invented, in which today challenge-response is the primary cryptographic protocols which provides the entity authentication.

For the case of symmetric-key challenge-response with a MAC, the key must be pre-shared prior to authentication. This could happen through a key exchange protocol, such as authenticated Diffie-Hellman exchange (DHE) or Rivest-Shamir-Adleman (RSA). This method is common for user authentication, but can also be used for device authentication, when the prover and verifier devices first communicate. For example, Bluetooth devices share personal identification numbers (PINs) when they are first pairing, which provides device authenticity and is used for future authentication. Symmetric-key sharing can also

occur through a direct installation, which is typical for cars, key fobs, and RFID systems used for device authentication. Once the keys are established, the protocol can be initiated on request from one of the parties after they identify each other. For example, with a bi-direction symmetric-key challenge-response, assuming the prover initiated the protocol, the verifier responds with a nonce (challenge), and the prover generates a MAC of the nonce and combines this response with a nonce of their own. The verifier confirms the MAC matches the nonce with their own key. If confirmed they respond with an acknowledgement and a response MAC to their challenge. The prover verifies the response and can then also respond with an acknowledgement. A uni-direction authentication operates similarly, but the prover does not generate a challenge and does not expect a response with the acknowledgment from the verifier. This uni-directional symmetric-key challenge-response entity authentication protocol can be seen in Figure 2.1.

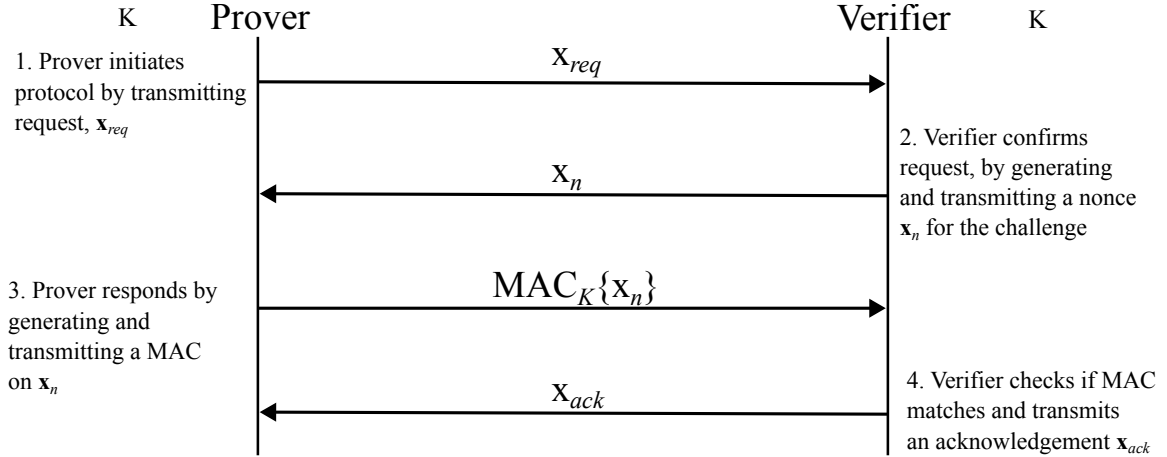


Figure 2.1: Uni-directional symmetric-key challenge-response entity authentication protocol

A public key challenge response operates similarly to the symmetric-key protocol, but the prover also shares a public-key that can be used to verify the digital signature. The public-key is typically shared through a certificate, which includes the identity of the prover and a signature of the public-key from a certificate authority. This method is typical for distributed challenge-response systems, where the prover and verifier have not previously met, and for which a symmetric-key system would be burdensome for distribution and storage reasons, or would require a connection to a verification server. An example of such a system is a single direction offline authentication in a credit-card payment system. Point-of-Sale terminals will have the certificate of the payment authorities (e.g. banks),

who have signed the public-key of the credit-cards. The credit-card signs the challenge of the terminal and includes a certificate in its response. The terminal verifies the certificate with the public key of the payment authority, before checking the response to the challenge. Later, the terminal can send bulk transfer of transactions to the payment authorities who can distribute the funds. This uni-directional symmetric-key challenge-response entity authentication protocol can be seen in Figure 2.2.

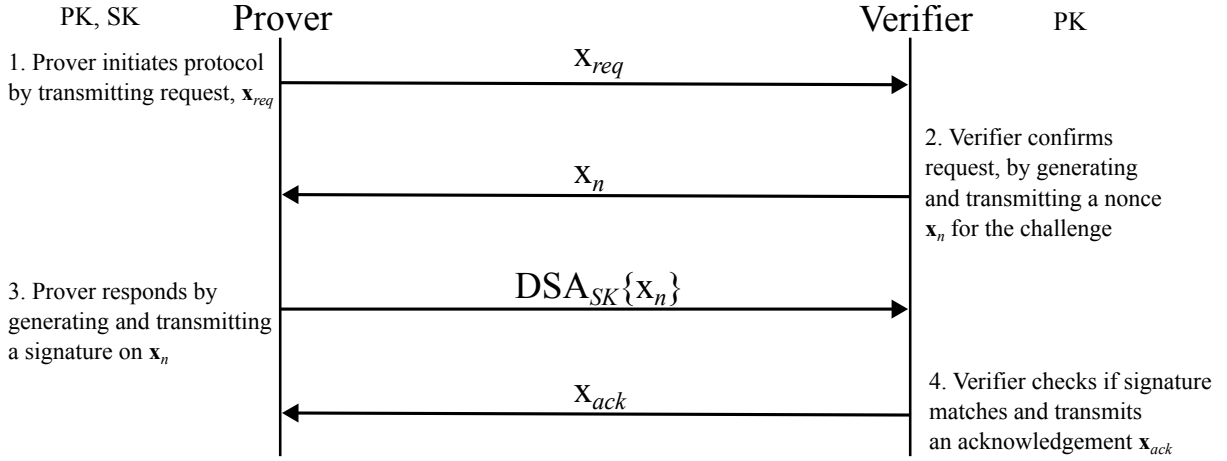


Figure 2.2: Uni-directional public-key challenge-response entity authentication protocol

Challenge-response protocols begin through an initiation process, which can begin manually or automatically. Automatic challenge-response protocols are typical in wireless systems. Either the prover or verifier broadcasts probes, which initiate the protocol once detected by the other party, who is also constantly analyzing the channel. For far field communication systems like Bluetooth and WiFi, these probing messages contain basic information, like the identity of the source. For near field systems, the probe can also contain information, but can also power the device, if it is a passive device found in RFID or NFC systems. When these protocols begin automatically as soon as a probe is received, some security also comes inherently from the distance of the communications. For example, an RFID or NFC payment systems only perform an automatic uni-directional challenge-response authentication within a distance less than 10 m. If the device comes within the distance, the authentication occurs to perform the payment. Similar processes occur with other uni-directional challenge-response protocols used for securing various systems. When the consequence of the challenge-response resolving positively is a real-world action, this system is referred to as a cyber-physical system, where a digital (security) process results

in something physical. With such a system an attacker could benefit from the protocol resolving, if they are nearby and can act prior to the legitimate parties.

2.2 The Relay Attack

The relay attack is a form of man-in-the-middle (MitM) attack, where the attacker sits between legitimate parties and forwards the messages those parties produce between them. They do not necessarily alter, filter, or inject any new, modified, or replayed messages into the communication link, though they may do these things to circumvent certain defenses. The attacker can coordinate multiple intermediaries that are between the communicating parties, and are complicit in the attack. The attack does not break semantic security for encryption and does not break unforgeability for authentication and integrity, if they exist in the messages. The motivation for the relay attack is to cause communications to occur, when they otherwise should not be occurring. For example, the attacker may want to extend the communication link beyond the intended distance. For most types of communications the relay attack would be inconsequential, as no information would be leaked or altered, if the communications generated by the legitimate parties was secure. The main consequences of the relay attack arises for communications that could have a direct affect on real world systems, which are colloquially known as cyber-physical systems.

The earliest reference to the relay attack was in 1976 from famed mathematician and computer scientist, John Conway [1]. In his work he describes a chess game between a novice child and two grandmasters. The child plays both grandmasters simultaneously and without their knowledge of the other game occurring. The child copies the moves of one of the grandmasters to the other, and responds likewise in the opposite direction, with no thought of their own. At most the child knows the basic rules of chess, particularly the ordering of moves (i.e., white moves first and players take turns making moves). At the end of the games (which end simultaneously), either there is a draw, or one of the grandmasters wins and one loses, meaning the child would have defeated a grandmasters, an otherwise unlikely outcome. In this example, the child is the relay and the grandmasters are the legitimate communicating parties. As obtuse as this example may sound, recent news in the chess world has had such accusations being made of this exact scenario occurring [94]. Two grandmasters were playing, where one of the grandmasters was accused of secretly acting as a relay for a computer, which allowed them to win against the other grandmaster (while losing to the computer).

In general, this form of attack is a method to defeat entity authentication protocols and is referred to as a “mafia-fraud,” which was first defined by Y. Desmedt *et al.* [2].

Specifically, a mafia-fraud is when two legitimate parties are being tricked into talking to each other via a relay, where neither party is aware they are talking via a relay. Colloquially, there is another form of relay attack called a “terrorist-fraud” [4]. In contrast with the terrorist-fraud, one of the legitimate parties is colluding with the relay attacker, where the legitimate colluding party wants to convince the other legitimate party that they are further away than they really are. This must happen without the legitimate party revealing any secret key that they own. One example of why one would want to collude with a relay is for plausible deniability. If one could prove they are physically somewhere else while colluding with an attacker to perform a relay attack, they may not be held accountable for the resulting resolution of communications. Finally, there is a third form of relay attack, referred to as a “distance-fraud”, where one of the legitimate parties is colluding with the attacker, but wants to convince the other legitimate party they are closer than they appear. Fundamentally, each type of fraud has the relay attacker operate the same, but they differ depending on the culpability of one of the communicating parties involved with which ever protocol is occurring. It is important to clarify that mafia-fraud attacks are an easier problem to solve, as any information received from the relay can be assumed to be confidential, intact, authentic, and not nefarious. This allows for feedback to be sent between the legitimate parties that may be used to detect mafia-frauds. Under terrorist-fraud and distant-fraud attacks, information can be nefarious due to the collusion of one of the legitimate parties, thus feedback cannot be used to reliably determine if a relay attack is occurring or not.

The relay attack functions with any form of communications, whether wired or wireless. An example of a wired relay attack can be found when somebody taps into a wired connection, stealing genuine authentication signals to provide services to a second device. Such attacks were prominent in previous decades, when cable television and landline telephones were widespread. In this thesis, only the wireless situation is considered as it is the most consequential and widespread, given the increasing use of wireless devices for authentications, and the ease of performing relay attacks with wireless communications. Furthermore, this thesis focuses on digital communications, in which binary symbols are being exchanged between the legitimate parties.

As stated in the introduction, this thesis will put exclusive focus on detecting the mafia-fraud relay attack occurring in wireless challenge-response entity authentication systems, as it is the most prominent in literature and yields the largest consequences to the real-world. Mafia-fraud relay attacks on entity authentication systems, where neither party is colluding with the attacker include the examples of tricking payment systems to authenticate transaction and accessing places and devices protected by passive access systems.

To describe the relay attack, consider two legitimate communicating parties, the prover

and the verifier. The prover and verifier communicate via a wireless link. The relay sits in between the prover and verifier and forwards messages from one to the other. From the perspective of the prover and verifier, the attacker is a proxy verifier and proxy prover, respectively. The ideal mafia-fraud relay attack can be seen in context to the uni-directional challenge-response authentication protocol in Figure 2.3.

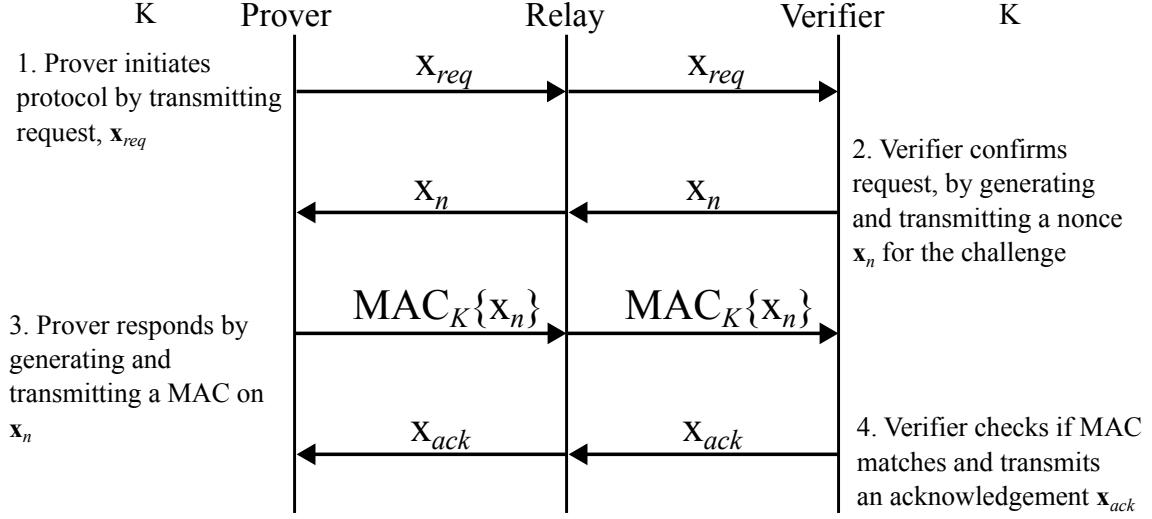


Figure 2.3: Mafia-fraud relay attack on a challenge-response authentication protocol [69]

In this ideal example, assume there are no errors between each transmission, whether from the prover, verifier, or relay. In reality, the relay attack in a wireless system is not ideal, and non-idealities appear between transmission and reception. The wireless relay attack in this scenario can be split into two main types of attacks depending on how the relay decides to operate. The first is known as a decode-and-forward relay attack and the second is an amplify-and-forward relay attack. In the decode-and-forward attack, the attacker processes the received signals by demodulating and decoding the symbols being transmitted from either the prover or verifier. After this, the symbols are processed again by coding and modulating before being forwarded to the opposite party. This means there is an intermediary step where the attacker has the information in digital form, though they may not have any knowledge of what the original message is due to encryption, and cannot edit or forge it due to integrity checks. An important distinction is that the attacker strips the effects of the wireless channel when performing the decoding and demodulation, through some estimations process. The information stripped includes the effects of noise and channel filtering of the transmit and receive hardware that exists between the transmitter and

relay. When the attacker forwards the message, these effects appear again, but they do not exist from the same sources as the original message. To extend the range even further, the attacker relay can be divided into two parts, an individual proxy prover and proxy verifier. In this case, the proxies are coordinating among each other and have some sort of link (wireless or wired) that can connect them over any relevant distance. The decoded information from one proxy is transmitted through the link to the other proxy, to be forwarded. The link for the decoded information can be done with any digital communication method, such the Internet Protocol (IP). An example of this would be having a prover and verifier being a credit-card and point-of-sale (PoS) system, respectively. The proxy prover and proxy verifier are two smartphones that relay information between them using WiFi and the internet, which establishes a significantly larger distance than the RFID communication system could ever allow. In this case the proxies essentially act as a single relay due to this connection. The layout of the decode-and-forward attack can be seen in Figure 2.4. with the details of the protocol steps detailed in Table 2.1

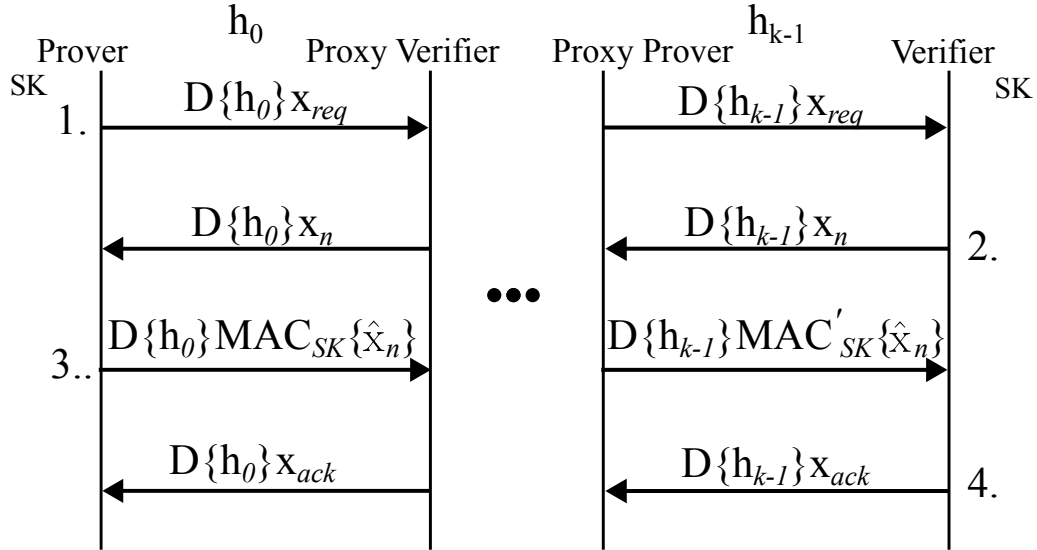


Figure 2.4: Decode-and-forward mafia-fraud relay attack

Table 2.1: Steps of the Decode-and-Forward Relay Attack on a Challenge-Response Authentication Protocol

step	Details
1	Prover initiates protocol by transmitting request, $\mathbf{x}_{req}$
2	Verifier confirms request, by generating a nonce $\mathbf{x}_n$. The nonce is transmitted to the prover
3	Prover uses the estimated received nonce $\hat{\mathbf{x}}_n$ to respond by transmitting a MAC on the estimate,
4	Verifier checks if received MAC matches the expected and transmits an $\mathbf{x}_{ack}$ with a pass or fail.

In this attack the transmission at the last hop is not necessarily the exact original transmission from the legitimate parties, but is an estimate of the original data, passed along each proxy. Each step is treated as a Hadamard multiplication of the channel filter $\mathbf{h}_i$ and the transmitted symbol (some vector of samples), but before relaying the information is estimated so the transform $\mathbf{h}_{l-1}\{\cdot\}$ is not accumulated. The relaying of step 2 can be seen in Figure 2.5

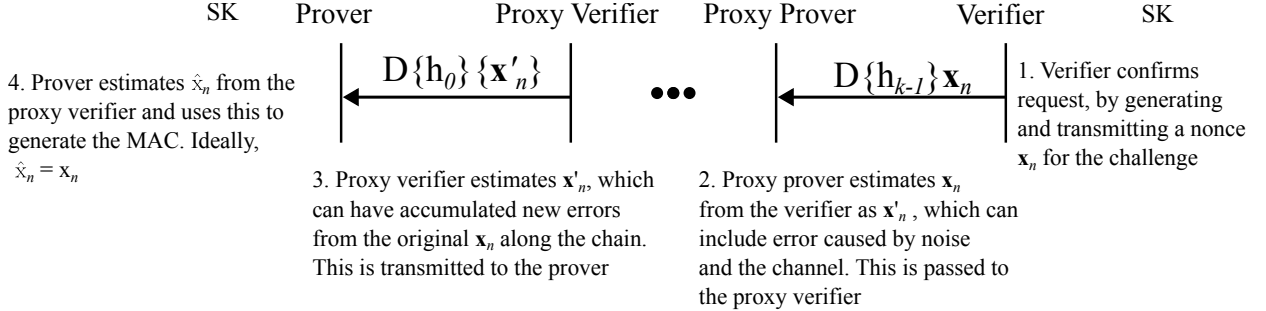


Figure 2.5: Decode-and-forward mafia-fraud step 2 in detail

The relay attack can also occur purely in the analog domain. The amplify-and-forward attack operates by forwarding the signal purely through radio-frequency (RF) hardware, and does not decode and does not demodulate (i.e., estimate) the original digital information from the received signals. Figure 2.6 shows an amplify-and-forward relay attack through a single relay acting as both the proxy prover and verifier. The signal is received

at one end, and is transmitted at the other end. Because there is no decoding and demodulation (and the reverse), all the effects of the communication (noise, fading, hardware non-idealities, etc.) are preserved between the parties. Each transmission along the analog relay path accumulates the path, represented by the non-linear transform $h_l\{\cdot\}$, opposed to the decode-and-forward relay, where only a single transform affects each transmission. The limitation is that the relay distance extension is limited to what can be amplified by a single device at the given frequency of communications. To get around this, an intermediary frequency can be used to increase the range of communication between several relays in series. If done wirelessly, there is a prower prover and proxy verifier, like in the decode-and-forward scenario, but also as many intermediary nodes as the attacker would like, operating at different frequencies. The number of relays in series is limited by the noise and other negative effects that are accumulated at each node, thus there is still a limit in distance, compared to decode-and-forward, which has no limit, as each step removes the channel effects. The extension can also be done via a wire, which limits the distance to the length of the wire (which can be very long).

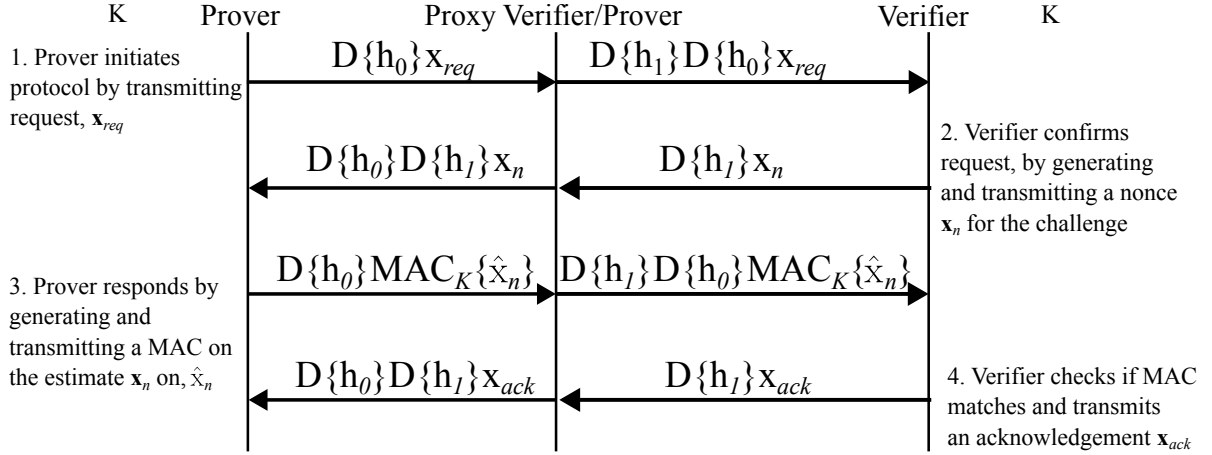


Figure 2.6: Amplify-and-forward mafia-fraud relay attack through a single relay

There is also another relaying method called “quantize-and-forward”. As far what could be found in literature, this has only briefly been mentioned in a single source by R. Abubaker [69]. Quantize-and-forward quantizes the received analog signal using an

analog-to-digital converter (ADC), then forwards it digitally, in the same manner as decode-and-forward to a proxy. The proxy utilizes a digital-to-analog converter (DAC), to then retransmit the signal to the legitimate party. Any link can be used to forward the quantized signal. This relay attack mixes the qualities of both decode-and-forward and amplify-and-forward relay attacks. The channel and hardware effects are preserved in the first and last steps, albeit with a new quantization error, but it is not limited in distance like the amplify-and-forward attack, as the quantized signal can be transmitted between proxies digitally. The issue with quantize-and-forward signals is the size of the relayed signal. In particular, each sample has to be transmitted, which can consist of thousands of samples, each being 16 to 32 bits (8 to 16 bits for each inphase and quadrature component). This can quickly become many megabits that need to be transmitted for each transmission, which might not be easily relayed, compared to amplify-and-forward which can be relayed instantly, and decode-and-forward, which only on a per symbol basis, not a per sample basis like quantize-and-forward.

In general, the choice between a decode-and-forward relay versus an amplify-and-forward relay is dictated by the capabilities of the attackers. When considering the proliferation of standardized platforms, like RFID, NFC, Bluetooth, and WiFi, cheap devices or even free software for readily available devices could easily allow anyone to perform a decode-and-forward relay attack. For example NFCProxy is an open source application for Android, which allows many consumer devices (e.g., smartphones and tablets) to easily capture and forge NFC packets, which could be used to perform decode-and-forward relay attacks [19]. Decode-and-forward attacks are also useful for performing long-distance attacks, as they inherently do not accumulate channel errors between transmissions, and the digital information can easily be transmitted between users in a few packets, regardless of the communication systems being used by any party. On the other hand, amplify-and-forward relay attacks requires dedicated hardware, which may not be readily available for all attackers. Since the hardware is not performing any demodulation, it may only include simple transceiver components (e.g., mixers, amplifiers, antennas, etc.) and does not need to be complex. The hardware may also be able to work on multiple devices at the same frequency, regardless of the protocols and communication systems being used. As mentioned, analog-and-forward relays need to be cascaded together to achieve longer distances, which can accumulate errors, with added noise being of particular concern. While quantize-and-forward systems might appear to be an ideal mix between the two systems, quantizing hundreds of samples into bits (typically 12-bit samples) can result in a significant amount of data that needs to be shared between the proxies. In practice, decode-and-forward relay systems are the most prominently used by attackers since relaying between proxies is generally easier, and demodulation is relatively easy for common devices, and only two

proxy relay devices are needed by the attacker. The quantize-and-forward relay attack is otherwise handled the same as an amplify-and-forward relay attack, so it will not be considered in the rest of the thesis.

When considering the actions of the attacker, what has been described in the decode-and-forward and amplify-and-forward attacks only considers relays that do not perform any modification on the information or signal, other than the reception and retransmission processes. Particularly for decode-and-forward relaying, it is assumed these processes are the same, regardless if a legitimate party or relay is transmitting. In this scenario, the attacker is referred to as a passive attacker, since they are not doing anything extra beyond retransmitting the digital information from the legitimate parties. In contrast, an attacker could interject and modify any part of the transmission process itself, including any preamble that exists. In this scenario the attacker is referred to as an active attacker. An example of an active attacker adapted in step 2 can be observed in Figure 2.7.

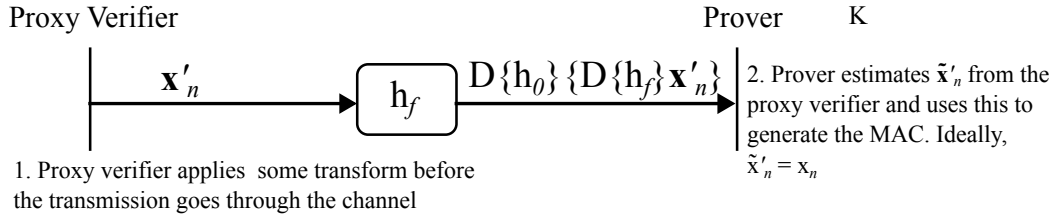


Figure 2.7: Active attacker in a decode-and-forward mafia-fraud relay attack through a single relay

The transform applied by the attacker can be anything, in theory. In this thesis, the assumption will be made that the attacker will apply an unspecified finite impulse response (FIR) filter, other than the constant (gain) filter, to perform an active attack. A gain can be applied by either the decode-and-forward or amplify-and-forward relays, and will be considered in subsequent chapters.

2.3 Wireless Communications

2.3.1 Basic Communications

At the basis of practical relay attacks is wireless communications. Expressing the mathematics in wireless communications is necessary to understand the relay attack and the

proposed solutions for it. When transmitting digital information, a general transmitted analog signal can be expressed in the form:

$$\begin{aligned} s(t) &= a(t) \cos(2\pi(f_c + f(t))t + \theta(t)) \\ &= a(t) \cos(2\pi f_c t + 2\pi f(t)t + \theta(t)), \quad 0 \leq t \leq T. \end{aligned} \quad (2.1)$$

The information to transmit can be placed in the amplitude $a(t)$, the phase $\theta(t)$, or the frequency $f(t)$. The duration of the transmission is T seconds, which also corresponds to the bandwidth of the signal $B_s = 1/T$, though it can practically be larger if $f(t)$ changes significantly within the period T . In the frequency domain, the signal is centred at the carrier frequency of f_c , due to the modulation of orthogonal signals $\cos(2\pi f_c t)$ and $\sin(2\pi f_c t)$. This signal is referred to as a bandpass signal because it is centred around f_c . The carrier frequency plays an important role for designing the overall system, but generally has no effect on the information being transmitted. Utilizing the trigonometric identity $\cos(u + v) = \cos(u)\cos(v) - \sin(u)\sin(v)$, the transmitted signal can be expressed by:

$$\begin{aligned} s(t) &= a(t) \cos(2\pi f_c t + 2\pi f(t)t + \theta(t)) \\ &= a(t) \cos(2\pi f(t)t + \theta(t)) \cos(2\pi f_c t) - a(t) \sin(2\pi f(t)t + \theta(t)) \sin(2\pi f_c t) \\ &= \Re\{a(t)e^{j2\pi f(t)t + \theta(t)}e^{j2\pi f_c t}\} \\ &= \Re\{u(t)e^{j2\pi f_c t}\} \quad 0 \leq t \leq T. \end{aligned} \quad (2.2)$$

In this form, it is clear how both $\cos(2\pi f_c t)$ and $\sin(2\pi f_c t)$ are being modulated by the information signals. Since the carrier frequency is agnostic to the information being transmitted, it can typically be ignored in analysis. This allows focus to be on the information contained in $u(t) = a(t)e^{j2\pi f(t)t + \theta(t)}$. The signal $u(t)$ is referred to as the complex baseband signal. If $u(t)$ is real, then in the frequency domain it can be considered conjugate symmetric, but if it is complex, then this does not hold. It is also useful to work with the complex baseband, because it allows simulations to be significantly easier to implement. generally, to re-create a signal from digital sampling, you must sample at the Nyquist rate or higher, which corresponds to 2 times the largest frequency (bandwidth). If a bandpass signal is used, this can be incredibly large. For example, in WiFi, a typical carrier frequency is 2.4 GHz with a signal bandwidth of 20 MHz. This would require a sampling rate of ≥ 5 GHz, which would be impossible for any computer to manage in terms of running time and memory. That is why focus is on the baseband, particularly for simulations, where a received and transmitted signal would only have to be sampled at ≥ 40 MHz, which is

still large, but practical with modern computers and converters used to convert $u(t)$ to and from a digital signal.

The real and imaginary parts of the complex baseband signal are referred to as the inphase component, $I(t)$, and the quadrature component, $Q(t)$, respectively

$$u(t) = I(t) + jQ(t).$$

To perform a transmission, a binary string is grouped into symbols that contain a certain amount of bits in them. For example, there can be 2-bit symbols, given by the set $\{00, 01, 10, 11\}$. The symbol-space has a cardinality of $M = 2^n$. The modulator maps these binary symbols into a complex baseband signal $u(t)$, and modulates the carrier signal to generate the output $s(t)$. Common modulation schemes include M-ary Amplitude Shift Keying (M-ASK), M-ary Phase Shift Keying (M-PSK), M-ary Frequency Shift Keying (M-FSK), and M-ary Quadrature Amplitude Modulation (M-QAM). These modulations map the binary-symbols to the amplitude, phase, or frequency. The symbol duration is limited in time to $0 \leq t \leq T$, which is a rectangular pulse show pulse. In the frequency domain this has a sinc function representation, which has significant frequency side-lobes. Furthermore, the instantaneous transition of the rectangular pulse is not realistic in the real world. Instead, some pulse $g(t)$ is used to ideally minimizes both the side-lobes in frequency and in time. To find an acceptable pulse, a choice must meet what is called the Nyquist criterion. This criterion requires that the pulse must be equal to 0 at all kT , where k is a non-zero integers. This is to prevent inter-symbol interference (ISI). Pulse shapes includes a sinc, raised cosine, or root-raised-cosine.

This thesis will only focus on the transmission of M-PSK symbols. For M-PSK, there is an index $0 \leq i \leq M - 1$, which is selected based on a mapping from a binary symbol to an index

$$I(t) = E_s \cos(\theta_i)g(t), \quad Q(t) = E_s \sin(\theta_i)g(t), \quad \theta_i = \cos(2\pi \frac{i}{M-1})$$

The amplitude (energy in the symbols) is constant, and the phase θ_i changes for depending on which symbol is being sent, where each symbol is seperated by $\geq 2\pi/M$ from the next closest symbol. For example, Binary-PSK (BPSK) consists of $\{\theta_1, \theta_2\} = \{0, \pi\}$.

2.3.2 Fading Channel Effects in far field Communications

In this section, details are given on the channel effects that occur in far field wireless communications. The far field indiciates any distance that is significantly larger than the

wavelength $\lambda = C/f_c$, where C is the speed of light (3×10^8 m/s). Typically, this distance would correspond to $\geq 10\lambda$. The theory and mathematics in this section comes from the textbook by A. Goldsmith [47, 48].

A baseband signal $u(t)$ that is transmitted becomes affected by the channel when it arrives at the receiver. These channel effects include additive white gaussian noise (AWGN), path-loss, shadowing, multipath fading, and Doppler's effect due to movement in the channel. AWGN is the thermal noise caused by the Brownian motion of electrons at the receiver side. It is represented by a random process $w(t)$, where each time instance is an independent Gaussian random variable with $\sim \mathcal{CN}(0, N_0/2)$, where N_0 is the noise power. Noise is not exclusive to far field transmissions. All measured analog signals are affected by noise. Path-loss represents the dissipation of power as signal radiates away from the source (transmitter). The dissipation is dependent on the environment, but in this thesis only Friis' free-space path-loss is considered, for simplicity. This is represented by the power attenuation

$$\rho = \left(\frac{\lambda}{2\pi d} \right)^2, \quad (2.3)$$

where d is the distance between the transmitter and receiver. Shadowing is another environmental effect caused by how the transmitted signal interacts with large objects, like building, trees, and hills. Shadowing is modelled as an additive log-normal effect on the received power. In this thesis shadowing is also ignore, for simplicity. Both path-loss and shadowing are affected by changes in the environment, particularly the distance between the transmitter and receiver. They are considered to be constant during the transmission period, as T is much smaller than the rate of the changes in the environment that would effect the large-scale fading.

Multipath effects are caused by the many different paths that a radiated signal can bounce off of before eventually arriving at the receiver. The multipaths arrive at different delays dictated by the total distance in the path and speed of light. The Doppler's effect is caused by the relative movement in the channel, which compresses or expands the wavelength of the arriving multipaths. Defining the received signal as $r(t)$, it has the expression

$$\begin{aligned} r(t) &= \sum_{n=1}^{N(t)} \sqrt{\rho_n} s(t - \tau_n(t)) + w(t) \\ &= \sum_{n=1}^{N(t)} \sqrt{\rho_n} \Re\{u(t - \tau_n(t)) e^{j2\pi(f_c + f_{d_n}(t))(t - \tau_n(t))}\} + w(t). \end{aligned} \quad (2.4)$$

Each of the paths is indicated by the indexed delay and phase-shift, which ranges between 0 to $N(t)$, and changes with time as objects in the channel move. The delay of the path is represented by $\tau_n(t)$ which also changes with time. Similarly, there is a frequency shift caused by the Doppler's effect along the path. The coefficient, ρ_n indicates the free-space path-loss in the path, caused by some path distance d_n . AWGN is always considered for a received signal.

In general there are infinite paths,

$$r(t) = \lim_{N(t) \rightarrow \infty} \sum_{n=1}^{N(t)} \sqrt{\rho_n} \Re\{u(t - \tau_n(t)) e^{j2\pi(f_c + f_{d_n}(t))(t - \tau_n(t))}\} + w(t). \quad (2.5)$$

Most of the paths will be extremely weak due to the effects of path loss. If the power in the path is less than the noise-power, then the signals on that path are irrelevant. These components are considered to be non-line-of-sight (NLOS), but there may be a single line-of-sight (LOS) path that arrives first, and has significantly more power, than the NLOS terms. This term is considered the reference for which all other delays are relative to.

$$r(t) = \sqrt{\rho_1} \Re\{u(t) e^{j2\pi(f_c + f_{d_1})t}\} + \lim_{N(t) \rightarrow \infty} \sum_{n=2}^{N(t)} \sqrt{\rho_n} \Re\{u(t - \tau_n(t)) e^{j2\pi(f_c + f_{d_n}(t))(t - \tau_n(t))}\} + w(t). \quad (2.6)$$

The factor Rician Factor K indicates how much more power (multiplicative) that the LOS term has compared to the NLOS terms. This is expressed as

$$K = \frac{\int_{t=0}^{+\infty} (\sqrt{\rho_1} \Re\{u(t) e^{j2\pi(f_c + f_{d_1})t}\})^2 dt}{(\int_{t=0}^{+\infty} \sum_{n=2}^{N(t)} \sqrt{\rho_n} \Re\{u(t - \tau_n(t)) e^{j2\pi(f_c + f_{d_n}(t))(t - \tau_n(t))}\})^2 dt}. \quad (2.7)$$

The received signal can also be expressed as being filtered through a time-varying channel, with an impulse response $h(t, \tau)$

$$r(t) = h(t, \tau) * s(t) + w(t), \quad (2.8)$$

thus $h(t, \tau)$ is equal to an impulse train,

$$h(t, \tau) = \sqrt{\rho_1} e^{j2\pi f_{d_1} t} \delta(t) + \lim_{N(t) \rightarrow \infty} \sum_{n=0}^{N(t)} \sqrt{\rho_n} \delta(t - \tau_n(t)) e^{j2\pi f_{d_n}(t)(t - \tau_n(t))}. \quad (2.9)$$

The channel is said to be narrowband at a time instance t , if $\max_n \tau_n(t) \ll T$, when only consider the paths $\tau_n(t)$ that are larger than the noise floor,

$$E_s \times \left(\frac{\lambda}{2\pi\tau_n(t) \times C} \right) < \sqrt{N_0}. \quad (2.10)$$

A narrowband channel implies all the delays are considered unresolvable. All the multipaths are considered to arrive at essentially the same time, so can be factored out as it is not dependent on the delay. The same cannot be said about the phases, as even small changes can have a significant effect. Note that the LOS and NLOS terms arrive at relatively the same moment, so the path-loss effects are equal

$$r(t) = \Re\{u(t)e^{j2\pi f_c t}(\sqrt{\rho}e^{j2\pi f_{d_1} t} + \sqrt{\rho} \lim_{N(t) \rightarrow \infty} \sum_{n=0}^{N(t)} e^{-j2\pi f_{d_n}(t)(t-\tau_n(t))})\} + w(t). \quad (2.11)$$

The phase changes caused by the path-loss and Doppler's effect in each unresolvable multipath are random, so the central limit theorem (CLT) can be invoked. The sum is then a complex gaussian with distribution $\sim \mathcal{CN}(0, \rho)$. The resulting distribution is known as Rayleigh fading, given by the Rayleigh distribution of the amplitude. The phase has a uniform distribution $[0, 2\pi]$. The variance is the path-loss in the LOS term $\rho = (\lambda/2\pi d)^2$, as all the paths can be considered to have arrived at the same time. The received signal then has a LOS term that is deterministic and with a random NLOS term. In a narrowband transmission, the received complex baseband signal is defined as $y(t)$, such that $r(t) = \Re\{y(t)e^{j2\pi f_c t}\} + n(t)$,

$$\begin{aligned} y(t) &= \left(\sqrt{\rho}e^{-j2\pi f_{d_1} t} + \sqrt{\rho} \lim_{N(t) \rightarrow \infty} \sum_{n=0}^{N(t)} e^{-j2\pi f_{d_n}(t)(t-\tau_n(t))} \right) u(t) \\ &= \sqrt{\rho} (e^{-j2\pi f_{d_1} t} + \beta(t)) u(t). \end{aligned} \quad (2.12)$$

The NLOS term $\beta(t)$ is a complex gaussian random process. In general, the NLOS term can be considered to be wide-sense stationary (WSS), with an autocorrelation that is dependent on the spread of the arrivals of the multipaths. The whole coefficient of $u(t)$, $(e^{-j2\pi f_{d_1} t} + \beta(t))$ is not WSS, due to the LOS term, which introduces a time-varying mean. The noise can also be incorporated into the received complex baseband signal as using it to modulate the carrier has no affect on the distribution and the total power is conserved,

$$y(t) = \sqrt{\rho} (e^{-j2\pi f_{d_1} t} + \beta(t)) u(t) + w(t), \quad (2.13)$$

where $w(t)$ is now a complex gaussian random process with the distribution at each time instance $\sim \mathcal{CN}(0, N_0/2)$.

In the thesis there is no assumption that the communications are narrowband. The multipaths are generalized such that each resolvable path consists of an infinite sum of unresolvable path that are similar to $\beta(t)$. Considering the time-varying impulse response seen in (2.8), each NLOS term can then be modelled as a single unresolvable path $\beta_n(t)$ that is a complex gaussian random process in time,

$$h(t, \tau) = \sqrt{\rho_1} e^{-j2\pi f_{d1}t} \delta(t) + \sum_{n=0}^N \sqrt{\rho_n} \beta_n(t) \delta(t - \tau_n). \quad (2.14)$$

The uncorrelated scattering modeled is assumed, where at each time instance the time-varying NLOS coefficients at each time delay are uncorrelated with each other. This can be considered to be a filter, where each tap is modeled as a narrowband channel. The autocorrelation of the impulse response can be expressed by

$$A_h(t_1, t_2; \tau_1, \tau_2) = E\{h(t_1, \tau_1)h^*(t_2, \tau_2)\}. \quad (2.15)$$

As mentioned, the delays are uncorrelated, such that $A_h(t_1, t_2; \tau_1, \tau_2) = 0$ when $\tau_1 \neq \tau_2$. The time domain is not WSS due to the LOS term affecting the mean, but the distribution is still only dependent on the time difference $\Delta t = t_2 - t_1$. The autocorrelation can be simplified to,

$$\begin{aligned} A_h(t_1, t_2; \tau_1, \tau_2) &= A_h(t, \Delta t; \tau) \\ &= E\{h(t, \tau)h^*(t + \Delta t, \tau)\}. \end{aligned} \quad (2.16)$$

Considering a time instant, $\Delta t = 0$, the autocorrelation is $A_h(t, \tau)$,

$$A_h(t, \tau) = \begin{cases} \sqrt{\rho} (e^{-j2\pi f_{d1}t} + A_{pdp}(0)), & \text{if } \tau = 0 \\ \sqrt{\rho} A_{pdp}(\tau), & \text{if } \tau > 0 \end{cases} \quad (2.17)$$

$A_{pdp}(\tau)$ is known as the power-delay profile (PDP), and it indicates the average power of each delay term τ . The first delay term $\tau = 0$ contains the LOS component, in a similar manner as the narrowband $\beta(t)$, but at just a fixed time instance ($\Delta t = 0$). In this thesis, including the simulations, the PDP is assumed to be a negative decaying exponential,

$$A_{pdp}(\tau) = \frac{1}{\frac{1}{\tau_{rms}} \int_0^\infty e^{-\tau_{rms}\tau} d\tau} e^{-\tau_{rms}\tau}. \quad (2.18)$$

This integral in the denominator is to normalize the power in the PDP. τ_{rms} is the root-mean square (rms) delay spread, which statistically corresponds to the rms spread between the LOS arrival and the last delay with significant power (higher than noise floor).

Considering the autocorrelation for a fixed τ in terms of time Δt ,

$$A_h(\Delta t, \tau) = \begin{cases} \frac{1}{f_{d1}} e^{-j2\pi f_{d1} \Delta t} + A_h(\Delta t), & \text{if } \tau = 0 \\ A_h(\Delta t) & \text{if } \tau > 0. \end{cases} \quad (2.19)$$

The autocorrelation in time, $A_h(\Delta t)$ is assumed to follow Jakes' Uniform Scattering model,

$$A_h(\Delta t) = J_0(2\pi f_{d_{max}} \Delta t), \quad (2.20)$$

where J_0 is the zeroth order Bessel function of the first kind.

For the PDP, the RMS delay value can be calculated by,

$$\sigma_{rms} = \sqrt{\frac{\int_0^\infty (\tau - \bar{\tau})^2 A_{pdp}(\tau) d\tau}{\int_0^\infty A_{pdp}(\tau) d\tau}},$$

which evaluates to $\sigma_{rms} = \tau_{rms}$ for the exponentially decaying PDP in equation (2.17). The delay spread is an important channel parameter because it dictates if the channel is narrowband and can be approximated by $\approx \sigma_{rms}$. As mentioned, if $T \ll \sigma_{rms}$, the transmission is narrowband. If it is not narrowband, it is wideband and suffers from intersymbol-interference (ISI). The Fourier transform of the PDP

$$A(\Delta f) = \mathcal{F}\{A_{pdp}(\tau)\}.$$

The period where the channel is no longer correlated is equivalent to the bandwidth B_c , which is approximately $B_c = \frac{1}{\sigma_{rms}}$. This is referred to as the coherence bandwidth, and is an important quality because it correlates with the frequency spacing required for measurements of $\mathcal{F}_h(t, \tau)$ to be uncorrelated, which implies independence when considering the CLT and the gaussian distributions.

Similarly, the autocorrelation in time for a fixed delay τ , $A_h(\Delta t)$ of the impulse response $h(t, \tau)$ correlates over time. Particularly of importance is how much time passes before the channel for a given delay decorrelates. This is known as the coherence time t_c . For the uniform scattering model this turns out to $t_c \approx \frac{1}{2f_{d_{max}}}$, where $f_{d_{max}}$ is the maximum Doppler spread, given by

$$f_{d_{max}} = \frac{v}{\lambda}.$$

The Fourier transform of the time autocorrelation tells shows how the power is spread amongst the Doppler frequencies (spread), but it not as important for the speeds focus in this thesis (terrestrial/ground speeds).

2.3.3 Orthogonal Frequency Division Multiplexing

From Section 2.3.2, the effects of ISI and frequency-selective fading were detailed, which was dependent on the delay spread of the channel τ_{rms} , relative to the symbol duration T . If $\tau_{rms} > T$, then the communications were wideband, and could suffer from ISI and frequency-selective fading, due to the channel decorrelating (i.e., change) in frequency over the bandwidth of the transmitted signal. Increasing the throughput means reducing T , which can eventually lead to the ISI situation. Similarly, the large bandwidth (from the small symbol duration) makes it more likely that some sub-bandwidth of the symbol is fading, which can destroy the whole received transmission. One method to compensate for this is to multiplex a fast single data stream in to multiple independent slower streams, each on different subcarrier frequencies. Furthermore, these carriers can be chosen to be orthogonal so that they can overlap in the frequency domain, resulting in high spectral efficiency. Utilizing multiple streams to transmit is known as frequency division multiplexing (FDM) and when the carriers are orthogonal it is orthogonal frequency division multiplexing (OFDM). By parallelizing a single fast data stream into multiple slower parallel streams, OFDM overcomes the effects of ISI and reduces the negative impact of frequency-selective fading. If a single data stream has M bits transmitted on a single symbol with a time duration T , OFDM with N data streams would transmit M/N bits over N symbols with duration NT . A visualization of this can be seen in Figure 2.8. OFDM has become the standard modulation scheme for all communication systems that want to achieve high-throughput communications due to its ability to combat ISI, frequency-selective fading, and achieve high spectral efficiency, which is important when bandwidth is such a scarce resource. This includes systems such as WiFi, 4G, 5G, and many others.

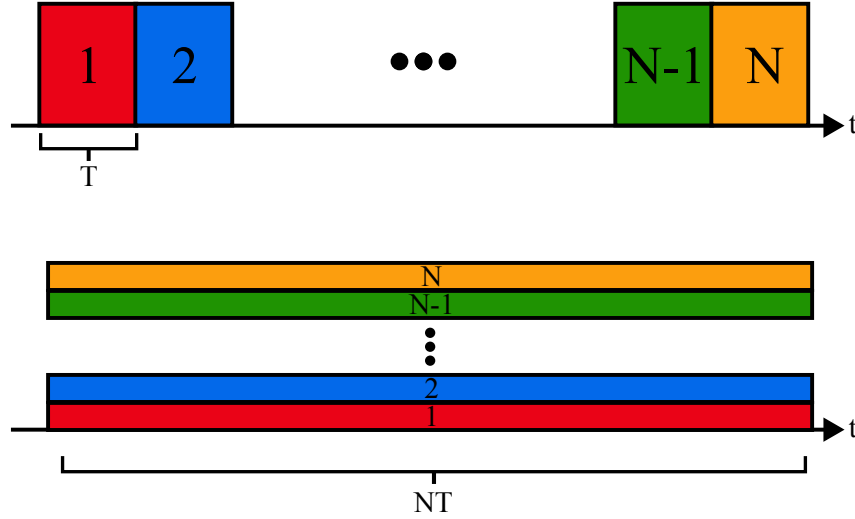


Figure 2.8: Symbols before and after multiplexing

OFDM is easy to implement when utilizing the DFT and an digital-to-analog and analog-to-digital converter (DAC/ADC), as transmissions can be easily processed digitally, before mixing and transmitting. To implement OFDM, the number of orthogonal paths is selected as N . For each OFDM symbol transmission (T), each path is given a complex symbol corresponding to a binary mapping to either M-ASK, M-PSK, M-QAM, or any other mapping to fixed symbols with constant amplitudes and phases (e.g., not M-FSK). The IDFT of these symbols results in the sampled I and Q at each frequency of $f_n = nf_s/N$, where $\{n \in \mathbb{Z} | 1 \leq n \leq N\}$ and $f_s = 1/T_s$ is the sampling frequency and period. This can then be converted through an ADC and mixed to the desired carrier frequency. Typically a cyclic-prefix is added after the IDFT, which acts as both a guardband to additional ISI and to ensure the convolution through the channel corresponds to cyclic convolution. The receiver structure is the opposite steps, but low-pass filters (LPF) are used to extract the I and Q portions. The structure of a typical OFDM transmitter and receiver are seen in Figure 2.9.

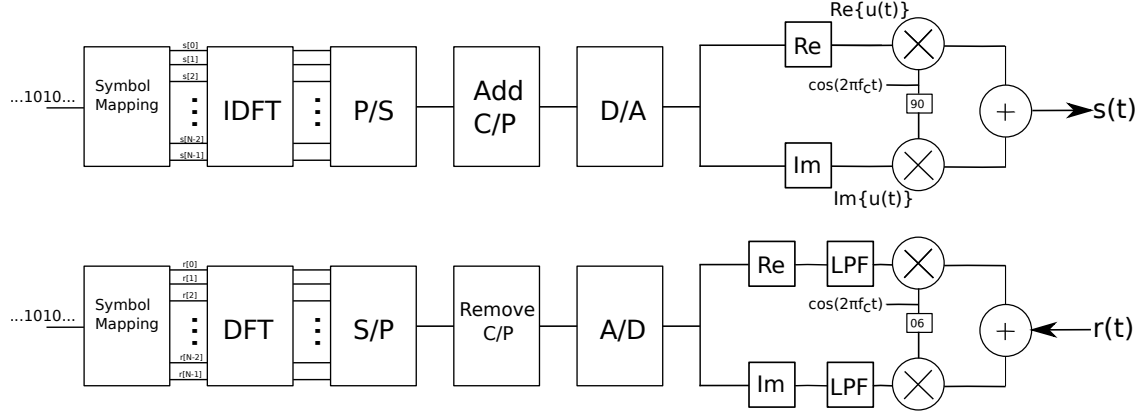


Figure 2.9: OFDM Transmitter and Receiver

In this structure, the signal analysis can be simplified into two different portions. The transmitted signal can be represented as $\mathbf{s} = (s[0], s[1], \dots, s[N-1])$ and the received signal is $\mathbf{r} = (r[0], r[1], \dots, r[N-1])$, where signals are the sampled basedband signals in the frequency domain. Given the OFDM system, the analysis in this thesis focuses on the bandband signals. The intermediary still follows the channel model represented in Section 2.3.2. Given this representation and considering (2.8), the received signal considered as

$$\mathbf{r} = \mathbf{D}\{\mathbf{h}_m\}\mathbf{s},$$

where the time symbol $0 \leq m \leq M-1$ corresponds to $h(t = mT_s, \tau)$. The frequency response $\mathbf{h}_m$ is the sampled Fourier transform of $h(t = mT_s, \tau = nTs)$ in τ and is the channel frequency response (CFR). Given the NLOS component of the channel in (2.8) having a covariance matrix given by

$$\Sigma_{\Delta t} = \begin{bmatrix} A_h(0) & A_h(T) & \cdots & A_h((M-1)T) \\ A_h(T) & A_h(0) & \cdots & A_h((M-2)T) \\ \vdots & \vdots & \ddots & \vdots \\ A_h((M-1)T) & A_h((M-2)T) & \cdots & A_h(0) \end{bmatrix}$$

for each tap, which is a circulant matrix of the sampled (2.20). Each tap in time m for $h(t = mT_s, \tau = nTs)$ is generated randomly from this distribution in time, while the power in the tap n comes from the power in the (2.18) for that tap.

2.4 Channel Reciprocity

From Maxwell's equation, a theorem known as Lorentz reciprocity details that two antennas in the far field operating in either transmit or receive will have the same antenna pattern regardless of who is transmitting or receiving, if both devices stay in the same position. This results in the voltage and current being the same at the antenna terminals, as the transmission travels through the same wireless environment. This can be manifested by representing the transfer impedance of the channel being the same in both directions, regardless of what is being transmitted. Either devices could theoretically find the channel impedance if they both probed the channel via a pilot exchange. Lorentz reciprocity can be extended to communications, where the channel impedance is represented by the channel impulse response (CIR) or the CFR, as seen in equation (2.8). Regardless if two devices are transmitting or receiving, the CIR or CFR is the same in both directions, if there has not been any movement between the devices or any other object in the channel. Considering a realistic channel, an estimate of the CFR can be extracted by two communicating parties if they communicate within the following conditions,

1. The same carrier frequencies and bandwidth around it are used for the corresponding transmissions. Particularly, the communications need to happen within the same coherence bandwidth,
2. The communications occur fast enough, such that the channel hasn't changed much. Particularly, the round-trip-time should happen within the coherence time of the channel.

Two legitimate devices exchanging pilots within the coherence time and at the same frequencies can extract a highly correlated estimate of the channel, given by the covariance matrices. The major cause of the difference is the AWGN being independent between the sets of measurements. Any third party that is listening in on the broadcasted communications could also estimate the channel, but if they are sufficiently far away, they will estimate a different measurement to the legitimate devices. In fact, if the communications occur at a distance larger than the movement given by the coherence time, an uncorrelated (independent) channel measurement will be received. For example, in the uniform scattering model from Section 2.3.2, from the temporal correlation given by (2.20), distances larger than 0.5λ will cause a third party to experience different channels than the ones seen by the legitimate communicating parties. If the signal-to-noise ratio (SNR) between the legitimate parties was significantly large, then their respective channels $\hat{\mathbf{H}}_{12}$ and $\hat{\mathbf{H}}_{21}$ in each direction, would be equivalent. A third party would have channels $\hat{\mathbf{H}}_{13}$ and $\hat{\mathbf{H}}_{31}$

with the first legitimate party and channels $\hat{\mathbf{H}}_{23}$ and $\hat{\mathbf{H}}_{32}$ with the second legitimate party. While $\hat{\mathbf{H}}_{13} \approx \hat{\mathbf{H}}_{31}$ and $\hat{\mathbf{H}}_{23} \approx \hat{\mathbf{H}}_{32}$ if the SNR is high, $\hat{\mathbf{H}}_{13} \neq \hat{\mathbf{H}}_{12}$ if they are far enough away. This can be visually seen in Figure 2.10 [69].

Figure 2.10:

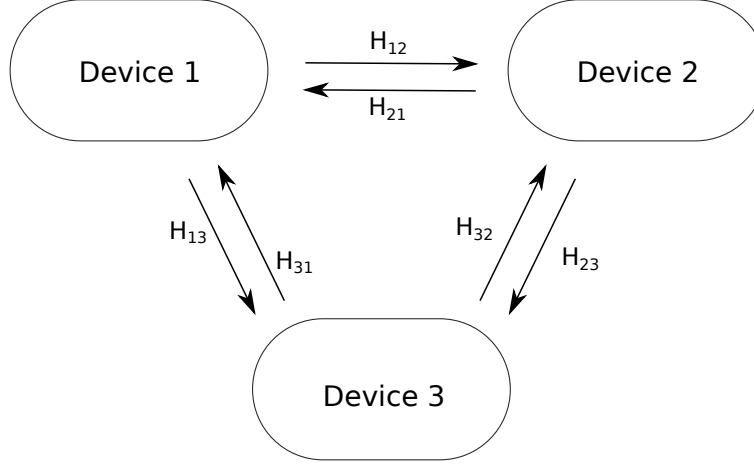


Figure 2.10: A 3-Device Half-Duplex System with a Reciprocal Exchange [69]

The randomness combined with reciprocity and channel decorrelation over a distance can be leveraged for security purposes, by having unestablished secret information (the channel) be accessible for two parties communication, but not for a third party eavesdropper.

2.4.1 Channel Non-Reciprocity

While the the ideal scenario from theory assumes this channel reciprocity, in reality a radio has many more components than just an antenna. For example, the receive and transmit path is seperated for many modern wireless communication devices so full-duplex communications can occur (albeit on different frequencies). Multiple transmit and receives paths also exist for multipl-input multiple-output (MIMO) communications, making it likely that a wireless devices selects a different antenna for different communications. Furthermore, devices add non-idealities to received signals, include synchronization offsets, which has a signifcant impact on the phase. These factors cause reciprocity to either not hold in the entirety, or to have a low cross-correlation.

2.5 Software Defined Radios

With hardware advancements, including field-programmable grid arrays (FPGAs), tuneable transceivers, DACs, ADCs, and the accompanying software interfaces, it has become easier than ever to create complex radio systems in a fast and iterative manner. Instead of designing and fabricating a device that can only communicate at a fixed frequency and for a specific modulation scheme, a customizable radio frontend that can be controlled through software can generate signals that are fed or extracted from a digital processor. The digital processor can be an FPGA, microcontroller, or a higher-level programming language that runs on a computer. The interface between the digital and analog side are the fast ADCs and DACs respectively. Such devices are known as Software Defined Radios (SDRs) and are prominent in many devices that could require configurable use cases, like meeting different regulatory and market requirements. They also exist as standalone devices, and are used primarily by industry and research for fast prototyping of new wireless devices. SDRs are great for exploring designs, but are not the most cost-effective method for generating profits compared to application specific radios, that can benefit from only including the necessary hardware. Typically, the controllable and tuneable frontend is what distinguishes an SDR from a homodyne or heterodyne transceiver that processes signals digitally. The general model of an SDR can be seen in Figure 2.11.

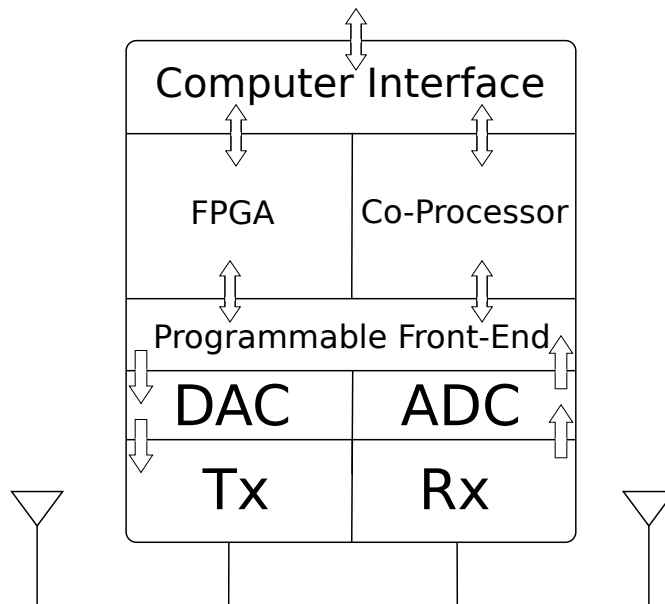


Figure 2.11: Typical layout of a Software Defined Radio

As mentioned, the digital signal processing can be done entirely on an FPGA, micro-processor, computer, or through a combination of them handling different parts of the signal. FPGAs are typically much faster and have less latency that is measurable, as they do not need to interact with an operating system (OS) or other unnecessary software. The trade-off is that higher-level software is typically easier to work with, and can leverage software libraries and code that are not easily implementable on an FPGA. For complicated protocols that have strict timing-requirements, it may be necessary to work with the FPGA, as the delay added from extra software and associated interfaces is too significant. Since the whole signal is processed digitally, the bandwidth can be quite large for high-throughput communications. This requires ADCs that have a very high sampling rate. Complex communication systems that include various synchronization processes, channel estimation, packet determination, and protocol state machines, can cause a significant processing requirement for high-throughput communications as well.

Chapter 3

Literature Review

This chapter contains a literature survey that focuses on three main areas of research. The first area of focus is on relaying, including the original research, implementations, and history regarding the relay attack, the vulnerabilities researchers have identified in current systems, and the design of different relay attacks. The second area of focus looks at the solutions that are proposed in the literature and industry for handling the relay attack. This includes theoretical and practical solutions that are starting to enter the market today. Focus is given on distance-bounding, contextualization, and neural network identification, which are the most common proposed solutions. Analysis is also given of their practicality. The third area of focus is on the literature that motivates the solution for the relay attack. This is primarily focused on physical-layer security and the few items of literature that are related to the ideas proposed in this thesis.

3.1 Relay Attack

The first proposition in the literature of something like a relay attack was by John Conway in the year 1976 [1]. As mentioned in Section 2.2, Conway conceptually detailed the attack in reference to a game of chess between an amateur and grandmasters, where the amateur was able to relay moves between them, ultimately winning one match at the expense of the other. While this is the first mention of the attack (though not in name), the relay attack was likely a known form of fraud for a very long time. Furthermore, other forms of “man-in-the-middle attacks” were likely not novel prior to 1976, even if they were not known by a specific name. The first titling of the relay attack as “mafia-fraud” was by Y. Desmedt *et al.*, where they detailed the attacks use for breaking the Feige-Fiat-Shamir

protocol, which is a zero-knowledge proof used for authentication (i.e., public-key) [2, 3]. Some of the use cases for the protocol were for authentication of passports or credit-cards. Y. Desmedt *et al.* also detailed the terrorist-fraud for the first time in the same paper, but did not refer to it by name. In the works by Y. Desmedt *et al.*, they did not directly propose a solution to the mafia-fraud, but acknowledged that if the physical description of the person was unique and could be adequately tested, the fraud could not occur (for passport verification). This notion of utilizing something unique (uncloneable) and testable (via physical observation), is a base-line for all solutions of the relay attack that exist. The first titling of the terrorist-fraud attack was by Bengio *et al.*, which they adapted from the mafia-fraud (which they refer to as the “middle person attack”) [4]. To thwart the relay attacks, they proposed the use of a Faraday cage (a metal container) when two devices are not performing authentication, which prevents wireless communications to any proxies. If the proxies cannot communicate, then there is no way for the relaying to occur. While this is a potential solution, it may not be feasible in many scenarios. For example, wireless devices that are tightly integrated into a larger systems are unlikely to be able to also integrate a Faraday cage.

In the literature there are many examples of the relay attack being performed on devices that utilize challenge-response entity authentication, almost entirely through the usage of a mafia-fraud. The first case of a mafia-fraud relay attack being performed on a consumer level device in literature was in work by G.P. Hancke on RFID tokens in the year 2006 [12]. The paper looked at performing the relay attack on a passive HF RFID token by using a decode-and-forward mafia-fraud relay attack. The results showed that security dependent on short communication distances was not sufficient against a capable attacker. The RFID tokens would typically only communicate a few centimetres, but the paper showed communications up to 30 cm away. Work by L. Francis *et al.* showed an NFC decode-and-forward relay attack, where the proxy devices communicated via Bluetooth, though they proposed other methods (HTTPS, SMS, etc.) could be utilized [11]. Modern attempts of the decode-and-forward relay attack have been shown to be easily implementable when utilizing smartphones and the built in NFC and Bluetooth capabilities that they have [16, 17, 18]. An open source Android application by E. Lee called *NFCProxy* was presented at DEFCON 2012 [19]. This application can be used on select android devices to snoop and forge NFC packets, allowing for potential relaying. With this application anyone can easily create a decode-and-forward relay attack with consumer Android devices, emphasizing the point that malicious actors can easily perform a relay attack, without having deep knowledge on the topic. For example, work by T. Bocek *et al.* showed this tool being used to relay contactless RFID payments from a PoS terminal to a VISA credit-card through android tablets running *NFCProxy* and communicating via

WiFi [96]. Though their implementation added significant delay of 670 - 2050 ms, they noted that the PoS terminal had no issues accepting the payment. This attack was repeated again by C. Thorpe *et al.*, where they also used the NFCProxy platform to perform a faster decode-and-forward relay attack within 14 - 68 ms on European debit and credit cards [18]. On American MasterCards, J. Vila *et al.* showed that a decode-and-forward relay attack could also be performed through an Android NFC relaying platform [40]. These cases were examples of decode-and-forward attacks using consumer-off-the-shelf (COTS) devices, but amplify-and-forward attacks are also prominent in literature. In work by A. Francillon *et al.*, they perform an amplify-and-forward mafia-fraud on car entry systems, where they performed mafia-fraud to unlock ten low-end and high-end vehicles [10]. In their experimentation, both wired and wireless amplify-and-forward methods were used to relay between the proxies. Relay attacks on vehicles have become prolific [13, 14, 15, 92, 93]. Relay attacks have been shown to be implementable on proxy devices against other NFC capable devices. A recent paper by P. Staat *et al.* performed an amplify-and-forward relay attack on a Bluetooth Low-Energy (BLE) used for PKES for a vehicle and on a smart lock. They were able to extend the unlocking to 65 m, which is much further than the single-digit metre distance beforehand [74]. To extend their communications they utilized a wired connection between the proxies, but the last-hop connections utilized BLE.

One extreme case of the relay attack being applied is in warfare. A story first identified by R. Anderson was during the Angolan Civil War [97]. Angolan rebels supposedly utilized a relay attack to extend identification, friend, foe (IFF) messages from South African bomber planes, tricking the South African air defense into thinking the Angolan rebel warplanes was the bomber. This resulted in the rebels bypassing air defenses and attacking a South African airbase, causing South African military casualties, motivating their eventual downgrading of involvement in the war. While the specific cause of the attack is disputed, relaying of IFF communications is a significant problem for governments and militaries, especially when considering that legitimate devices may be hundreds of kilometres away and travelling at high-speeds for a plane, or underwater for a submarine. In the literature surrounding IFF systems, the challenge-response authentication protocol is sometimes referred to as the “interrogation” process, where the verifier is referred to as the interrogator, which is trying to authenticate a prover referred to as the transponder [135].

3.2 Distance-Bounding

The first proposed verifiable solutions for the relay attack was based around looking at the distances between the legitimate communicating parties. If the distance is beyond an

expected threshold, then it is presumed that the communications must be occurring via a relay. This is colloquially known as “distance-bounding”, and was given this title by S. Brands *et al.* [5]. In their paper they proposed using distance-bounding as a method to deal with mafia-frauds and challenge-response entity authentication. They also define distance-fraud for the first time. This version of distance-bounding works by measuring the time-of-flight (ToF) or round-trip-time (RTT) between a random challenge and response bit exchange to estimate the distance between the communicating parties. Their result showed that an attacker has $1/2^n$ probability of impersonation, where n is the number of bits exchanged. It is important to distinguish that despite the verifier initiating the distance measurement through its own transmission, distance-bounding is not an active distance estimation, like an active radar system. Distance-bounding only measures the time difference from the returning transmission from the prover, relative to the original transmission from the verifier. The estimated distance for each bit exchange from the distance-bounding protocol is given by

$$\hat{d}_n = \frac{c(\Delta t_n - \hat{t}_p)}{2}, \quad (3.1)$$

where $\hat{d}_n$ is the estimated distance for the n^{th} bit exchange, c is the speed of light, Δt_n is the time difference measured by the verifier for the n^{th} bit exchange, and $\hat{t}_p$ is the processing time of the prover. The processing time t_p represents any additional time needed by the parties, between the legitimate parties, and must be estimated and securely exchanged between them. Under a relay attack, the real processing time increases as the relay may need to do processing (for decode-and-forward and quantize-and-forward). It is common for distance-bounding protocols to propose a fixed t_p , which is used to accommodate the randomness of processing on the devices. As a trade-off, a large t_p can be utilized by a relay to increase the communication distance, which makes the distance-bounding security weaker.

Distance-bounding is a popular method for detecting relay attacks, in the literature and in practice. Early works have looked at implementing distance-bounding for different authentication protocols. A. Levi *et al.* [6] proposed that distance-bounding could be used for preventing a relay attack in Bluetooth piconets. G.P. Hancke *et al.* proposed using distance bounding protocol for RFID systems, where they adapted the original distance-bounding protocol to function on a low-powered device [7]. This protocol is known as the Hancke and Kuhn Protocol (HKP) in literature. Other work by G.P. Hancke *et al.* attempts to improve their previous protocol, by dealing with errors during the round-robin exchanges with error-correcting codes and looks at the applicability to terrorist-frauds [8]. Other distance-bounding protocols have been released over the years that are updates on

the work of S. Brands *et al.* and G.P. Hancke *et al.*. J. Munilla *et al.* modify HKP by introducing nulls into the bit exchange, to preempt the attacker injecting random guess to make themselves look closer [20].

In general, these ToF/RTT distance-bounding protocols suffer from a general issue: in practice, it is difficult to get a precise and accurate measurement of time differences, especially within nanosecond magnitude error. Since the distance is calculated using the RTT and the speed of light, even error within dozens of nanoseconds can result in the distance estimate being off by several metres, which could cause the verifier to have a high false positive and false negative rate. The ability to precisely measure the ToF for short range communications is dependent on the time synchronization and processing delay of the devices and communications. For example, a low-powered RFID device may add significant delay due to its slow processors, making it difficult for the verifier to select an estimate $\hat{t}_p$. Symbol timing synchronization can easily add dozens of nanoseconds when identifying the start of a symbol, especially when the symbol duration is long, the sampling rate is low, and there is frequency-selective fading. A single verifier may also have to accommodate a range of COTS devices that have various processing delays, so it becomes difficult to estimate an accurate $\hat{t}_p$. This issue is noted in some papers looking at distance-bounding protocols [18, 21]. Other issues with distance-bounding include vulnerability to various MITM attacks. For example, W. Guoheng *et al.* proposed a MITM attack that takes advantage of the lack of authentication that occurs in the initialization phase of many distance-bounding protocols, like HKP [24]. Their attack first gets the commitment from the prover, before initializing with the verifier. Others MITM attacks try to speed up the protocol by “guessing” the bit being sent, potentially reducing the total processing time in (3.1). Protocols like ASK have a flat voltage for the whole symbol, and can technically be deduced from the first time-instance. An attack could skip the rest of the message and quickly forward to the legitimate party, reducing the total processing time [23]. Negotiating and fixing $\hat{t}_p > t_p$ can be one method to minimize its direct effect, but this allows more potential time savings to be extracted by the attacker.

Ideally, to perform accurate RTT estimation for distances with digital communications, a fast symbol duration is required. Modern versions of distance-bounding attempt to overcome its timing issues by utilizing an ultra-wideband (UWB) implementation of some distance-bounding protocol. UWB utilizes short symbol duration to perform accurate RTT estimates of an exchange symbol, allowing for finer resolution estimates. According to the United States of America (USA) Federal Communications Commission (FCC) and the International Telecommunication Union Radiocommunication (ITU-R), UWB is defined as communications that utilize a bandwidth of 500 MHz or larger [102, 101]. The short symbol duration allows the transmission to be “multipath resistant,” meaning it is so short,

relative to the delay paths, that it is easy to distinguish the LOS term from any other path [107]. Furthermore, the symbol duration typically has timing error within a magnitude of the symbol duration, which could affect the estimated arrival time. Timing error of the signal can be off with an error within $\pm 0.1\%$ of the symbol duration. A shorter symbol would have a smaller magnitude error, which would correspond to a smaller distance error. Previous work by N. Tippenhauer *et al.* also noted that the shortening of the symbol duration in a distance-bounding bit exchange can provide protection against timing reduction attacks [98]. The Institute of Electrical and Electronics Engineers (IEEE) introduced the IEEE 802.15.4a ammendment to the IEEE 802.15.4-2006 specification for low rate wireless communications in 2007 [103, 104]. This was the first international standard specifying ultra-wideband ranging through the measurement of the RTT on the detected LOS term, which could be used as a less secure distance-bounding, as it does not have the secure bit exchange. In the years since, the standard has had little to no adoption by industry and consumers. With new demand from industry for distance measurements capabilities (e.g., location services, device tracking, etc.), with associated data transmissions, the new IEEE 802.15.4z-2020 ammendment looked to renew the push for a modern UWB system. The updated standard reduces the amount of messages that need to be sent, which provides energy savings for low powered devices. The standard also improves timing estimation and security by introducing the use of improved sequences, which allows for higher throughput communications [109]. Companies like Samsung and Apple, with their newest U2 chip, have increased the adoption of UWB devices, with the main service being device tracking. Work by P. Leu *et al.* has shown that high-rate pulse (HRP) mode of IEEE 802.15.4z-2020 (which is the major contribution of the standard) is vulnerable to range reduction attacks, but some recent work by K. Joo *et al.* has shown potential solutions to these problems [99, 100]. The low-rate pulse (LRP) mode from IEEE 802.15.4a is not vulnerable to the same attack, but has a trade-off in terms of even lower achievable data rates compared to the HRP mode.

One of the major issues with UWB based distance-bounding is that it inherently introduces complications for high-throughput communications. Due to the large bandwidth, UWB communications suffers significantly from frequency-selective fading and ISI, as detailed in Section 2.3.2. UWB also requires super or extremely high frequencies to accommodate the large bandwidth, without interference. Many physical problems present themselves at these frequencies, which has prevented millimeter wave (i.e., extremely high frequency) technologies from currently being adopted by consumers. These issues include path loss, penetration issues, and the infamous rain scatterer problem, which led to the lack of adoption for 5G millimeter wave technology [105, 106]. To compensate for these issues, MIMO systems must be used, which is not practical for the COTS devices that would

utilize distance-bounding. This also makes UWB distance-bounding likely impractical for devices that are far away from each (e.g., more than 100 metres). UWB requires fast clocks, very high sampling rate converters, and processors capable of processing the data fast enough, which are not cheap. Furthermore, a secure ASIC UWB distance-bounding capable device, that properly integrates a challenge-response authentication, would also likely require different hardware than what is currently found in most COTS devices [98]. Currently, COTS devices that are UWB capable cost more than 25 United States Dollars (USD), despite being on the market for years [108]. These prices are too costly for many of the type of devices that may utilize distance-bounding, like programmable key-fobs. The channel related issues caused by the frequency and bandwidth also make UWB a poor lowcost “dual-use” technology (i.e., devices intended for distance-bounding and high-throughput communications), as fading and ISI would ruin communications, before the path-loss issues are even accounted for.

While distance-bounding has primarily operated through ToF/RTT measurements, there are other methods to measure the distance between the legitimate communicating parties. Some papers propose using the received signal strength (RSS)[25] as a way to estimate a distance, since signal power is inversely proportional to distance. In fact, Bluetooth has a method to reject a minimum RSS value [75]. Unfortunately, it is trivial for the attacker to amplify the signals, to make them appear closer than they are. Similarly, the phase difference in a carrier is an accurate way to measure distance, but can also be manipulated [74]. Other distance-bounding methods include localization methods (e.g., GNSS) [29, 31]. If the legitimate parties could compare their coordinates, they might be able to figure out their distance. Unfortunately, localization does not work everywhere, and suffers from spoofing, replay, and even relay attacks [27]. Another proposed method for distance-bounding is the use of ambient noise and created audio noise [36, 26, 38]. W. Choi *et al.* proposed using smartphones to record and compare background noise as a method to verify distance [26]. In the case of vehicular relay attack prevention, the car would play a sound that only nearby phones could hear. The issue with sound based distance-bounding is that an attacker could “jam” (pun intended) with their own noise near both legitimate parties.

As mentioned, Bluetooth currently has the capability to use the RSS for distance measurement, but in 2024 the Bluetooth Special Interest Group (SIG) has proposed the usage of “channel sounding” as a future method for their devices to perform distance measurements [118, 119]. With the high cost of UWB devices and the current lack of major adoption, the Bluetooth SIG is looking to preempt the new market competition, with the hope that the lower cost and ubiquitous BLE devices could provide accurate distance measurements with the hardware that is already being manufactured. Their proposal to provide low

cost and precise distance measurement is to utilize the phase-shift between multiple pilot tones to estimate the time delay, which is an established method of distance estimation, as a portion of the phase-shift is proportional to the delay (2.4). Researchers have shown that phase based distance-bounding is not secure, as an arbitrary phase-shift can easily be applied through a digital filter, without a significant delay [116, 117]. Since the phase is easily manipulated by an active relay attacker (i.e., select an appropriate $\mathbf{h}_f$ in Figure 2.7), the Bluetooth SIG proposes using a secure frequency hopping of the pilot tones to add difficulty to the attacker, as they would have to track the appropriate frequency of the transmitted signal. As backstop, the Bluetooth SIG also gives the option to utilize RTT measurements, like in distance-bounding, but it would lack the benefits of UWB as Bluetooth symbols only occupy a 2 Mhz bandwidth. The Bluetooth SIG acknowledges that the low bandwidth would give an inaccurate RTT distance measurements, as this method is only used as a back-up or for fusion with the phase based method. While frequency hopping might be viable against attackers looking to utilize COTS hardware, Bluetooth only occupies 80 MHz while hopping, which would be trivial for an SDR to perform an amplify-and-forward active relay attack, by capturing all 80 MHz of the potential bandwidth and apply a phase-shift on the whole received signal, before retransmission. This brings into question the security and precision of the channel-sounding method proposed by the Bluetooth SIG as a distance-bounding method against a relay attack by a capable attacker, though it might be sufficient for non-adversarial distance measurements.

3.3 Contextualization

Beyond distance-bounding, the second most proposed method for preventing relay attacks fits under the umbrella of contextualization [32]. With contextualization, the two legitimate parties collect and compare local measurements from sensors to a physical context of their settings. If the context does not match what their expectations are, then they deem a relay attack is occurring. For example, a smartphone can use its ambient light sensor to detect if it is in a pocket or bag. If it is, then it can recognize that any NFC communications occurring may be happening due to a relay attack. Other sensors that could be used to gather context include: accelerometers, gyroscopes, magnetometers, barometers, and proximity sensors, etc. In general, contextualization requires feature extraction, training, and classification, to be able to gather context and detect when a relay attack may occur. T. Halevi *et al.* proposed using the accelerometers and magnetometer to determine the posture of the device [30]. By collecting training data, they can perform a cross-correlation with the previously trained data, then they classify the observation of the sensors of the

tag (prover), to determine if a relay attack is occurring. J. Li *et al.* proposed using barometer and accelerometer data to detect relay attacks against vehicle PKES with a smartphone [33]. Particularly, they check to see if the owner of the phone is approaching the car door, attempting to open the door, and closing the door, before unlocking and starting the vehicle. Another paper by J. Yafei *et al.* proposes that the legitimate devices compare the variation in the RSS due to movements during their communications [35]. Non-physical contexts like time can also be utilized. U. Ahmad *et al.* looked at using primarily the date and time to determine if a relay attack might be occurring between a smartphone and the PKES of a vehicle [28]. Active involvement in the communications can also be classified as contextualization. Google and Apple NFC payment methods require the user to unlock their phone before the communications begin. This can use both touch or facial recognition. Other works propose similar methods of measuring user interaction with devices [37].

While contextualization has been shown to be a promising method to deal with relay attacks, it is not applicable in every scenario. Many low powered and cheap devices do not have the necessary sensors to perform context detection. Until technologies like microelectromechanical systems (MEMS) become prolific, it is unlikely that low cost RFID devices will have the means to gather context. Furthermore, some devices may find themselves in situations where the context is consistent. If someone is already using their smartphone, it may already satisfy the context, despite not talking to the legitimate party. This could happen with fake proxies. For example, a fake PoS terminal that is a proxy in the relay attack may be set up to trick the user into recreating the correct context.

Another issue with contextualization is that the context can be manipulated by an active attacker. B. Shrestha *et al.* detail how pressure, temperature, humidity, and gas sensors can be manipulated with consumer devices [34]. The security of contextualization is dependent on how secure the sensors are. Work by M. Mohamed *et al.* showed that the management of sensor data on Android data is vulnerable to manipulation [39]. In the paper they exploit Android debugging tools to read and write to different sensor data files that should normally be inaccessible. Their attack is possible through the installation of a malicious application and access to the internet. They showed that sensor manipulation can be used to defeat gesture defenses that would be used in contextualization based solutions to the relay attack.

3.4 Physical-Layer Security

Both distance-bounding and contextualization look to gather and compare information that comes from an unrelayed source. While an attacker can forward analog and digital signals, they cannot substantially forward and presumably control a physical environment. F. Stajano *et al.* attempted to formalize this notion as the requirement for dealing with relay attacks [41]. In their work, they note that the only way to prevent the relay attack is to have an “unrelayed channel,” in which certain information can be shared by communicating parties, but cannot be relayed without losing its integrity. The unrelayed channel is not necessarily a communication link, but can also be a shared source of information between a group of listeners. For example, a wireless channel is an unrelayed channel within a certain distance and frequency. They note that there are four properties that are required in the communicating of information or in the collecting of common information, for a channel to be considered unrelayed. The required properties are: weak unclonability, strong unclonability, unsimulability, and untransportability. Unclonability should make it impossible (strong) or at least prohibitively (weak) difficult to copy the source. Another interpretation is that the source should be random and independent between different observation pairs. Unsimulability should make it prohibitively difficult to create a source. Specifically, the attacker should not be able to create a source of information or channel. Untransportability should make it prohibitively difficult to transfer the unrelayed source to another location. If an unrelayed channel or process is used for exchanging information, any discrepancies or oddities can be leveraged to detect relay attacks. For example, distance-bounding can be used as a source in an unrelayed channel. Unclonability, unsimulability, untransportability are met because the speed of light cannot be altered and any retransmission of the data would increase the delay. Unless there is some faster than light technology the attacker has access to, then there is a physical limit. The issue with distance-bounding is in its practicality, not its design.

In general, unrelayedness is tied to a physical phenomenon, whether it is a fundamental theory (like speed of light) or a reading from a sensor (contextualization). These phenomenon can be leveraged to prevent relay attacks. In fact, these sources could also be utilized to generate a key, if the unrelayed channel was also secure. Due to wireless channel correlation in space, time, and frequency, it is possible to also use the communication channel as a source for checking if a relay attack has occurred. A wireless channel can be used to generate symmetric-keys, also known as physical-layer key generation (PLKG). A PLKG protocol can achieve information theoretic perfect secrecy as a key source [42, 43, 44, 45, 46]. Furthermore, PLKG can be less power intensive compared to DH and RSA, which is an important consideration for low powered devices. Depending

on the communication systems, many devices already perform the steps that PLKG algorithms would utilize to generate the keys. For example, channel estimation and received signal strength (RSS) measurements are proposed in many PLKG protocols to provide security, which are already collected in IEEE 802.11 (WiFi), Bluetooth, and 4G/5G cellular systems. Using known information to derive symmetric-keys could become a trivial extra step in already established communication systems.

As mentioned, most of the implementation of PLKG protocols come from extracting the unrelayed randomness inherit to the large-scale and small-scale fading effects in communication channels [48, 49]. Shadowing in large-scale fading is considered to have a log-normal distribution, while small-scale fading has a Gaussian distribution due to multipath effects. Some work has been done on exploring large-scale fading for PLKG [50], but most works look at exploiting small-scale fading as it can change faster, which allows for a higher amount of information to be exchanged. From Section 2.3.2, the security of small-scale fading comes from multipath, spatial decorrelation, and channel reciprocity. The combination of these three effects mean that two parties can extract highly correlated random information from a channel over time, while a third party would extract something uncorrelated. Testing by J. Zhang *et al.* has shown that the theory matches with reality [51].

A symmetric-key can be produced by quantizing the random and correlated channel measurements that are collected by two communicating parties. These measurements are gathered in a noisy half-duplex exchange. Trying to generate as many bits as possible and as fast as possible from the exchange, while maintaining randomness and reciprocity, has been the basis of most research in PLKG. Most of the early and seminal works in the area looked at utilizing the RSS as a measurement of the channel, while exploring different quantization methods to generate faster keys [52, 53, 54, 55]. While RSS is an easy measurement to gather, it only provides one measurement at a time. Utilizing the CIR or the CFR has been shown to provide more information at each time sample [54, 56]. OFDM transmissions are often used as a method to get the CFR, and has provided the ability to get significantly more channel information than just the RSS [57, 67]. Multiple receive antennas that are spatially decorrelated can provide even more information [58]. While multiple transmit antennas at once do not provide extra information due to superposition, multiple antennas can reduce the effects of noise. To further improve the reciprocity after quantization, there can be a reconciliation process, that attempts to correct any discrepancies that may appear. While extracting more information per channel sample is important, the coherence time of the channel ends up being a significant limitation on the ability to extract fast key bits [65]. Most works claim that leakage of information from the reconciliation process can be accommodated through privacy amplification, though

the cryptographic sense of this is questionable [66]. If an input seed into a key-derivation function (KDF) is small, but secure, any larger key generated from the KDF is only as secure as the input seed. Many works have looked at how to extract more information in a slow-varying channel. Some works propose injecting noise, which is known as cooperative interference [59]. Other works introduce randomness through the use of a pre and post processing filter [60, 61]. Similar to the use of a processing filter, a intelligent reflective surface (IRS) can also be used to randomly change a channel faster than the inherit environment [62]. To evaluate the suitability of keys generated using physical-layer means, the key mismatch rate (KMR), key generation rate (KGR), and randomness are typically considered. The KMR looks at the hamming distance between keys generated by the two communicating parties. The KGR looks at the rate key bits are generated. The rate can expressed in *bits/symbol* or *bits/s*. Randomness can be evaluated by comparing generated key bits to randomness tests, such as the ones provided by the National Institute of Standards and Testing (NIST) [68].

One method to extract more information within a coherence time has been through the utilization of a decorrelating transform or filter. The usage of the Karhunen-Loeve transform (KLT) has been explored as a method to decorrelate time and frequency samples [61, 63, 64, 110]. The CFR has a distribution of $\sim \mathcal{CN}(\mathbf{0}_N, \Sigma_f)$ and $\sim \mathcal{CN}(\mathbf{0}_M, \Sigma_\tau)$ in frequency due to multipaths and time due to movement, respectively. The KLT is then equivalent to a whitening filter. The basis of the KLT is the use of the eigenvector decomposition of the covariance matrices $\Sigma_f = \Lambda_f \Phi_f \Lambda_f^{-1}$ and $\Sigma_\tau = \Lambda_\tau \Phi_\tau \Lambda_\tau^{-1}$ to decorrelate the measurements. For example, with temporal covariance, if the eigenvalues Φ_τ and eigenvector Λ_τ matrices are known, the identity matrix can be derived through the transform of $W = \Phi_\tau^{-1/2} \Lambda_\tau^T$. The temporal covariance is a circulant symmetric matrix, where the first row is a sample of the temporal autocorrelation function (ACF). Similarly, the frequency covariance is the Fourier transform of the power-delay profile (PDP) of the channel. Both of these are dependent on the channel conditions, specifically the coherence time and delay spread respectively, thus these cannot be known a priori. Estimating the covariance matrix is then the most important step for applying decorrelation transformations. How this is estimated must be parameterized by the amount of samples being correlated, and which samples are being correlated. Work by G. Li *et al.* looks at utilizing OFDM pilots to sample the CFR in frequency and time [61]. The correlated samples are reshaped into vectors to presumably be used for covariance estimation. How these are chosen is not quite defined, though they show good experimental results. Earlier work does not have practical methods for estimating the covariance matrix, as they assume they have many independent observations of the channel [63, 64]. R. Abubaker *et al.* proposed an automatic method to estimate the covariance matrix without any user input [110].

The ideas utilized to extract a shared symmetric-key from a wireless channel can also be applied towards preventing a relay attack. If two legitimate parties compare the samples of the wireless channel they receive, they could detect if a relay attack is occurring. To be more specific, a decode-and-forward relay attack could theoretically be detected because it strips the channel information between the proxies during the decoding process, as shown in Figure 2.3. If the channel transformations could be compared in each direction, it may be possible to determine if a relay attack is occurring. R. Abubaker explored this idea by leveraging a PLKG protocol, but tailored towards detecting decode-and-forward relay attacks [69]. Since the relay attack is utilized during a challenge-response authentication protocol, the legitimate parties already share a key for communications. This was leveraged to simplify the process compared to just a PLKG protocol, no longer needing to perform reconciliation or amplification. Work by C. T. Zenger *et al.* first proposed something along this line, where they extracted the small-scale fading from the RSS (a common method for PLKG) [70]. The focus on the paper was on a wormhole attack, which is a form of relay attack meant to trick devices in ad-hoc networks into creating an inefficient map of the network. The work was inspired by an earlier paper by S. Jain *et al.*, where they also used a physical-layer authentication to perform the wormhole attack detection [71]. The physical-layer utilized in the paper comes from another paper by P. Yu *et al.*, where the process is similar to MAC, but instead of appending the tag to the binary message, the tag is an additive symbol (like interference) [159]. This method is vulnerable to decode-and-forward and amplify-and-forward relay attacks.

It is important to note the distinction of distance, when considering the relay attack prevention methods. The distance between the communicating parties can influence the methods that can be utilized. For example, ToF/RTT distance-bounding and context based methods are (in theory) agnostic to the distance between the communicating parties, but channel based methods inspired from PLKG are dependent on far field conditions. Specifically, near field communications may not be able to utilize channel methods due to the weaker multipath effects. While multipaths may exist, a strong LOS term may dominate the received signal. Furthermore, far field communications display plane wave characteristics, which is a major assumption for multipath theory, which is not necessarily applicable to near field communications.

A proposed method for detecting relay attacks that is distance agnostic is the fingerprinting of wireless devices. Fingerprinting in a wireless device attempts to identify unique features of the analog signals generated by the specific transmitter of the device. This allows a verifier to perform entity authentication, without a cryptographic key. Relay attack detection may also be achieved, since a relay is a different device performing the transmissions to the verifier. The features used for fingerprinting are a result of minor

manufacturing defects and differences that create tolerances within every device. Ideally, even a device made in the same batch should have a tolerance in its features that could make them identifiable. This would be even more pronounced between devices made from different manufacturers. Different machine learning methodologies can be applied to classify these devices, which can be white-listed for future identification. Fingerprinting has primarily been proposed as a method to provide,

1. Localization: A device that has never been seen before and has a changing ID, can be tracked if it broadcasts messages (typical in Bluetooth, WiFi, and cellular communications),
2. Clone Detection: RFID devices can be cloned, extracting the capabilities of a device (e.g., for authentication), without having to use the device. Fingerprinting can be used to identify when a device is not the original,
3. Relay Attack Detection: Similar to clone detection for RFID devices, but is focused on relaying the capabilities of the devices,
4. Device Authentication: A device fingerprint could replace cryptographic entity authentication, as the physical device proves its identity through its intrinsic features (like a physical key).

B. Danev *et al.* claim to be the first to look at following these procedures to identify HF (passive) RFID devices at the physical level [76]. In the paper they look at the modulation shape (in time) and also the spectral features after performing the FFT. They classify these features by getting the Euclidean and Mahalanobis distance respectively from the previously reference of the devices. In general, these are not specific hardware features, but ambiguous results of the transmission. S. Periaswamy *et al.* looked at the power response of passive HF RFID devices (e.g., credit cards) [77]. Since the devices are passive, they get their power from the carrier of some active source. The power response is how much power the source needs to generate for the passive RFID device to respond (i.e., turn on). They classify devices by the power response to a range of frequencies, where they show through experimental results that backscattering is frequency dependent. An earlier paper by D. Zannetti *et al.* looks at the backscatter frequency response of passive UHF RFID tags (e.g., product tracking) [79]. J. Li *et al.* looks at the reflection coefficient (similar to power response) of passive UHF RFID tags at different frequencies [78]. Instead of sweeping multiple carriers, they send a single response generated through an OFDM symbol, which speeds up the feature extraction.

For far field devices there have been more features that have been identified. J. Hall *et al.* proposed looking at the transient between the noise to signal portion [81]. Particularly, they looked at the amplitude, frequency, phase, and wavelet transform of the transient. Classification was used by k-means clustering the extracted covariance and centroids of a profiled device. This work inspired others to explore the effectiveness of transients for other communication devices. K. B. Rasmussen *et al.* looked at transients for wireless sensor nodes [80]. They also consider the duration and number of peaks in the transient. Work by É. Helluy-Lafont *et al.* looked at the frequency offset, preamble duration, and hopping clock skew for fingerprinting a Bluetooth device [82]. The paper also details how temperature changes have a linear affect on the oscillator. In particular, frequencies decrease linearly with the temperature. They also identify two broad categories of features to fingerprint a device through wireless communications: modulation and waveform features. Modulation features look at how specific details of the information being sent change, while waveform features look at the small differences in the signals transmitted. In their paper they combine these two. Similar work by H. Givvehchian *et al.* used the I/Q Imbalance and carrier offset of BLE training samples to extract the Mahalanobis distance from a “clean” expected signal [83]. This paper was unique in that they were able to track devices in a public environment over time. Recent work by D. Guang *et al.* used SDRs to look at three different waveforms of other SDRs [88]. They specifically look at: a continuous wave, a chirp wave (a linear frequency sweeping), and a on-off continuous wave. They did not do it in a wireless setting (i.e., they used a wire to connect transmitter and receivers), so the only channel effect was AWGN.

Since there are numerous components and aspects that can be observed in an analog transmission, it is quite difficult to identify an optimal feature to fingerprint. Even weighting the different features has not been well explored. In recent years, neural networks have become the predominant method proposed in literature for fingerprinting devices. Trying to identify features that are sufficiently distinguishable can be difficult. Neural networks can abstract feature identification, while providing resilience to variation and noise, as long as sufficient training data is gathered. W. Lee *et al.* compared multiple deep neural network types, including a basic feedforward neural network (FNN), convolutional neural network (CNN), and a recurrent neural network (RNN) against other machine learning classifiers (support vector machine, random forest, and regression) [112]. Their results found that all the neural networks performed much better than the other machine learning methods, with the basic FNN having slightly better performance than the CNN and RNN. Other devices have also shown success fingerprinting using neural networks. P. Yin *et al.* proposed the first fingerprinting of LTE devices by the base station. Their motivation was to prevent denial-of-service (DoS) attacks from malicious devices, through the use of

a CNN on incoming time samples [111]. They focused on using the preamble (i.e., pilot) as the input. Y. Yuan *et al.* use a CNN (Lenet-5) to identify ZigBee devices through the capturing of their time samples [113].

The major issue with fingerprinting is that it is not well tested. Physical features may change over time and in different conditions (e.g., temperature, location, channel conditions, etc.). Long term testing is not something found in any of the papers identified, and needs to be expanded upon. Regardless of the fingerprinting method proposed, it is difficult to gather large and robust datasets in academia, and no datasets from industry are publicly available. This problem was identified by A. Elmaghubub *et al.*, where they noted cross domain identification significantly degrades performance [114]. They show that CNN based fingerprinting degrades significantly as the location changes from the original training position, even when moving just a metre away from the original position. Time was not as significant of a degradation, if the position stayed the same. They proposed observing more agnostic representations of received signals. Particularly, they created a transformation that is the envelope of the Hilbert transform of the scaled and delayed original inphase/quadrature (I/Q) signal. While they showed good results, they still only focus on a very narrow time and location variation (3 days and 3 metres). It is uncertain how this would hold in a totally different environment (e.g., outdoor versus indoor, LOS versus NLOS, etc.), without significantly more training. Another issue not explored with device fingerprinting, is how an amplify-and-forward relay attack would perform on them. It may be possible for some of the features to get captured by the relay’s signal, which could be sufficient for the neural network to classify. Other active relay attacks may also be used to modify a relay’s transmissions to make it look like the legitimate device.

While device fingerprinting could theoretically be used for relay attack detection to perform entity authentication, this has never been specifically tested in literature. Given some of the proposed problems, particularly the unreliability under different conditions and susceptibility to amplify-and-forward attacks, it is worthwhile to focus the usage of neural networks specifically on relay attack detection, instead of entity authentication, and utilize cryptography for authentication. Along this line, M. Fomichev *et al.* propose the first neural net use specifically for performing “copresence detection” (i.e., contextualization) for detecting relay attacks [115]. The paper looked at using a FNN for a binary classification of a distance-bounding. To do this they used the normalized channel-state information (CSI) extracted from WiFi packets as the input. They showed promising results, when compared to other contextualization methods, but they did not actually compare their results against an actual relay attack, only for distance detection. The key difference in this paper compared to the other neural net and contextualization works is that they focused on the wireless channel, rather than features of the device itself. While the wireless

channel is stochastic, it may have consistent qualities that are more consistent with the distance, like the coherence bandwidth. One potential issue with their proposal is that it could be possible for the relay to perform an active relay attack as described in Section 2.2. Since the pilots utilized in WiFi are publicly known, an attacker could estimate the channel and perform equalization to make the channel look a different way, potentially tricking the verifier’s neural network.

The paper that is most similar to the ideas proposed in this thesis is in recent work by P. Staat *et al.* [73]. In the paper they describe that they believe the only way to deal with a relay attack in all scenarios is to utilize a hybrid channel reciprocity and device fingerprint methodology. They do not describe a specific method for either, just go over the concepts of both, the strengths and weaknesses, and why they are needed for different situations. They state that much work needs to be done to figure out the best methods to implement and combine these methods, which is the basis for work in this thesis.

3.5 Motivation

The relay attack is a major issue for challenge-response authentication protocols that passively initiate through wireless communications. The consequences of the attack are pronounced for authentication protocols incorporated for cyber-physical systems, like vehicles, payment systems, and entry systems. Distance-bounding utilizing UWB technology is the current best proposed solution in literature, with industry seemingly leaning towards its adoption with the IEEE 802.15.4z standard. While work in literature has shown that it can be a solution for detecting relay attacks, there are trade-offs and limitations. Particularly, the need for dedicated hardware leading to increased costs, the short range of operation, and the limits on being able to perform high-throughput communications. Some of these concerns may have been the reasons for the lack of adoption from industry of the predecessor IEEE 802.15.4a standard, which already provided UWB distance measuring capabilities. Other methods like RSS and phase distance-bounding can be modified by an attacker, thus should not be considered secure. Localization is ironically vulnerable to relay attacks, as well as spoofing, making it unreliable for distance-bounding. Contextualization through sensors is promising, but is inflexible, can be manipulated, and is not currently feasible on every device. For example, credit cards do not have sensors to collect local information. Research into device fingerprinting through machine-learning for device authentication has grown over recent years, but has not been well explored in the context of relay attacks. Device fingerprinting suffers from overfitting to narrow dataset, while the lack of longterm and more challenging datasets do not invoke confidence. It may

also be possible for amplify-and-forward and active relay attacks to preserve the analog details of the original device, allowing attackers to circumvent device fingerprinting. New proposals for relay detection seems to be moving towards the training and usage of neural networks, but little work has explored the specific application to relay attacks, outside of device fingerprinting.

Utilizing channel state information (CIR or CFR) to determine if a relay attack can overcome many of the issues that are present with the other solutions. Channel state information is already utilized by most high-throughput systems already. Wireless communication systems like WiFi, 4G, and 5G already use channel estimation to equalize channel effects, which is a necessity for high-throughput wireless communications. These wireless communications systems (and others) are already ubiquitous in many devices at a low cost. Utilizing channel state information can provide an alternative to the current proposed methods to detect a relay attack, regardless of the type of relay attack being performed. At the moment, only 2 published papers explore using channel state information to detect relay attacks [70, 115]. One uses the RSS and another uses a neural network to perform distance-bounding. Neither paper specifically tested a relay attack, and neither paper considered the different attacker capabilities (i.e., decode-and-forward, amplify-and-forward, passive, and active relays). This leaves the exploration of relay attack detection through only the use of channel state information as an area of research with a lot of potential, and is the basis of motivation for this thesis. The novel protocols presented in this thesis only use the channel state information to detect the four relay attacks identified.

Chapter 4

One-Way Decode-and-Forward Relay Detection Protocol under Passive Attacks using Channel Reciprocity¹

In this chapter a novel one-way symmetric-key challenge-response protocol that can detect decode-and-forward relay attacks is described in detail and evaluated using simulations. A protocol that can detect only decode-and-forward relay attacks is useful against attackers that utilize COTS devices that have limited capabilities. Physical-layer signal processing is likely inaccessible to a COTS device, so upper-layer decoded messages are what are solely being forwarded. As detailed in Section 3.1, most practical attacks follow this format, as attackers may not have a deep understanding of the signals, protocols, and hardware components involved with the system under attack. For example, the scenario where attackers use an application on their smartphone to relay HF RFID/NFC challenge-response authentication is one of the most prominent attacks. In this scenario, the attacker is unlikely to even have access to the physical layer of their smartphones communications, let alone have the ability to interact with the ADC or DAC and introduce an adaptive filter into their smartphones communication chain.

The basis of this protocol is to utilize reciprocity that exists in device-to-device wireless communications to detect if a prover and verifier are talking directly to each other, or through a relay attack, which produces two uncorrelated channels. To determine if they are, the prover and verifier perform a pilot exchange, where the verifier uses a comparison

¹Most of this section comes from an unpublished paper available on IEEE TechRxiv by R. Abubaker and G. Gong and is to be submitted to a IEEE Transaction on Forensic Security [128]

of the securely shared information to determine if a relay attack is occurring. Specifically, from the mean square error (MSE) of the cross-correlation and autocorrelation, the verifier can make the determination.

4.1 Protocol Details

The following terminology is defined for use in the protocol:

- CK is an 128-bit encryption key generated from a symmetric-key K using a KDF and is only known by the prover and verifier,
- IK is an 128-bit authentication key generated from a symmetric-key K using a KDF only known by the prover and verifier,
- $\text{Enc}_{CK}\{\cdot\}$, $\mathbb{C}^N \rightarrow \mathbb{C}^N$, is a secure symmetric-key encryption that takes an input symbol and generates an encrypted output symbol using CK ,
- $\text{MAC}_{IK}\{\cdot\}$, $\mathbb{C}^N \rightarrow \mathbb{C}^N$, is a secure message authentication code that takes an input symbol and generates a tag output symbol using IK ,
- $\mathbf{p}_i \in \mathbb{C}^N$ is pilot sequence used by device $i = \{0, 1\}$, which contains the frequency sample and is publicly known,
- $\mathbf{h}_{i,m} \in \mathbb{C}^N$ is an FIR filter representing the channel seen by device $i = \{0, 1\}$ at the m^{th} time instance,
- $\mathbf{W}_m$ is a matrix of AWGN samples $\sim \mathcal{CN}(0, N_0)$.

Protocol 1 consists of 5 steps, which can be seen in Table 4.1. The details of these steps are given in the subsequent subsections.

4.1.1 Step 1: Initiation

The prover initiates the protocol by transmitting a request symbol $\mathbf{x}_{\text{req}} \in \mathbb{C}^N$ with a pilot in the preamble $\mathbf{p}_1 \in \mathbb{C}^N$. The request is known to the verifier. The pilot is also known between all parties, and is used to perform equalization and other signal processes needed to demodulate.

Table 4.1: The Steps and their Descriptions of the Protocol

step	Description
1	The prover initiates with a request $[\mathbf{p}_1, \mathbf{x}_{\text{req}}]$ to the verifier.
2	The verifier initiates a pilot exchange by transmitting a pilot $\mathbf{p}_2$ to the prover.
3	The prover encrypts and authenticates its pilots and received symbols to share with the verifier.
4	The verifier estimates both their channel and the prover's channel using the set of pilots and received symbols.
5	The verifier evaluates the peak-to-average cross-correlation ratio (PACR) to evaluate if a decode-and-forward attack has occurred.

4.1.2 Step 2: Channel Probing through a Pilot Exchange

After a request is received for an authentication, a pilot exchange begins, which is used to perform a probing of the wireless channel. The pilots are transmitted as vectors $\mathbf{p}_i = (p_i[0], p_i[1], \dots, p_i[N-1])$ and the channel is represented with an FIR filter $\mathbf{h}_{i,m} = (h_{i,m}[0], h_{i,m}[1], \dots, h_{i,m}[N-1])$, where $\mathbf{h}_{i,m} \in \mathbb{C}^N$ and is an approximate of the physical channel that is cut-off at the signal length in time (NT_s) at the m^{th} time instance. Note that both the channel and pilot are in the frequency domain. Given this, and the cut-off, the received frequency domain samples are given by,

$$\mathbf{r}_{i,m} = \mathbf{D}\{\mathbf{p}_i\}\mathbf{h}_{i,m} + \mathbf{w}_m,$$

where $\mathbf{w}_m \in \mathbb{C}^N$ has the distribution $\sim \mathcal{CN}(\mathbf{0}^N, N_0\mathbf{I})$

In this pilot exchange, the verifier and prover both select their own pilot symbol prior to the communications, which are fixed for each time instance m and publicly known. The pilots from the prover and verifier are transmitted M times in a half-duplex exchange. For example, at time instance m , the pilot $\mathbf{p}_i$ is sent from the i^{th} device, through the channel $\mathbf{h}_{i,m}$, and to the receiving device. After the probes from both parties have been transmitted, the verifier and prover will have the following set of information that is fixed for the rest of the protocol,

	Prover ($i = 1$)	Verifier ($i = 2$)
Received Symbols	$\mathbf{R}_1 = \mathbf{D}\{\mathbf{p}_2\}\mathbf{H}_1 + \mathbf{W}_1$	$\mathbf{R}_2 = \mathbf{D}\{\mathbf{p}_1\}\mathbf{H}_2 + \mathbf{W}_2$
Known Pilots	$\mathbf{p}_1, \mathbf{p}_2$	$\mathbf{p}_1, \mathbf{p}_2$

where $\mathbf{R}_i, \mathbf{H}_i, \mathbf{W}_i \in \mathbb{C}^{N \times M}$. The matrix channel matrix can be express as $\mathbf{H}_i = (\mathbf{h}_{i,0}, \mathbf{h}_{i,1}, \dots, \mathbf{h}_{i,M-1})$ and $\mathbf{W}_i = (\mathbf{w}_{i,0}, \mathbf{w}_{i,1}, \dots, \mathbf{w}_{i,M-1})$.

4.1.3 Step 3: Channel Feedback

After the pilot exchange occurs, the received symbols that the prover has are encrypted and authenticated using any authenticated encryption (e.g., AES-GCM) with the symmetric-key to prevent any modification or replay from an attacker. The encrypted and authenticated received symbols are only sent from the prover to the verifier in this one-way protocol. The data to be transmitted by the prover to the verifier is

$$\mathbf{x}_{\text{resp}} = \text{Enc}_{CK}(\mathbf{R}_1) || \text{Tag}_{IK}(\mathbf{R}_1), \quad (4.1)$$

where $\mathbf{x}_{\text{resp}} \in \mathbb{C}^{N \times}$ is the symbol representation of the encrypted binary data mapped to a symbol. The transmission can be done utilizing the established communication system or any other communication system, and is not considered in the details of this protocol. The scheme utilized to generate and exchange information may also reveal the channel state information, as an attacker cannot take advantage of this information any longer, as the prover and verifier have already established their respective channels in step 2 (4.1.2). The reveal by the prover is also encrypted and authenticated, preventing any tampering with the transmitted channel information. After the transmission the information that the prover and verifier have

	Prover ($i = 1$)	Verifier ($i = 2$)
Received Symbols	$\mathbf{R}_1$	$\hat{\mathbf{R}}_1, \mathbf{R}_2$
Known Pilots	$\mathbf{p}_1, \mathbf{p}_2$	$\mathbf{p}_1, \mathbf{p}_2$

where $\hat{\mathbf{R}}_1$ indicates that there may exist some error between the revealed information from the prover, which could occur during the transmission.

4.1.4 Step 4: Decode-and-Forward Relay Detection using the Cross-Correlation Evaluation

Two devices that are communicating between themselves using far field communications should observe a reciprocal channel that is highly correlated, except for the effects of noise and the Doppler spread (Section 2.4). Under a decode-and-forward relay attack, the channel effects and noise are stripped during the demodulation process between the legitimate parties (Section 2.2). This results in two uncorrelated channels existing between the prover and relay and between the verifier and relay. This protocol proposes observing the cross-correlation of the two channels that these legitimate parties see, to then determine if the devices are communicating directly to each other or through a relay. To perform the decode-and-forward relay detection, the mean square error (MSE) between the verifier's autocorrelation and the cross-correlation between the prover's CSI and the verifier's CSI is proposed. If the MSE of these correlations is above some threshold, then it is considered that a decode-and-forward relay attack is occurring (i.e., the channels are not likely correlated), otherwise a decode-and-forward relay attack is not occurring. The cross-correlation is calculated by the verifier after the reveal phase.

With the received information in Section 4.1.3, the verifier can estimate both parties channels using the LSE which provides an estimate of time instance of the channels, $\hat{\mathbf{h}}_{i,m} = (\hat{h}_{i,m}[0], \hat{h}_{i,m}[1], \dots, \hat{h}_{i,m}[N-1]) \in \mathbb{C}^N$, at the m^{th} time instance.

$$\hat{\mathbf{h}}_{i,m} = (\mathbf{D}^T \{\mathbf{p}_i\} \mathbf{D}^{-1} \{\mathbf{p}_i\})^{-1} \mathbf{D}^T \{\mathbf{p}_i\} \mathbf{r}_{i,m} \quad (4.2)$$

$$= \mathbf{D}^{-1} \{\mathbf{p}_i\} (\mathbf{D} \{\mathbf{p}_i\} \mathbf{h}_{i,m} + \mathbf{w}_m) \quad (4.3)$$

$$= \mathbf{h}_{i,m} + \mathbf{D}^{-1} \{\mathbf{p}_i\} \mathbf{w}_m, \quad (4.4)$$

where (4.3) comes from the symmetry in the diagonalized matrix and (4.4) comes from the invertibility of the diagonal matrix. Each estimate can be combined to get $\hat{\mathbf{H}}_i = (\hat{\mathbf{h}}_{i,0}, \hat{\mathbf{h}}_{i,1}, \dots, \hat{\mathbf{h}}_{i,M-1})$. The matrices can also be estimated directly

$$\hat{\mathbf{H}}_1 = \mathbf{D} \{\mathbf{p}_2\}^{-1} \mathbf{R}'_1 \quad (4.5)$$

$$= \mathbf{H}_1 + \mathbf{D} \{\mathbf{p}_2\}^{-1} \mathbf{W}_1, \quad (4.6)$$

and

$$\hat{\mathbf{H}}_2 = \mathbf{D} \{\mathbf{p}_1\}^{-1} \mathbf{R}_2 \quad (4.7)$$

$$= \mathbf{H}_2 + \mathbf{D} \{\mathbf{p}_1\}^{-1} \mathbf{W}_2. \quad (4.8)$$

Considering the temporal autocorrelation of the verifier's channel $\hat{\mathbf{H}}_2$, where the temporal channel is defined as

$$\hat{\mathbf{h}}_{2,n} = (\hat{h}_{2,0}[n], \hat{h}_{2,1}[n], \dots, \hat{h}_{2,(M-1)}[n]) \quad (4.9)$$

$$= (h_{2,0}[n], h_{2,1}[n], \dots, h_{2,(M-1)}[n]) \quad (4.10)$$

$$\begin{aligned} &+ \left(\frac{w_{i,0}[n]}{p_1[n]}, \frac{w_{2,1}[n]}{p_1[n]}, \dots, \frac{w_{2,(M-1)}[n]}{p_1[n]} \right) \\ &= (h_{2,0}[n], h_{2,1}[n], \dots, h_{2,(M-1)}[n]) \\ &+ \frac{1}{p_1[n]} (w_{i,0}[n], w_{2,1}[n], \dots, w_{2,(M-1)}[n]), \end{aligned} \quad (4.11)$$

which can be interpreted as the n^{th} row or n^{th} subcarrier of $\hat{\mathbf{H}}_2$. The temporal autocorrelation is,

$$C_{\hat{\mathbf{h}}_{2,n}}[l] = \mathbb{E}\{\hat{\mathbf{h}}_{2,n} \hat{\mathbf{h}}_{2,n+l}^H\} \quad (4.12)$$

$$= \mathbb{E}\left\{ \left(\mathbf{h}_{2,n} + \frac{1}{p_1[n]} \mathbf{w}_{2,n} \right) \left(\mathbf{h}_{2,n+l}^H + \frac{1}{p_1^*[n]} \mathbf{w}_{2,n+l}^H \right) \right\} \quad (4.13)$$

$$= \mathbb{E}\left\{ \mathbf{h}_{2,n} \mathbf{h}_{2,n+l}^H + \mathbf{h}_{2,n} \frac{1}{p_1^*[n]} \mathbf{w}_{2,n+l}^H + \right. \quad (4.14)$$

$$\left. \mathbf{h}_{2,n+l}^H \frac{1}{p_1[n]} \mathbf{w}_{2,n} + \frac{1}{p_1^*[n]} \frac{1}{p_1^*[n]} \mathbf{w}_{2,n} \mathbf{w}_{2,n+l}^H \right\}$$

$$= \mathbb{E}\{\mathbf{h}_{2,n} \mathbf{h}_{2,n+l}^H\} + \mathbb{E}\left\{ \mathbf{h}_{2,n} \frac{1}{p_1^*[n]} \mathbf{w}_{2,n+l}^H \right\} + \quad (4.15)$$

$$\mathbb{E}\left\{ \mathbf{h}_{2,n+l}^H \frac{1}{p_1[n]} \mathbf{w}_{2,n} \right\} + \mathbb{E}\left\{ \frac{1}{p_1^*[n]} \frac{1}{p_1^*[n]} \mathbf{w}_{2,n} \mathbf{w}_{2,n+l}^H \right\}$$

$$= \mathbb{E}\{\mathbf{h}_{2,n} \mathbf{h}_{2,n+l}^H\} + \frac{1}{p_1^*[n]} \mathbb{E}\{\mathbf{h}_{2,n}\} \mathbb{E}\{\mathbf{w}_{2,n+l}^H\} + \quad (4.16)$$

$$\frac{1}{p_1^*[n]} \mathbb{E}\{\mathbf{h}_{2,n+l}^H\} \mathbb{E}\{\mathbf{w}_{2,n}\} + \left| \frac{1}{p_1[n]} \right|^2 \mathbb{E}\{\mathbf{w}_{2,n} \mathbf{w}_{2,n+l}^H\}$$

$$= \mathbb{E}\{\mathbf{h}_{2,n} \mathbf{h}_{2,n+l}^H\} + \left| \frac{1}{p_1[n]} \right|^2 \mathbb{E}\{\mathbf{w}_{2,n} \mathbf{w}_{2,n+l}^H\} \quad (4.17)$$

$$= C_{\mathbf{h}_{2,n}}[l] + \frac{1}{\gamma_n} \boldsymbol{\delta}, \quad (4.18)$$

where $C_{\mathbf{h}_{2,n}}[l]$ is the temporal autocorrelation of the channel, γ_n is instantaneous SNR at the n^{th} time sample, and $\boldsymbol{\delta} = (1, 0, \dots, 0)$ is the discrete dirac-delta vector. The temporal autocorrelation $C_{\mathbf{h}_{2,n}}[l]$ would depend on the signal.

Following a similar process to (4.18) for the cross-correlation between the verifier's channel and prover's channel

$$C_{\hat{\mathbf{h}}_{2,n}, \hat{\mathbf{h}}_{1,n}}[l] = E\{\mathbf{h}_{2,n} \mathbf{h}_{1,n+l}^H\} + \left| \frac{1}{p_1[n]} \right| \left| \frac{1}{p_2[n]} \right| E\{\mathbf{w}_{2,n} \mathbf{w}_{1,n+l}^H\} \quad (4.19)$$

$$= E\{\mathbf{h}_{2,n} \mathbf{h}_{1,n+l}^H\} \quad (4.20)$$

$$+ \left| \frac{1}{p_1[n]} \right| \left| \frac{1}{p_2[n]} \right| E\{\mathbf{w}_{2,n}\} E\{\mathbf{w}_{1,n+l}^H\} \\ = C_{\mathbf{h}_{2,n}, \mathbf{h}_{1,n}}[l] \quad (4.21)$$

where line (4.19-4.20) comes from the uncorrelated assumption that noise is independent between the 2 channels. If a decode-and-forward relay attack is occurring

$$C_{\hat{\mathbf{h}}_{2,n}, \hat{\mathbf{h}}_{1,n}}[l] = C_{\mathbf{h}_{2,n}, \mathbf{h}_{1,n}}(l) \quad (4.22)$$

$$= E\{\mathbf{h}_{2,n}\} E\{\mathbf{h}_{1,n+l}^H\} \quad (4.23)$$

$$= \mathbf{0}^M. \quad (4.24)$$

Since the channels have a Rayleigh distribution, they are uncorrelated and must be independent as well. Every term also has 0 mean and is WSS.

For the cross-correlation, if there is no relay attack, then

$$C_{\hat{\mathbf{h}}_{2,n}, \hat{\mathbf{h}}_{1,n}}[l] = C_{\hat{\mathbf{h}}_{2,n}}(l) - \frac{1}{\gamma_n} \boldsymbol{\delta} \quad (4.25)$$

$$= C_{\mathbf{h}_{2,n}}[l] \quad (4.26)$$

where (4.26) differs from (4.24), as the noises are uncorrelated (independent), so it does not appear.

To determine if a decode-and-forward attack is occurring, use the normalized mean-square error (MSE) between the autocorrelation (4.18) and cross-correlation (4.24) or (4.26) as an indicator for thresholding by the verifier. Furthermore, we ignore the first sample ($l = 0$) in the correlation to filter out the effects of noise, as shown in (4.24).

$$\text{MSE} = \frac{1}{N|C_{\hat{\mathbf{h}}_{2,n}}[1]|} \sum_{n=0}^{N-1} \sum_{l=1}^{2M-1} |C_{\hat{\mathbf{h}}_{2,n}}[l] - C_{\hat{\mathbf{h}}_{2,n}, \hat{\mathbf{h}}_{1,n}}[l]|^2 \quad (4.27)$$

If there is no decode-and-forward relay attack

$$\text{MSE} = \frac{1}{N|C_{\hat{\mathbf{h}}_{2,n}}[1]|} \sum_{n=0}^{N-1} \sum_{l=1}^{2M-1} \frac{1}{\gamma^2} \boldsymbol{\delta} \\ = 0 \quad (4.28)$$

If there is a decode-and-forward relay attack

$$\begin{aligned}
\text{MSE} &= \frac{1}{N|C_{\hat{\mathbf{h}}_{2,n}}(1)|} \sum_{n=0}^{N-1} \sum_{l=1}^{2M-1} |C_{\mathbf{h}_{2,n}}[l] + \frac{1}{\gamma}\delta|^2 \\
&= \frac{1}{N|C_{\hat{\mathbf{h}}_{2,n}}[1]|} \sum_{n=0}^{N-1} \sum_{l=1}^{2M-1} |C_{\mathbf{h}_{2,n}}[l]|^2 \\
&> 0,
\end{aligned} \tag{4.29}$$

which gives a running sum of the absolute value squared of the temporal autocorrelation. This will be some strictly positive value as any given channel will have some non-zero autocorrelation given by the Dopplers frequency, unless it is moving extremely fast relative to the pilot exchange rate.

Verification passes against a decode-and-forward relay attack if $\text{MSE} \leq Th$, which implies that $\hat{\mathbf{H}}_1 \approx \hat{\mathbf{H}}_2$

$$\text{MSE} \leq Th \iff \hat{\mathbf{H}}_1 \approx \hat{\mathbf{H}}_2 \tag{4.30}$$

The inverse is also implied

$$\text{MSE} > Th \iff \hat{\mathbf{H}}_1 \not\approx \hat{\mathbf{H}}_2 \tag{4.31}$$

which means the communications are considered to be under a decode-and-forward relay attack, as the channels between the prover and verifier are not reciprocal.

4.1.5 Step 5: Challenge-Response Authentication

The protocol presented provides a challenge-response authentication implicitly through the relay attack detection processes. Due to the reciprocity established in the detection phase, the randomness of the channels, and the use of the session keys, CK and IK , a unique challenge and response is provided through the physical-layer.

Following the assumption that the channel is unique, the channel can act as a secure and fixed provider of a challenge to the prover, which the verifier also receives. An idealized representation of this can be seen in Figure 4.1

When performing the reveal phase of the protocol, the prover encrypts and authenticates the data with sessions keys derived from the shared key. If the verifier is capable of decrypting, authenticated, and passing the decode-and-forward test in the cross-correlation phase, then this is sufficient evidence that the user has ownership of the symmetric-key and that a replay attack is not occurring. If they did not have the key, then (4.1) would not

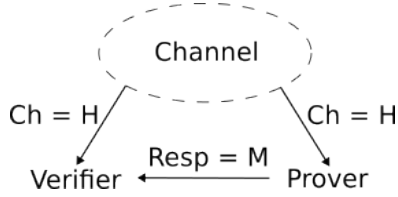


Figure 4.1: Challenge-Response Authentication

be authenticated and its decrypted value would not pass the decode-and-forward stage. If the attacker was performing a replay attack, then there would be 2 different channels and the relay detection would fail.

A typical challenge-response authentication protocol, would require the verifier to generate a challenge and transmit within the pilot exchange (wasting bandwidth), or to send the challenge after the initial protocol (wasting time). The prover would have to send a response in a similar manner. The usage of the channel for the challenge does not waste subcarriers during the pilot exchange, and saves extra steps that would be required in a subsequent response.

4.2 Feedback Size and Complexity

During the transmission of the channel in step 3 (Section 4.1.3), the symbols are preserved in their in-phase and quadrature (I/Q) form, thus they are transmitted as complex numbers. For example, if the received symbols and pilots are processed as L -bit floating-point numbers, then that is how they should be transmitted. The number of data bits that need to be transmitted is given by

$$\text{Bits Transmitted} = 2LNM, \quad (4.32)$$

where N and M are the number of subcarriers and number of OFDM pilot exchanges, respectively. For example, if $L = 16$ bits, $N = 64$, and $M = 100$, then the data bits transmitted from the prover to the verifier is ≈ 25.6 KB.

The protocol estimates the autocorrelation and cross-correlation using the power spectral density estimate (PSD). Using the FFT, each process would have a running time of $O(N \ln N)$. The MSE has a linear running timing in N , since it is calculated using the average difference, so the running time is a lower growth than the FFT. Each pilot exchange results in a single PSD estimate and MSE, which are then all averaged. The running time for the average is $O(M)$.

4.3 Simulation Experimentation

This section looks at the Monte-Carlo results from the simulations. The performance metrics that are observed are the false-positive rate and the false-negative rate of the ability to detect a decode-and-forward relay attack.

The false-positive rate (FPR) is defined as the number of decode-and-forward attacks indicated when none have occurred, out of a total set of none relayed authentications.

$$\begin{aligned} \text{FPR} &= \frac{\#FP}{N_{CA}} \\ &= 1 - \frac{\#Passes}{\#Passes + \#Fails}, \end{aligned} \quad (4.33)$$

where $\#FP$ is the number of false-positives and N_{CA} is the total number of authentications with no relaying.

The false-negative rate (FNR) is the number of attacks missed, out of a total set of relayed authentications.

$$\begin{aligned} \text{FNR} &= \frac{\#FN}{N_R} \\ &= 1 - \frac{\#Fails}{\#Passes + \#Fails}, \end{aligned} \quad (4.34)$$

where $\#FN$ is the number of false-negatives and N_R is the total number of authentications with relaying.

For both the FPR and FNR a lower value is desired for the protocol to show that it can discern relay attacks from expected authentication, as this is their objective. The simulation was performed in the MATLAB programming language running on a computer using an i9-10900F processor and 64 GB of RAM on Ubuntu 22.04. The simulation uses baseband sampling with perfect timing synchronization and no transceiver non-idealities. The channel was simulated using the uncorrelated scattering model, where each tap has a Rayleigh distribution and is correlated in time using the uniform scattering model. The temporal correlation for the inphase and quadrature component of each tap is generated through Jake's method. Feedback is automatically assumed to be available to the verifier and is not considered in the this simulation (or protocol).

The parameters for the simulation are shown in Table 4.2, the pilot-noise SNR at the verifier plays a role in the signal, and is the main independent variable in the simulations. The threshold for the MSE (Th) was selected heuristically from observing simulation experiments. Ideally, the threshold could be selected by observing a receiver operating

characteristic (ROC) curve [160]. The ROC curve compares the true-positive rate (which corresponds to the false-negatives rate) against the false-positive rate, over different thresholds. This would show how different threshold cause a performance trade-off between the false-negative and false-positive rates. Depending on how weighting is given to the rates, the optimal threshold could be selected. How the false-positive and false-negative rates are weighted will depend on the priority of the system to defend against a relay attack versus guaranteeing accessibility to the user, respectively. The scaled correlation from (4.27) was found to be relatively large (close to 1) under a decode-and-forward relay attack, while it is very small ($< 10^{-5}$) when no relay attack occurs. Each simulation at every SNR was repeated 1000 times.

Table 4.2: DF Simulation parameters

	Parameter	Value
1	Symbol Duration (μs)	1
2	Distance (m)	1
3	Carrier Frequency (GHz)	2.4
4	RMS Delay Spread (ns)	250
5	SNR (dB)	-
6	Number of Exchanges (#)	100000
7	Number of Pilots (#)	64
8	Speed (m/s)	1
9	Power-Delay Profile	Exponential
10	Th	0.01

The results for the false-positive and false-negative rates can be seen in Figure 4.2 and Figure 4.3 respectively.

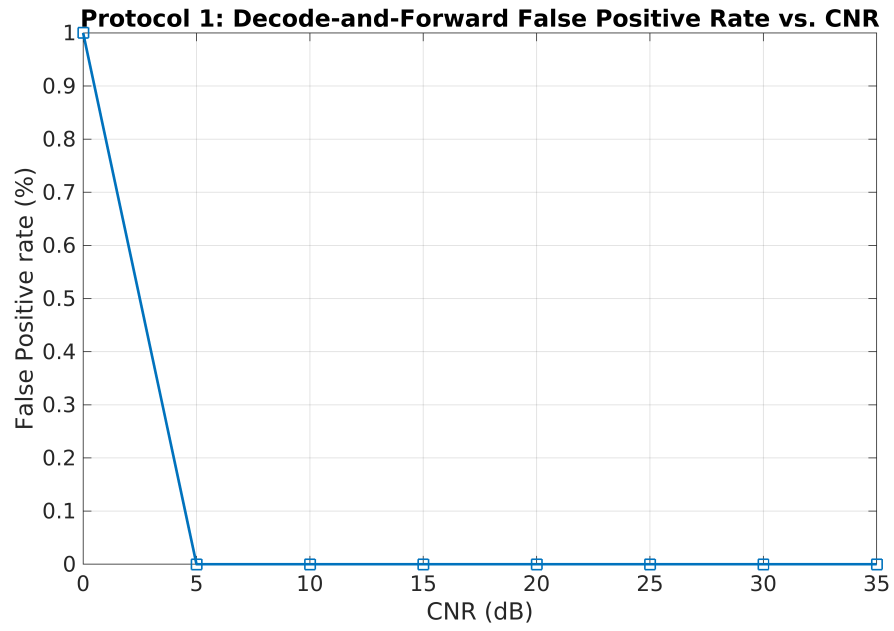


Figure 4.2: false-positive rate for detecting DF relay attacks versus CNR

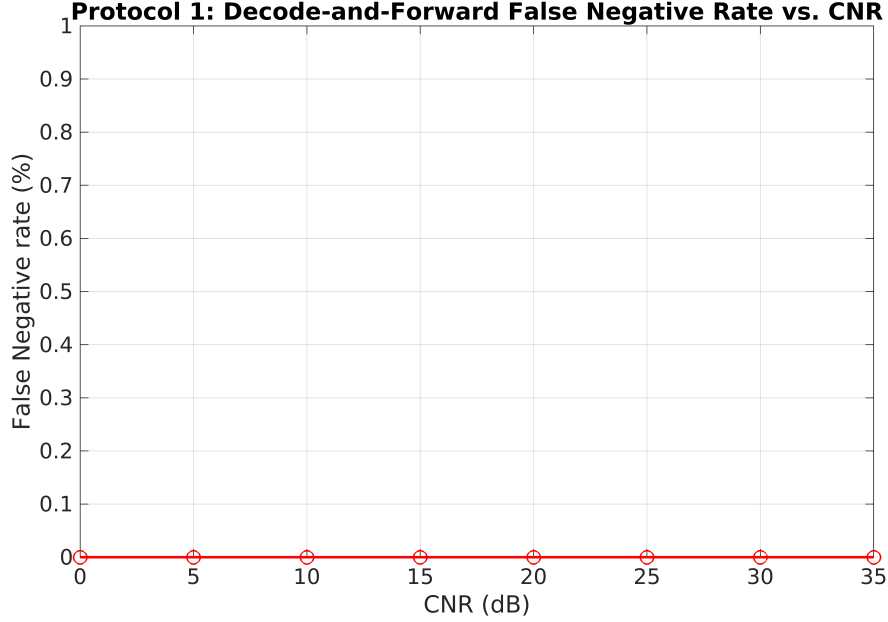


Figure 4.3: false-negative rate for detecting DF relay attacks versus CNR

The results for both the FPR and FNR is very low, regardless of the SNR, except when the SNR is 0 dB. This corroborates our theoretical derivations in (4.28) and (4.29), which shows that noise should play a limited role, as the first sample in the autocorrelation is skipped, which is where the noise power exists. The case of 100% false-positive at 0 dB may be a result of a poor correlation estimate, but it was consistent over every simulation experiment that was performed did.

4.4 Active Relay Attack

The theoretical weakness in this protocol comes from the vulnerability to an active relay attack. Since the pilots $\mathbf{p}_1$ and $\mathbf{p}_2$ are publicly known, it is possible in step 2 (4.1.2) for an attacker to perform the decode-and-forward active relay attack detailed in Section 2.2. Specifically, during the request and pilot exchange, the relay should be able to get an estimate of $\mathbf{h}_{1,m}$ and $\mathbf{h}_{2,m}$ since reciprocity exists between the prover and relay and between the verifier and relay, respectively. While the pilot exchange is occurring, the relay can then apply an active filter $\mathbf{h}_f = \mathbf{D}^{-1}\{\hat{\mathbf{h}}_{1,m}\}\hat{\mathbf{h}}_{2,m}$ when relaying the verifiers pilot $\mathbf{p}_2$ to the

prover. This results in the prover estimating $\mathbf{h}_{2,m}$, instead of $\mathbf{h}_{1,m}$, which is the reciprocal channel, despite a decode-and-forward relay attack occurring.

Chapter 5

One-Way Decode-and-Forward Relay Detection Protocol under Active and Passive Attacks

In this chapter a novel one-way symmetric-key challenge-response entity authentication protocol is detailed. Compared to the Protocol 1 in Chapter 4, this protocol can detect passive and active decode-and-forward relay attacks, while similarly providing implicit entity authentication. The protocol is also more robust in the channels conditions that it can work with, as it does not require the assumption of channel reciprocity found in other physical-layer solutions. Furthermore, it generally does not depend on the channel distribution. For example, it does not require a Rayleigh channel to function. Similar to Protocol 1, the protocol must utilize a feedback mechanism dependent on the symmetric-key. The only limitation in the protocol is the near field active attack scenario, which is discussed in Chapter 8.

The main idea of the protocol is to utilize a pseudorandom FIR filter generated by the prover and verifier to obscure the channel through a “multiplicative encryption,” which prevents the relay from discovering the channel. This allows for the prevention of active attacks, as the attacker cannot discern the generated filtered from the real channel. The random filter is also used to generate a challenge by the verifier, where only the prover could extract the actual channel, if they have the correct symmetric-key.

5.1 Protocol Details

For this protocol, the following terminology is defined for use in the protocol:

- CK is an 128-bit encryption key generated from a symmetric-key K using a KDF and is only known by the prover and verifier,
- IK is an 128-bit authentication key generated from a symmetric-key K using a KDF only known by the prover and verifier,
- $\text{Enc}_{CK}\{\cdot\}, \mathbb{C}^N \rightarrow \mathbb{C}^N$, is a secure symmetric-key encryption that takes an input symbol and generates an encrypted output symbol using CK ,
- $\text{MAC}_{IK}\{\cdot\}, \mathbb{C}^N \rightarrow \mathbb{C}^N$, is a secure message authentication code that takes an input symbol and generates a tag output symbol using IK ,
- $\mathbf{p} \in \mathbb{C}^N$ is pilot vector, which contains the frequency sample and is publicly known,
- $\mathbf{h}_{xy,m} \in \mathbb{C}^N$ is an FIR filter representing the channel from y to x at the m^{th} time instance,
- $\mathbf{h}_k \in \mathbb{C}^N$ is an FIR filter defined as the keyed filter that is generated using CK and an encryption function to generate pseudorandom bitstream that is then quantized,
- $\mathbf{h}_{ch} \in \mathbb{C}^N$ is an FIR filter defined as the challenge filter that is randomly generated complex sequence by the verifier,
- $\mathbf{W}_m$ is a matrix of AWGN samples $\sim \mathcal{N}(0, N_0)$.

The protocol when no decode-and-forward attack is occurring can be seen in Figure 5.1.

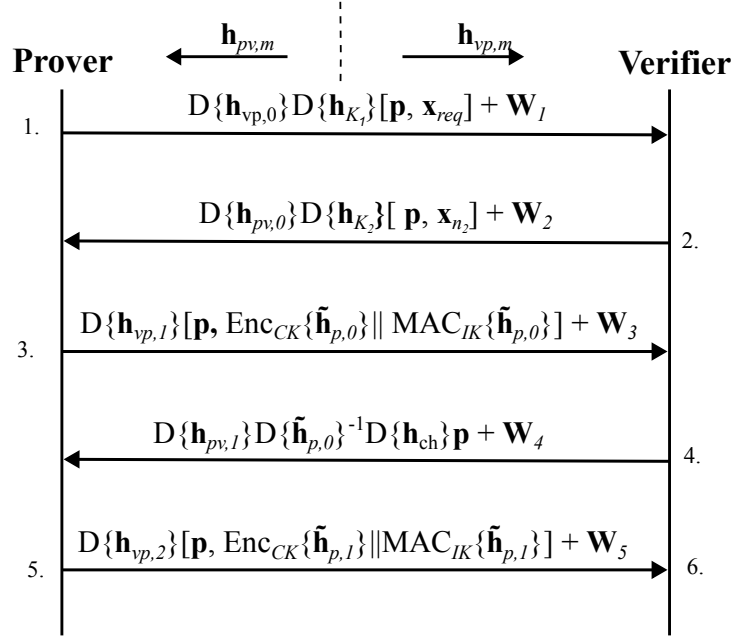


Figure 5.1: Protocol with no relaying

The protocol detects a decode-and-forward relay attack by assuming that the first hop CFR seen by the prover is the same first hop CFR from the verifier's transmission. If a relay attack is being mounted, then this does not hold, and the mean-square error (MSE) of the challenge will become large. The symmetric-keys are used to provide feedback, challenge-response authentication, and to generate random filters. If any attacker tries to interact with the system without knowledge of the keys, then this will manifest as error, which will also increase the MSE. The MSE is how both authentication and relay detection are checked by the verifier. The protocol under a decode-and-forward relay attack can be seen in Figure 5.2.

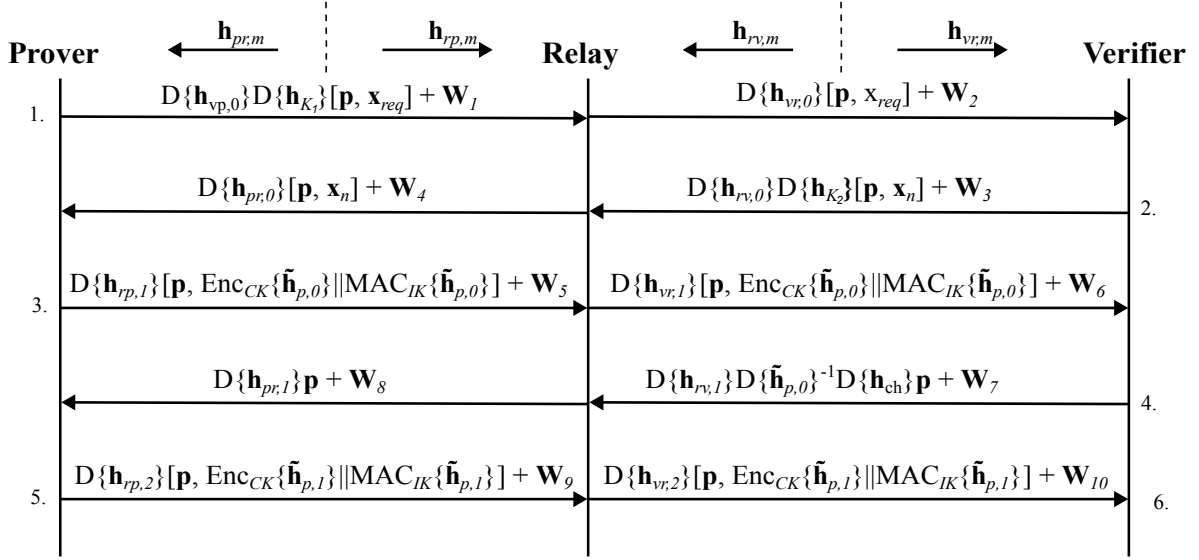


Figure 5.2: Protocol with a decode-and-forward relay

Table 5.1: Steps of the Channel Based Relay Attack Detection Protocol

step	Details
1	Prover initiates protocol by transmitting request and a nonce $\mathbf{x}_{req}$ filtered through a random filter $\mathbf{h}_{K_1}$.
2	Verifier confirms request and generates a nonce $\mathbf{x}_{n_2}$ and a filter $\mathbf{h}_{K_2}$ using CK and $\mathbf{x}_{n_2}$. The nonce is transmitted through the filter to the prover
3	Prover uses the received nonce $\mathbf{x}_n$ and CK to generate $\mathbf{h}_{K_2}$ and equalizes the received pilots and estimates $\mathbf{h}_{pv,0}$ as $\tilde{\mathbf{h}}_{pv,0}$. Prover encrypts $\tilde{\mathbf{h}}_{pv,0}$, generates a MAC using CK and IK , and transmits both.
4	Verifier generates a random challenge filter $\mathbf{h}_{ch}$ and transmits the pilots through this filter while applying equalization using $\hat{\mathbf{h}}_{p,0}$
5	Prover estimates the challenge filter $\mathbf{h}_{ch}$, as $\hat{\mathbf{h}}_{p,1}$, encrypts it, and transmits it with a MAC
6	Verifier compares $\mathbf{h}_{ch}$ and $\hat{\mathbf{h}}_{p,1}$ using the MSE. If it is above a threshold Th , then a relay attack is occurring and authentication fails, otherwise no relay attack is occurring and authentication passes

In this scenario, the relay will not only be unable to properly recover its first hop channel at step 3, but the equalization done by the relay at step 4 will not match the first hop channel to the relay. The filters $\mathbf{h}_{K_1}$ and $\mathbf{h}_{K_2}$ prevent the relay from estimating the channel $\mathbf{h}_{rp,0}$ and $\mathbf{h}_{rv,0}$, respectively, which prevents active decode-and-forward relay attack, detailed in Section 2.2. A graphical summary of the steps in the protocol can be seen in Figure 5.3.

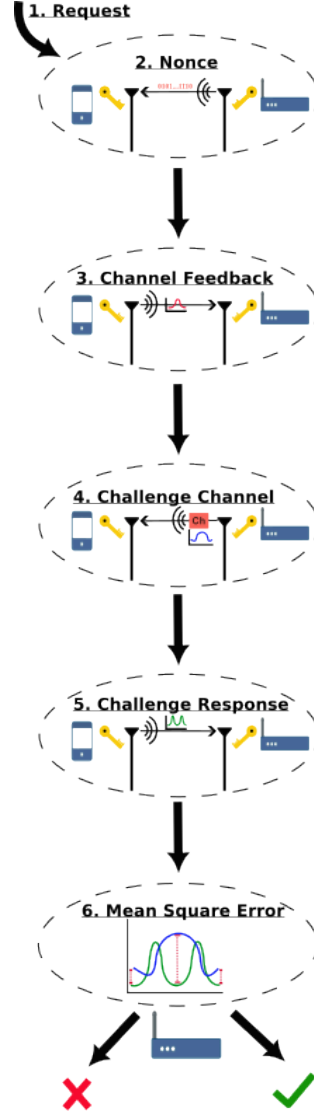


Figure 5.3: Graphical summary of the steps of the proposed protocol

The rest of this section is dedicated to detailing each step of the protocol and how security is provided.

5.1.1 Step 1: Initiation

In the first step, the prover initiates the protocol by sending a request symbol transmission $\mathbf{x}_{req}$ to the verifier. The first transmission with a pilot must occur from the prover to the verifier, as the verifier must not reveal any channel state information to a potential relay before the usage of a random keyed filter. The symbol $\mathbf{x}_{req}$ is a fixed OFDM data symbol in the frequency domain.

The prover also filters the transmission using a random filter $\mathbf{h}_{K_1}$. This filter is necessary if the channels are reciprocal. Otherwise, the attacker can estimate $\mathbf{h}_{pv,m}$ by assuming $\mathbf{h}_{pv,0} \approx \mathbf{h}_{vp,0}$. This problem is expanded upon in the subsequent subsections.

5.1.2 Step 2: Nonce Transmission

After the verifier receives a request, it will check that the demodulated symbol matches the expected request $\mathbf{x}_{req}$. If it does, then the verifier will generate a random nonce that has ≥ 128 -bit length, where the length is known to both parties beforehand. The nonce will be used to provide the challenge in the challenge-response authentication. Specifically, the nonce is the seed or initial value (IV) utilized for the stream cipher encryption and MAC generation in subsequent steps. The nonce is transmitted in an OFDM symbol $\mathbf{x}_n$.

The second goal of this step is for the verifier to generate a random keyed filter ($\mathbf{h}_{K_2}$) using the encryption key (CK) and the nonce. The purpose of the keyed filter is to prevent the relay from recovering the channel from the verifier to relay. Figure 5.4 shows the difference in step 2 when the decode-and-forward relay attack occurs and does not occur.

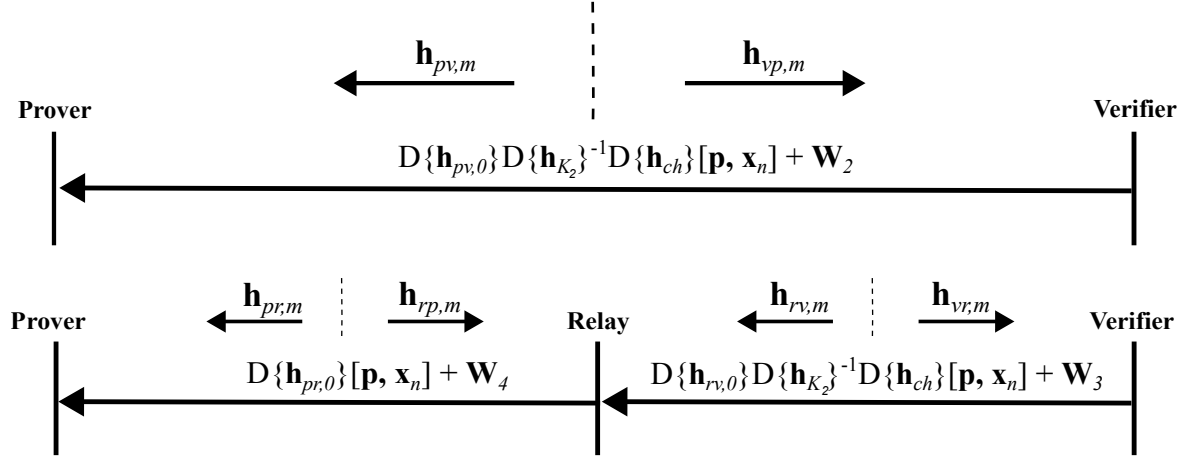


Figure 5.4: Step 2: with (bottom) and without (top) a decode-and-forward relay attack

When no relay attack is occurring, the prover estimates the channel

$$\hat{\mathbf{h}}_{p,0} = \mathbf{D}\{\hat{\mathbf{h}}_{pv,0}\} \hat{\mathbf{h}}_{K_2}. \quad (5.1)$$

From this point, the prover can utilize $\mathbf{x}_n$ and CK to generate $\mathbf{h}_{K_2}$. Using the keyed filter and $\hat{\mathbf{h}}_{p,0}$, the prover can extract an estimate of $\mathbf{h}_{pr,0}$,

$$\begin{aligned} \tilde{\mathbf{h}}_{p,0} &= \mathbf{D}\{\mathbf{h}_{K_2}\}^{-1} \hat{\mathbf{h}}_{p,0} \\ &= \mathbf{D}\{\mathbf{h}_{K_2}\}^{-1} \mathbf{D}\{\hat{\mathbf{h}}_{pv,0}\} \hat{\mathbf{h}}_{K_2} \end{aligned} \quad (5.2)$$

$$= \mathbf{D}\{\mathbf{h}_{K_2}\}^{-1} \mathbf{D}\{\hat{\mathbf{h}}_{K_2}\} \hat{\mathbf{h}}_{pv,0}, \quad (5.3)$$

where (5.2-5.3) follows from commutativity of the diagonalization. The estimate holds, if

$$\mathbf{D}\{\mathbf{h}_{K_2}\}^{-1} \mathbf{D}\{\hat{\mathbf{h}}_{K_2}\} \approx \mathbf{I}. \quad (5.4)$$

The estimate ($\tilde{\mathbf{h}}_{p,0}$) of the channel from the verifier to the prover is used in subsequent steps to test the first hop channel from the verifier, and to provide feedback if reciprocity does not hold.

On the other hand, if a decode-and-forward relay attack is occurring, the prover's channel ($\mathbf{h}_{pr,0}$) is not the same as the channel from the verifier to the relay. The resultant cumulative channel from the verifier to the relay that is estimated is

$$\hat{\mathbf{h}}_{r,0} = \mathbf{D}\{\hat{\mathbf{h}}_{rv,0}\} \hat{\mathbf{h}}_{K_2} \quad (5.5)$$

Despite knowledge of $\mathbf{h}_{rp,0}$, from step 1, without knowledge of $\mathbf{h}_{K_2}$, it should be prohibitively difficult for the relay to estimate $\mathbf{h}_{rv,0}$ from $\hat{\mathbf{h}}_{r,0}$ and to perform the active decode-and-forward relay attack described in Section 2.2. Furthermore, the channel estimate that the prover estimates is

$$\hat{\mathbf{h}}_{p,0} = \hat{\mathbf{h}}_{pr,0}, \quad (5.6)$$

This causes the estimate channel to become

$$\begin{aligned} \tilde{\mathbf{h}}_{p,0} &= \mathbf{D}\{\mathbf{h}_{K_2}\}^{-1} \hat{\mathbf{h}}_{pr,0} \\ &= \mathbf{D}\{\mathbf{h}_{K_2}\}^{-1} \hat{\mathbf{h}}_{pr,0}, \end{aligned} \quad (5.7)$$

which will cause the estimate to differ to what the real channel $\mathbf{h}_{pr,0}$ is, due to the keyed filter. When comparing (5.3) to (5.7), the estimated channel will vary significantly, which will effect the final MSE estimation and thresholding.

5.1.3 Step 3: Channel Feedback

In this step, the prover provides feedback to the verifier of the estimated channel $\tilde{\mathbf{h}}_{p,0}$. It is critical that the feedback is encrypted using a strong stream cipher and authenticated using a MAC. Encryption is required to prevent the attacker from eavesdropping the data. With no encryption, the attacker can perform an active filter, which results in the channel appearing like no relay attack is occurring. This is shown in Section 5.1.4. The MAC is applied for precautionary reasons to prevent the attacker from modifying the estimate.

5.1.4 Step 4: Relay Challenge

In this step, after decrypting and verifying the feedback, the verifier uses $(\tilde{\mathbf{h}}_{p,0})$ to challenge the prover to confirm that it is directly talking to the verifier across a single channel, and not a decode-and-forward relay. Figure 5.5 shows the difference in step 4 when the decode-and-forward relay attack occurs and does not occur.

When a decode-and-forward relay attack is not occurring, the verifier filters the transmission to equalize the channel from the verifier to the prover. If the steps have happened fast enough, then the estimate $\tilde{\mathbf{h}}_{p,0}$ should be close to $\mathbf{h}_{pv,1}$, thus

$$\begin{aligned} \tilde{\mathbf{h}}_{p,1} &= \mathbf{D}\{\hat{\mathbf{h}}_{pv,1}\} \mathbf{D}\{\tilde{\mathbf{h}}_{p,0}\}^{-1} \hat{\mathbf{h}}_{ch} \\ &\approx \hat{\mathbf{h}}_{ch} \end{aligned} \quad (5.8)$$

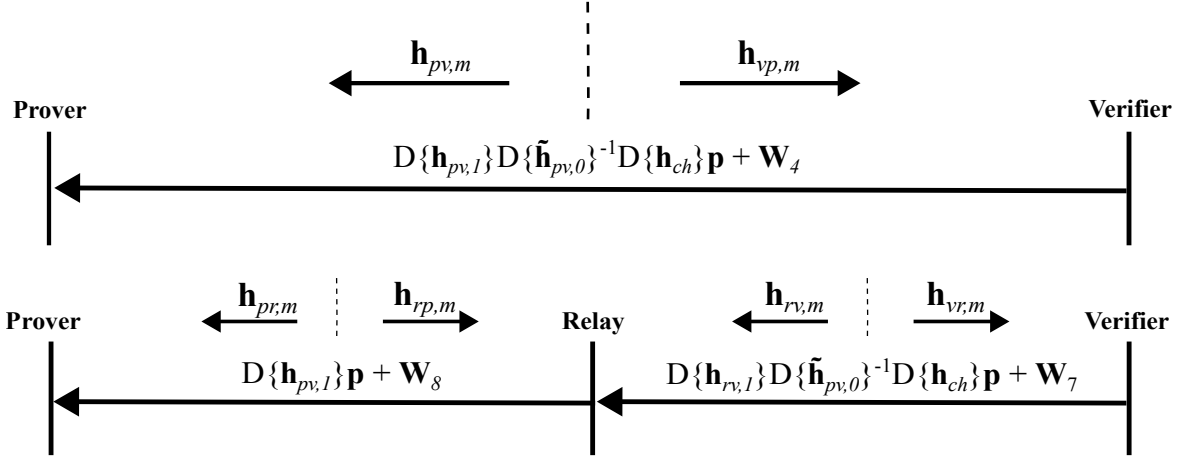


Figure 5.5: Step 4: with (bottom) and without (top) a decode-and-forward relay attack

which comes from the assumption that $D\{\mathbf{h}_{pv,1}\}D\{\tilde{\mathbf{h}}_{p,0}\}^{-1} \approx \mathbf{I}$. This means that the prover can estimate a channel that is ideally close to the challenge $\mathbf{h}_{ch}$ that the verifier has generated.

If a decode-and-forward relay attack is occurring, then the prover will not receive the challenge channel ($\mathbf{h}_{ch}$)

$$\tilde{\mathbf{h}}_{p,1} = \hat{\mathbf{h}}_{pr,1} \quad (5.9)$$

which will have a large MSE with $\mathbf{h}_{ch}$, since the channels are uncorrelated/independent. The relay will see the channel

$$\hat{\mathbf{h}}_{r,1} = D\{\hat{\mathbf{h}}_{rv,1}\}D\{\hat{\tilde{\mathbf{h}}}_{p,0}\}^{-1}\hat{\mathbf{h}}_{ch} \quad (5.10)$$

where $\mathbf{h}_{rv,1}$ and $\tilde{\mathbf{h}}_{p,0}$ are uncorrelated, which should obscure $\mathbf{h}_{ch}$ from the relay. If the attacker knew what $\mathbf{h}_{vr,0}$ was, it would be possible for them to estimate the proper filter to apply an active relay attack to the prover.

As mentioned in Section 5.1.3, if no encryption was utilized in step 3 the attacker could perform the following active relay attack:

1. Attacker knows $\tilde{\mathbf{h}}_{p,0}$ from (5.7) if there is no encryption.

2. Attacker multiplies $\tilde{\mathbf{h}}_{p,0}$ with (5.2), which they know:

$$\mathbf{D}\{\tilde{\mathbf{h}}_{p,0}\}\hat{\mathbf{h}}_{r,0} = \mathbf{D}\{\mathbf{h}_{K_2}\}^{-1}\mathbf{D}\{\hat{\mathbf{h}}_{pr,0}\}\mathbf{D}\{\hat{\mathbf{h}}_{rv,0}\}\hat{\mathbf{h}}_K \quad (5.11)$$

$$= (\mathbf{D}\{\mathbf{h}_{K_2}\}^{-1}\mathbf{D}\{\hat{\mathbf{h}}_{K_2}\})\mathbf{D}\{\hat{\mathbf{h}}_{pr,0}\}\hat{\mathbf{h}}_{rv,0} \quad (5.12)$$

$$\approx \mathbf{D}\{\hat{\mathbf{h}}_{pr,0}\}\hat{\mathbf{h}}_{rv,0} \quad (5.13)$$

3. Attacker filters retransmission with (5.10) and $\tilde{\mathbf{h}}_{p,0}$ and equalizes with (5.13)

$$\mathbf{h}_f = \mathbf{D}\{\hat{\mathbf{h}}_{r,1}\}\mathbf{D}\{\tilde{\mathbf{h}}_{p,0}\}\mathbf{D}\{\mathbf{D}\{\hat{\mathbf{h}}_{pr,0}\}\hat{\mathbf{h}}_{rv,0}\}^{-1} \quad (5.14)$$

$$= \mathbf{D}\{\hat{\mathbf{h}}_{pr,0}\}^{-1}\mathbf{D}\{\hat{\mathbf{h}}_{rv,0}\}^{-1}\mathbf{D}\{\hat{\mathbf{h}}_{rv,1}\}\mathbf{D}\{\tilde{\mathbf{h}}_{p,0}\} \cdot \mathbf{D}\{\hat{\mathbf{h}}_{p,0}\}^{-1}\hat{\mathbf{h}}_{ch} \quad (5.15)$$

$$\approx \mathbf{D}\{\hat{\mathbf{h}}_{pr,0}\}^{-1}\hat{\mathbf{h}}_{ch}, \quad (5.16)$$

where (5.15) to (5.16) is a result of commuting the diagonal matrices and assuming $\mathbf{D}\{\hat{\mathbf{h}}_{rv,0}\}^{-1}\mathbf{D}\{\hat{\mathbf{h}}_{rv,1}\} \approx \mathbf{I}$ and $\mathbf{D}\{\hat{\mathbf{h}}_{1,0}\}^{-1}\mathbf{D}\{\tilde{\mathbf{h}}_{p,0}\} \approx \mathbf{I}$.

This attack results in the prover's channel becoming $\mathbf{D}\{\mathbf{h}_{p,1}\}\mathbf{D}\{\hat{\mathbf{h}}_{pr,0}\}^{-1}\hat{\mathbf{h}}_{ch}$, which is the same as (5.8), the case where no relay attack is occurring.

5.1.5 Step 5: Prover Generation

In step 5, the prover repeats the processes in step 2 (Section 5.1.2), where they provide the encrypted and authenticated channel estimate to the verifier, so the final authentication can occur. As mentioned, if no decode-and-forward relay attack is occurring, the prover will estimate $\hat{\mathbf{h}}_{ch}$ (5.8). If a decode-and-forward relay attack is occurring they will estimate $\hat{\mathbf{h}}_{pr,1}$ (5.10). Encryption and authentication are applied as a precaution against potential active relay attacks, though no specific attack is known at the moment. The size of the transmission is the same as in step 2 (5.18).

5.1.6 Step 6: Decode-and-Forward Relay Detection with Authentication

In step 6, after decrypting and verifying the feedback, the verifier checks that the received feedback $\hat{\mathbf{h}}_{p,1}$ has a low MSE with the transmitted challenge ($\mathbf{h}_{ch}$). The MSE of $\hat{\mathbf{h}}_{p,1}$ and

$\mathbf{h}_{ch}$ is measured as

$$\text{MSE} = \frac{1}{N} \left(\hat{\mathbf{h}}_{p,1} - \mathbf{h}_{ch} \right)^H \left(\hat{\mathbf{h}}_{p,1} - \mathbf{h}_{ch} \right) \quad (5.17)$$

If $\text{MSE} \leq Th$, then the verifier concludes that the prover has (1) the secret key (2) no decode-and-forward relay is occurring. The threshold Th is selected heuristically, with higher threshold values increasing the false-positive rate, while lowering the false-negative rate, for a given SNR.

5.2 Feedback Size and Complexity

The size of the feedback transmitted in this protocol is given by the size of $\tilde{\mathbf{h}}_{p,0}$, where the number of subcarriers N and sample resolution (bits/sample) is M . The MAC is assumed to be a an HMAC using SHA3-256. The final transmission has a size of

$$\text{Transmission Size} = NM + 256 \quad (5.18)$$

The overall size of the transmission is relatively small. For example, if there are 128 subcarriers with a 12-bit DAC, then the transmission size is $128 \times 12 + 256 = 1792$ bits. This is significantly smaller than Protocol 1 and Protocol 3 (to be shown).

The complexity of this protocol is very low compared to the other protocols. The MSE (5.17) is a linear calculation $O(N)$. This is in contrast to the other protocols that require Fourier transforms and linear models.

5.2.1 Filter Generation and Security

The security against active decode-and-forward relay attacks is dependent on the obscurity of the channel from the verifier to the relay, provided by the filters $\mathbf{h}_{K_1}$ and $\mathbf{h}_{K_2}$ in Step 1 and Step 2 (5.1.1-5.1.2), respectively. The challenge filter $\mathbf{h}_{ch}$ in Step 4 (5.1.4) must also be randomly generated, such that a relay adversary has a negligible probability to bruteforce the filter and use it in an active relay attack. The result of the random filtering is the multiplication of the channel and the filter, which is referred to as “multiplicative encryption.” For example, in step 2 (as shown in Figure 5.4), with $\mathbf{h}_{K_2}$ and $\mathbf{h}_{rv,0}$, the resulting multiplication is

$$D\{\mathbf{h}_{K_2}\}\mathbf{h}_{rv,0} = (h_{K_2}[0]h_{rv,0}[0], h_{K_2}[1]h_{rv,0}[1], \dots, h_{K_2}[N-1]h_{rv,0}[N-1]).$$

If the relay does not know what $\mathbf{h}_{K_2}$ and $\mathbf{h}_{rv,0}$ are, then the multiplicative encryption assumes that it cannot determine these sub-filters from only observing their multiplication $D\{\mathbf{h}_{K_2}\}\mathbf{h}_{rv,0}$. This assumption is used to protect the discovery of $\mathbf{h}_{rv,0}$, $\mathbf{h}_{rp,0}$, $\mathbf{h}_{rp,1}$, and $\mathbf{h}_{ch}$, which should prevent active relay attacks from occurring, as the relay would not be able to come up with an appropriate $\mathbf{h}_f$.

The generation of both the keyed filter or challenge filter follows a general process of generating $M < N$ random tuples (a_l, b_l) , where the elements $a_l, b_l \in \{-2^{n-1}, -2^{n-1} + 1, \dots, -1, 0, 1, \dots, 2^{n-1} - 1\}$ are the in-phase and quadrature components, respectively, that are also each uniformly generated. The exponent n is the resolution of the analog-to-digital and digital-to-analog converters (ADCs and DACs, respectively) of the transmitting and receiver devices (we assume they are the same), where a_l and b_l are generated as $2s$ -complement signed integers. The result of this generation is M points that are used as seeds for interpolating the random filters. The number of the random bits is $2 \times n \times M$, which is the amount of bits in the interpolated sequence. In all the interpolations in this thesis, pulse shaping using a nyquist pulse (i.e., sinc) is used, but there are many different interpolation methods that could be used.

5.2.2 Simulation Experimentation

This section looks at the Monte-Carlo results from the simulations. The performance metrics that are observed are the false-positive rate and the false-negative rate of the ability to detect the decode-and-forward relay attack. The false-positive rate (FPR) is defined as the number of decode-and-forward attacks indicated when none have occurred, out of a total set of none relayed authentications, as defined in (4.33). Similarly, the false-negative rate (FNR) is the number of attacks missed, out of a total set of relayed authentications, as defined in (4.34).

For both the FPR and FNR a lower value is desired for the protocol to show that it can discern relay attacks from expected authentication, as this is their objective. The simulation was performed in the Julia programming language running on a computer using an i9-10900F processor and 64 GB of RAM on Ubuntu 22.04. The simulation uses baseband sampling with perfect timing synchronization and no transceiver non-idealities. Unlike Protocol 1, this protocol and the simulation will consider Rayleigh and Rician channels. Since only a single exchange is needed, the temporal correlation also does not matter. This software implementation also considers all the processes in the SDR detailed in chapter 9, except for the waterfiling algorithm in Section 9.7, where power allocation is automatically shared between the prover, verifier, and relay for simplification.

The parameters for the simulation are shown in Table 5.2, the pilot-noise SNR at the verifier plays a role in the signal, and is the main independent variable in the simulations. Other independent variables include the delay spread and Rician factor. The threshold for the MSE (Th) was selected heuristically from observing simulation experiments. Each simulation at every SNR was repeated 1000 times to get the pass and fail counts.

The results for the FPR in a NLOS environment can be seen in Figure 5.6. It can be seen, up to 5 dB SNR that the performance is relatively flat. As 5 dB SNR is reached, the FPR starts to rise. It is likely this would continue to rise as it approaches 0 dB, but this was not evaluated as the coarse synchronization process cannot detect signals that are that weak.

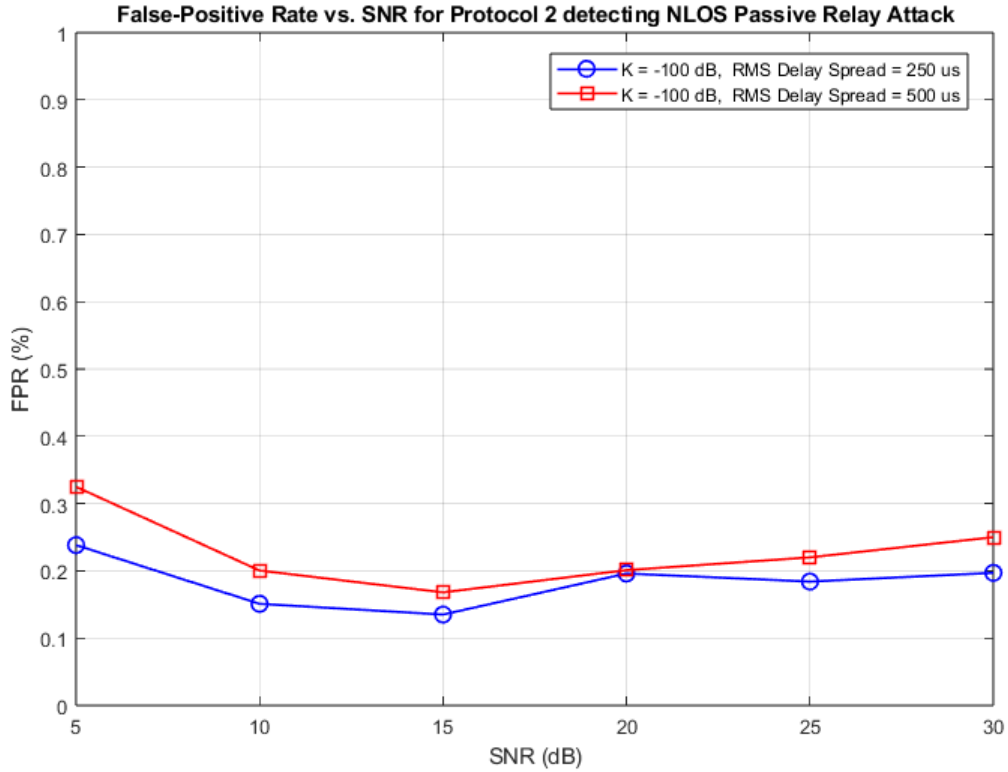


Figure 5.6: FPR versus SNR simulation results in a NLOS channel with small and significant frequency selectivity

The results for the FPR in a LOS environment can be seen in Figure 5.7. Compared to the NLOS environment, the LOS term seems to help significantly with performance.

While there is a similar upswing as 0 dB is approached, the overall performance is much better.

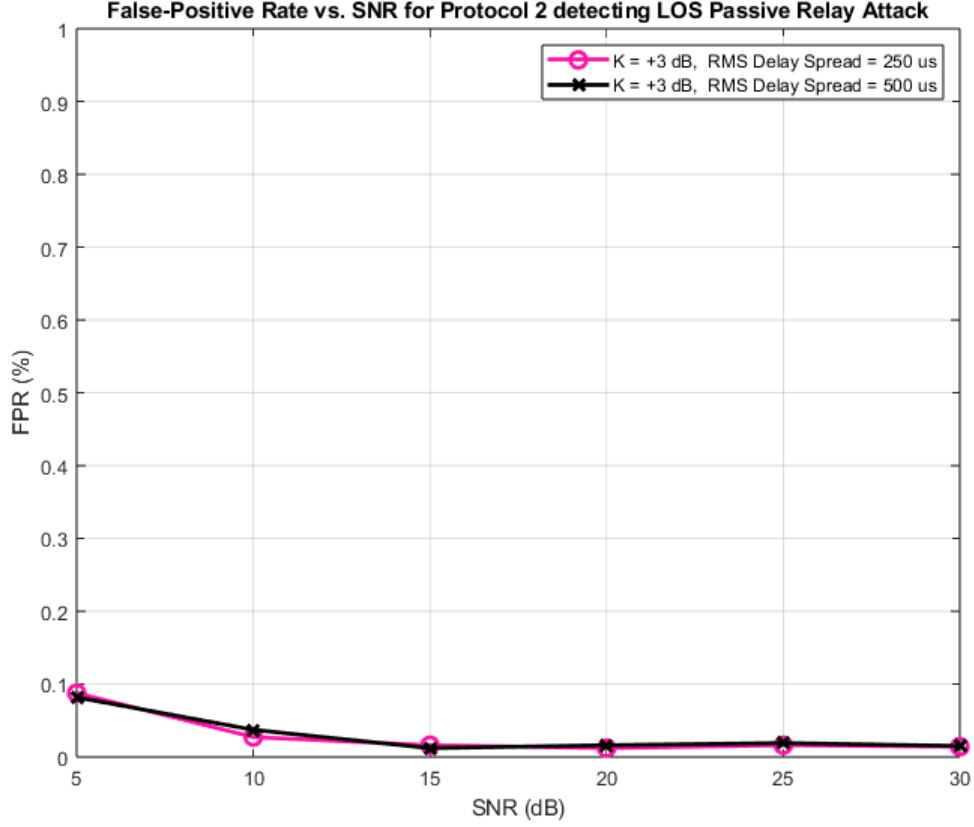


Figure 5.7: FPR versus SNR simulation results SNR in a LOS channel with small and significant frequency selectivity

The results for the FNR in a NLOS environment can be seen in Figure 5.8 and Figure 5.9, respectively. In general, these results are perfect. As noted in the Protocol 1 simulation results (Section 4.3), detecting a relay attack is generally easy, especially at low SNR, as the channel conditions and it becomes uncorrelated and authentication fails. On the other hand, having a protocol that can simultaneously detect relay attacks, while having a low FPR is difficult. In general, the results for this protocol are good, except for in an NLOS environment, which is generally an extreme channel condition.

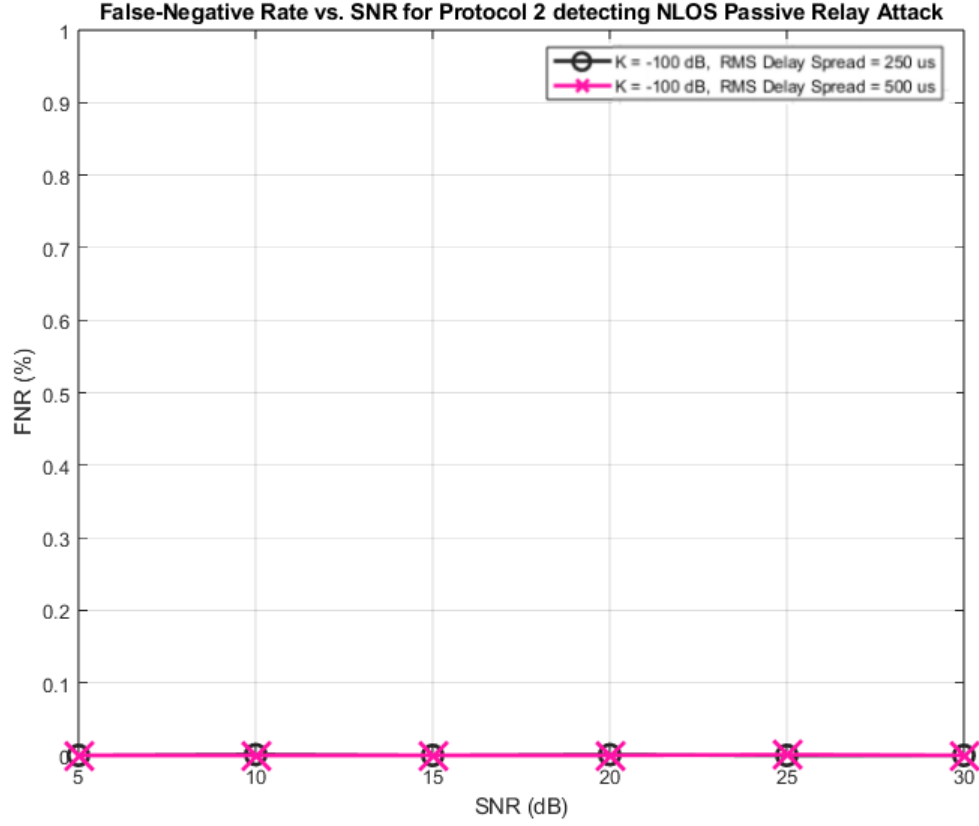


Figure 5.8: FNR versus SNR simulation results in a NLOS channel with small and significant frequency selectivity

Table 5.2: DF Simulation parameters for Protocol 2 with OFDM-QPSK transmissions

	Parameter	Value
1	Symbol Duration (μs)	1
2	Carrier Frequency (MHz)	900
3	RMS Delay Spread (ns)	-
4	Rician Factor (dB)	-
5	SNR (dB)	-
6	Subcarriers (#)	64
7	BW (MHz)	10
8	Power-Delay Profile	Exponential
9	Th	0.016

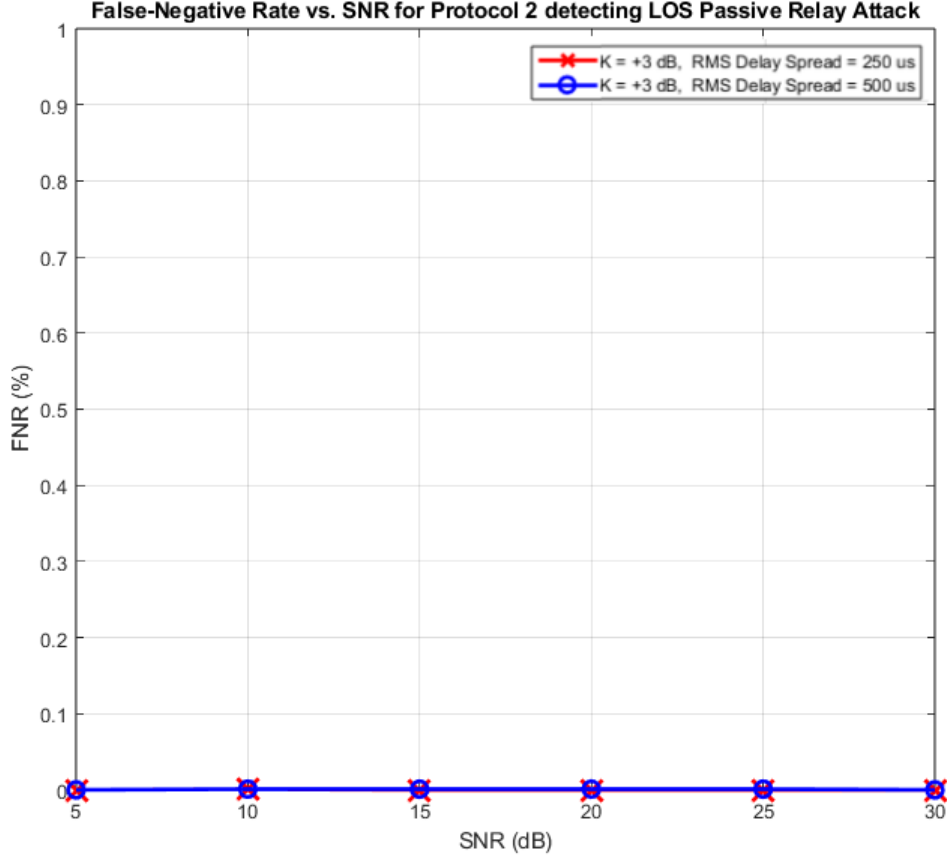


Figure 5.9: FNR versus SNR simulation results in a LOS channel with small and significant frequency selectivity

5.2.3 Hardware Experimentation

In this section hardware experiments are performed on a Libre SDR as the verifier and prover and a Pluto SDR as the relay, to evaluate the practical performance of the proposed protocol for detecting passive decode-and-forward relay attacks. The performance criterion under evaluation are the FPR (4.33) and FNR (4.34). To control the SDRs implementing the verifier and relay remotely, these devices are each connected to a Raspberry Pi 5,

remote desktop, and a battery pack. To control the signal-to-noise ratio (SNR), the signal and noise power are measured to get an estimate of the SNR using the sample variance, absent any interference with a 23 dB PA, and the SDR transmission gain is scaled back to achieve the desired SNR. Specific details of the hardware implementation can be found in Chapter 9. Each SNR experiment is repeated 30-50 times to collect the FNR and FPR. The transmission frequency used is 915 MHz, which has a wavelength of 0.33 m. The devices are evaluated in an indoor office environment, separated by 5 m (to ensure far field channel conditions) and with direct LOS. The communication system implemented is 64 subcarriers with QPSK-OFDM transmission utilizing a 10 MHz bandwidth with a 20 MSPS sampling rate, with a 25 % cyclic-prefix. The pilot $\mathbf{p} = \mathbf{F}\{\text{FZC}(8)\}$ is the DFT of a Frank-Zadoff-Chu (FZC) sequence. The parameters can be seen in Table 5.3

Table 5.3: Hardware experiments parameters for Protocol 2 with OFDM transmissions

	Parameter	Value
1	Symbol Duration (μs)	40
2	Carrier Frequency (MHz)	915
3	Distance (m)	5
4	SNR (dB)	-
5	Subcarriers (#)	64
6	BW (MHz)	10
7	f_s (MSPS)	20
8	Modulation	QPSK
9	Cyclic Prefix (%)	25
10	Th	0.016

Channel estimation utilizes the least square estimator (LSE) and symbol estimation using maximum-likelihood estimation (MLE). When performing relaying, to verify that the relay is working during the protocol, a challenge-response authentication with a nonce and MAC is simultaneously relayed along with the protocol. If the challenge-response passes during a relay attack, then the relay was working correctly and was not just retransmitting garbage. This acts as a “sanity check” to verify that the relay detection can actual discern when an attack is occurring.

For the protocol only a passive decode-and-forward attack is utilized. Given this, for simplicity the keyed filters are removed in this hardware implementation of the protocol.

Since no amplify-and-forward attack is performed, they are not necessary and add burden to the hardware implementation. The simulation in Section 5.2.2 is performed with the keyed filters, to ensure they do not cause performance issues (which they do not). Similarly, the encryption is not considered for simplicity, as it just adds unnecessary processing of the binary information, which also adds burden to the hardware implementation.

The result of the FPR implemented on hardware can be seen in Figure 5.10. The result of the FNR implemented on hardware can be seen in Figure 5.11. In general, the results are quite close to what is seen in the simulations, which lends to the confidence in the protocol being able to detect passive decode-and-forward relay attacks. The FPR is relatively flat at $< 10\%$, until the SNR reaches 10 dB. The reason for this is that as the SNR decreases, noise adds more error in the MSE in (5.17), as more AWGN appears in the channel estimate the prover gets of $\hat{\mathbf{h}}_p$. On the other hand, the FNR has a similar result as Protocol 1 (chapter 4) and the simulation results (Figure 5.9), with a very low rate for every SNR. As the SNR decreases, the noise causes the MSE between the different channels to further increase, which will only improve the performance.

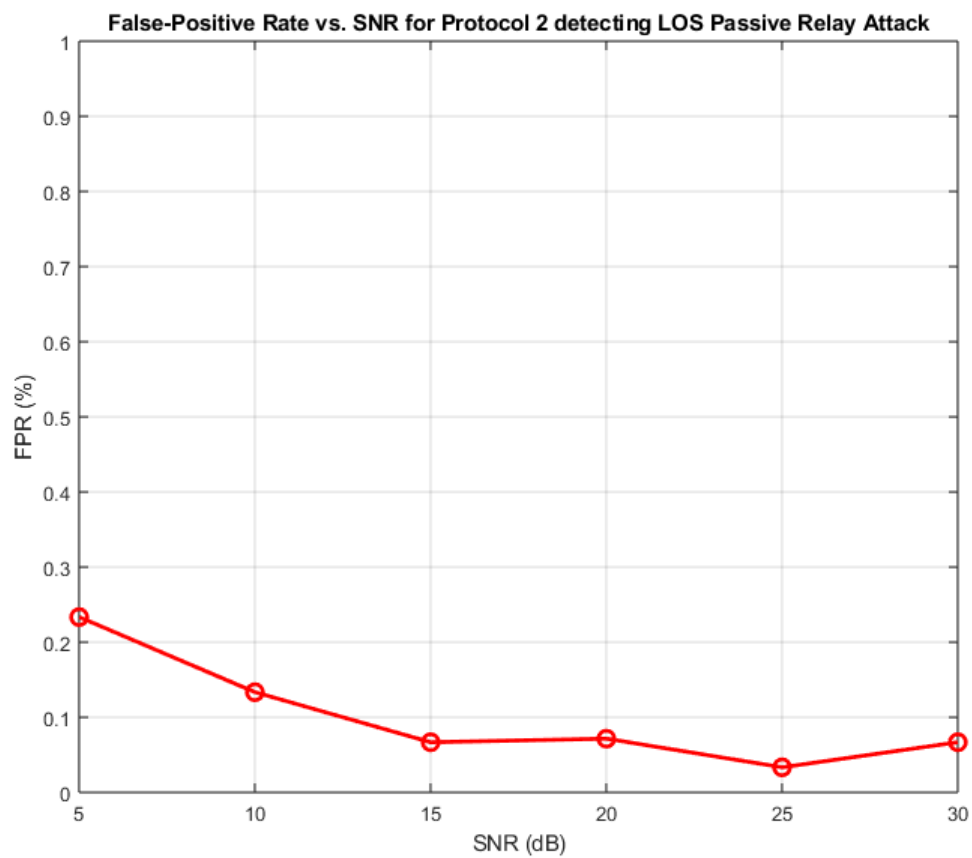


Figure 5.10: FPR versus SNR hardware results in a LOS channel with small and significant frequency selectivity [128]

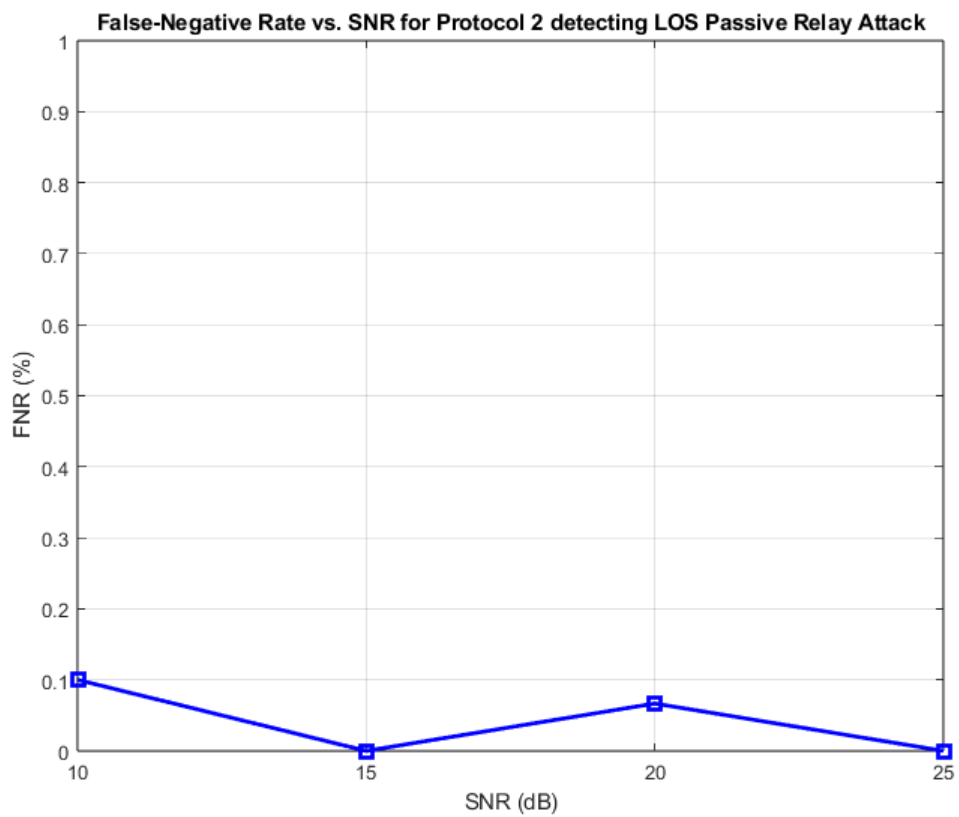


Figure 5.11: FNR versus SNR hardware results in a LOS channel with small and significant frequency selectivity [128]

Chapter 6

One-Way Amplify-and-Forward Relay Detection Protocol under Passive Attacks using Channel Distribution ¹

In this chapter a novel one-way symmetric-key challenge-response protocol that can detect amplify-and-forward relay attacks is described in detail and evaluated using simulations. A protocol that can only detect amplify-and-forward attacks is useful for using in conjunction with a separate decode-and-forward relay attack detection protocol to expand the types of attacks that can be detected by a system. Similarly to Protocol 1 (Chapter 4), this protocol does not handle active relay attackers that attempt to manipulate transmissions, instead of only relaying them. While this can limit the effectiveness of this protocol against capable attackers that have a deep knowledge of the target challenge-response protocol and its underlying communication system, most practical attackers are not going to have this knowledge or access to hardware that allows that level of manipulation. For example, a car thief is not likely to have an SDR with such capabilities and the know how to program it themselves. On the other hand, a passive amplify-and-forward relay attack is one of the simplest ways to perform a relay attack, as it only requires some antennas, mixers, filters, and amplifiers.

The basis of this protocol is to utilize the expected Rayleigh distribution to distinguish

¹The main work in this chapter comes from an unpublished paper currently available on IEEE TechRxiv R. Abubaker and G. Gong [128].

if a relay attack is occurring or not. As an amplify-and-forward channel preserves channel characteristics between the relayed channel, the distribution of a single channel varies from a channel with two or more channels compounded together. To determine if a relay attack is occurring, the verifier uses a normality test.

6.1 Protocol Details

For this protocol, the following terminology is defined for use in the protocol:

- CK is an 128-bit encryption key generated from a symmetric-key K using a KDF and is only known by the prover and verifier,
- IK is an 128-bit authentication key generated from a symmetric-key K using a KDF only known by the prover and verifier,
- $\text{Enc}_{CK}\{\cdot\}$, $\mathbb{C}^N \rightarrow \mathbb{C}^N$, is a secure symmetric-key encryption that takes an input symbol and generates an encrypted output symbol using CK ,
- $\text{MAC}_{IK}\{\cdot\}$, $\mathbb{C}^N \rightarrow \mathbb{C}^N$, is a secure message authentication code that takes an input symbol and generates a tag output symbol using IK ,
- $\mathbf{p}_i \in \mathbb{C}^N$ is pilot sequence used by device $i = \{0, 1\}$, which contains the frequency sample and is publicly known,
- $\mathbf{h}_{i,m} \in \mathbb{C}^N$ is an FIR filter representing the channel seen by device $i = \{0, 1\}$ at the m^{th} time instance,
- $\mathbf{W}_m$ is a matrix of AWGN samples $\sim \mathcal{CN}(0, N_0)$.

Similar to Protocol 1 (Chapter 4), assume that both communication entities share a cryptographic key SK and use KDF to generate IK for symmetric-key based message authentication and CK for encryption. This protocol consists of 5 steps (including the initiation), which can be seen in Table 6.1. The details of these phases are given in the subsequent subsections.

Table 6.1: Phases and their Descriptions in Protocol 3

step	Description
1	The prover initiates with a request $[\mathbf{p}_1, \mathbf{x}_{\text{req}}]$ to the verifier.
2	The verifier initiates a pilot exchange by transmitting a pilot $\mathbf{p}_2$ to the prover.
3	The prover encrypts and authenticates its pilots and received symbols to share with the verifier.
4	The verifier estimates both their channel and the prover's channel using the set of pilots and received symbols.
5	The verifier evaluates the distribution of its channel to evaluate if an amplify-and-forward attack has occurred.

The base of this protocol is similar to Protocol 1 (Chapter 6). The first three steps are similar to Protocol 1, with some difference detailed in the subsequent subsections. The last step is entirely different than Protocol 1 since this protocol detects amplify-and-forward attacks, versus decode-and-forward. The divergence in the last step allows both protocols to be evaluated by the verifier simultaneously, which can provide decode-and-forward and amplify-and-forward detection with a single pilot exchange. A visual of Protocol 3 and where it diverges from Protocol 1 can be seen in Figure 6.1.

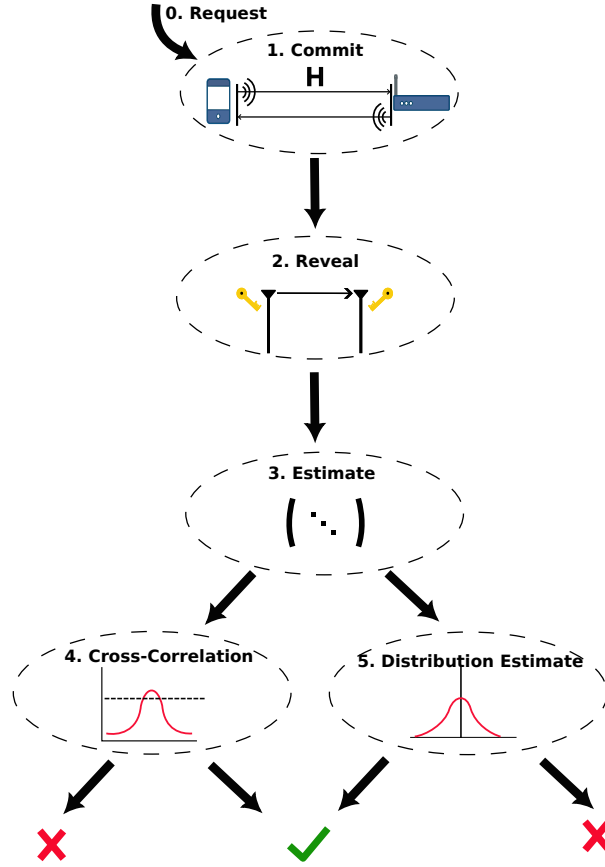


Figure 6.1: General One-way Challenge-Response Authentication Protocol [128]

6.1.1 Step 1: Initiation

The prover initiates the protocol by transmitting a request symbol $\mathbf{x}_{\text{req}} \in \mathbb{C}^N$ with a pilot in the preamble $\mathbf{p} \in \mathbb{C}^N$. The request is known to the verifier. The pilot is also known between all parties, and is used to perform equalization and other signal processes needed to demodulate.

6.1.2 Step 2: Channel Probing through a Pilot Exchange

Identically to Protocol 1, after a request is received for an authentication, a pilot exchange begins, which is used to perform a probing of the wireless channel. The pilots are trans-

mitted as vectors $\mathbf{p}_i = (p_i[0], p_i[1], \dots, p_i[N-1])$ and the channel is represented with an FIR filter $\mathbf{h}_{i,m} = (h_{i,m}[0], h_{i,m}[1], \dots, h_{i,m}[N-1])$, where $\mathbf{h}_{i,m} \in \mathbb{C}^N$ and is an approximate of the physical channel that is cut-off at the signal length in time (NT_s) at the m^{th} time instance. Note that both the symbol and pilot are in the frequency domain. Given this, and the cutt-off, the received frequency domain samples are given by,

$$\mathbf{r}_{i,m} = \mathbf{D}\{\mathbf{p}_i\}\mathbf{h}_{i,m} + \mathbf{w}_m,$$

where $\mathbf{w}_m \in \mathbb{C}^N$ has the distribution $\sim \mathcal{N}(\mathbf{0}^N, N_0\mathbf{I})$

In this pilot exchange, the verifier and prover both select their own pilot symbol prior to the communications, which are fixed for each time instance m and publicly known. The pilots from the prover and verifier are transmitted M times in a half-duplex exchange. For example, at time instance m , the pilot $\mathbf{p}_i$ is sent from the i^{th} device, through the channel $\mathbf{h}_{i,m}$, and to the receiving device. After the probes from both parties have been transmitted, the verifier and prover will have the following set of information that is fixed for the rest of the protocol,

	Prover ($i = 1$)	Verifier ($i = 2$)
Received Symbols	$\mathbf{R}_1 = \mathbf{D}\{\mathbf{p}_2\}\mathbf{H}_1 + \mathbf{W}_1$	$\mathbf{R}_2 = \mathbf{D}\{\mathbf{p}_1\}\mathbf{H}_2 + \mathbf{W}_2$
Known Pilots	$\mathbf{p}_1, \mathbf{p}_2$	$\mathbf{p}_1, \mathbf{p}_2$

where $\mathbf{R}_i, \mathbf{H}_i, \mathbf{W}_i \in \mathbb{C}^{N \times M}$. The matrix channel matrix can be express as $\mathbf{H}_i = (\mathbf{h}_{i,0}, \mathbf{h}_{i,1}, \dots, \mathbf{h}_{i,M-1})$ and $\mathbf{W}_i = (\mathbf{w}_{i,0}, \mathbf{w}_{i,1}, \dots, \mathbf{w}_{i,M-1})$.

6.1.3 Step 3: Channel Feedback

After the pilot exchange occurs, the random pilots generated by the prover and received symbols that the prover received are encrypted and authenticated using an authenticated encryption with the symmetric-key to prevent any modification or replay from an attacker. The encrypted and authenticated received symbols and pilot symbols are only sent from the prover to the verifier. The data to be transmitted by the prover to the verifier is

$$\mathbf{M}_1 = \text{Enc}_{CK,1}(\mathbf{R}_1, \mathbf{p}_1) || \text{Tag}_{IK,1}(\mathbf{R}_1, \mathbf{p}_1) \quad (6.1)$$

The scheme utilized to generate and exchange information may reveal the channel state information, but an attacker cannot take advantage of this information as the prover and verifier have passed the commit phase, and the reveal by the prover is authenticated. After the transmission the information that the prover and verifier has is

	Prover ($i = 1$)	Verifier ($i = 2$)
Received Symbols	$\mathbf{R}_1$	$\mathbf{R}'_1, \mathbf{R}_2$
Known Pilots	$\mathbf{p}_1$	$\mathbf{p}'_1, \mathbf{p}_2$

where $\mathbf{R}'_1$ and $\mathbf{p}'_1$ indicate that there may exist some error between the revealed information from the prover, which could occur during the transmission. Compared to Protocol 1, this protocol will require significantly more pilot exchanges, which is will be discussed in Section 6.2.

Similarly to Chapter 4, the verifier can estimate the channel matrices can also be estimated

$$\begin{aligned}\hat{\mathbf{H}}_1 &= \mathbf{D}\{\mathbf{p}_2\}^{-1}\mathbf{R}'_1 \\ &= \mathbf{H}_1 + \mathbf{D}\{\mathbf{p}_2\}^{-1}\mathbf{W}_1,\end{aligned}\tag{6.2}$$

and

$$\begin{aligned}\hat{\mathbf{H}}_2 &= \mathbf{D}\{\mathbf{p}_1\}^{-1}\mathbf{R}_2 \\ &= \mathbf{H}_2 + \mathbf{D}\{\mathbf{p}_1\}^{-1}\mathbf{W}_2.\end{aligned}\tag{6.3}$$

6.1.4 Step 4: Amplify-and-Forward Relay Detection

When an amplify-and-forward relay attack occurs, the signal is re-transmitted through K analog relay nodes. When utilizing OFDM, if a cyclic-prefix is used and the CIR is short enough such that no ISI occurs, then the CFR of the end-to-end transmission can be approximated as the product of the CFR of each intermediary channel, from the prover to the verifier. Due to the reciprocity of the CFR in each direction and the commutative property of convolution, the same multiplication of the CFR happens in both directions and differs due to Doppler spread and the AWGN for the legitimate parties. Considering

a unity amplification with no AWGN for simplicity, a single forward direction can be expressed as

$$\mathbf{h}_m = \left(\prod_{k=0}^K h_{k,m}[0], \dots, \prod_{k=0}^K h_{k,m}[N-1] \right), \quad (6.4)$$

where each element $h_{k,m}[n]$ is the CFR at the n^{th} subcarrier at the m^{th} transmission at the k^{th} amplify-and-forward relay. A visualization of the effect of multiple relays can be seen in Figure 6.2.

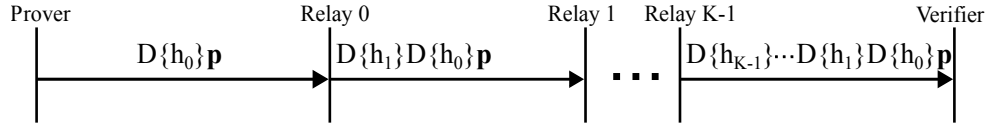


Figure 6.2: K amplify-and-forward relay nodes with K uncorrelated channels, resulting in a final channel that is the multiplication of each CFR

The channel CFR matrix after M exchanges is $\mathbf{H} \in \mathbb{C}^{N \times M}$ and can be expressed as

$$\mathbf{H} = \begin{pmatrix} \prod_{k=0}^K h_{k,0}[0] & \prod_{k=0}^K h_{k,1}[0] & \cdots & \prod_{k=0}^K h_{k,M-1}[0] \\ \prod_{k=0}^K h_{k,0}[1] & \prod_{k=0}^K h_{k,1}[1] & \cdots & \prod_{k=0}^K h_{k,M-1}[1] \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{k=0}^K h_{k,0}[N-1] & \prod_{k=0}^K h_{k,1}[N-1] & \cdots & \prod_{k=0}^K h_{k,M-1}[N-1] \end{pmatrix}.$$

The columns of the matrix when $K = 0$ is a Gaussian random process with $\mathbf{h}_{k,m} \sim \mathcal{CN}(\mathbf{0}^N, \mathbf{\Sigma}_f)$, where the 0 comes from the assumption that the channel is Rayleigh. If $K > 0$, which indicates an amplify-and-forward relay attack, then the resulting channel elements can be expressed as the product of Gaussian random variables (RVs), which has a distribution given by the product of K independent Gaussian random variables, following the assumptions in Section 2.2. In general, it is difficult to express the distribution for $K \geq 2$, but the base case of $K = 1$ (i.e., a single amplify-and-forward relay) is a well explored problem [141, 142]. The distribution is given by $h_{0,m}[n] \sim \mathcal{CN}(0, \rho_0)$ and $h_{1,m}[n] \sim \mathcal{CN}(0, \rho_1)$. The distribution of the product of 2 Gaussians has been derived in closed form by *Cui et. al* [142]. The losses $0 < \rho_0, \rho_1 \leq 1$ are the free-space path-loss coefficients in each channel, respectively (2.3). This expression for the single terms is given by

$$\begin{aligned}
f_{\Re\{h_{0,m}[n]\}\Re\{h_{1,m}[n]\}}(h) &= \frac{1}{\pi\sqrt{\rho_0}\sqrt{\rho_1}} J'_0\left(\frac{|h|}{\sqrt{\rho_0}\sqrt{\rho_1}}\right), \\
&= f_{\Im\{h_{0,m}[n]\}\Im\{h_{1,m}[n]\}}(h)
\end{aligned} \tag{6.5}$$

where the inphase components are only considered for simplicity for the rest of the chapter. J'_0 is the modified Bessel function of the 2nd kind and 0th order. The results shows how the cascade of 2 independent channels no longer has a Gaussian distribution, if ρ_0 and ρ_1 have a similar magnitude. These variances are related to the path-loss in the 2 separate channels and the power delay profiles (i.e., long-term fading effects). This result extends to successive amplify-and-forward relays. As long as at least 2 channels in the cascade have similar long-term fades, then the overall channel will not have a Gaussian distribution. These observations act as the indicator for the existence of an analog relay attack. If the channel is Gaussian distributed, then there is no amplify-and-forward relay attack, otherwise there is.

When considering the inphase and quadrature terms together, the distribution of both becomes increasingly complicated. For example, the inphase component of the received estimated channel where $K = 1$ (1 amplify-and-forward relay) the estimate of the n^{th} and m^{th} element of the verifier's estimated channel $\hat{\mathbf{H}}$ can be expressed as

$$\begin{aligned}
\Re\{\hat{h}_m[n]\} &= \Re\{h_{1,m}[n]\}(G_0\Re\{h_{0,m}[n]\} + G_0\Re\{\frac{w_{0,m}[n]}{p[n]}\}) \\
&\quad - \Im\{\hat{h}_m[n]\}(G_0\Im\{h_{0,m}[n]\} + G_0\Im\{\frac{w_{0,m}[n]}{p[n]}\}) + \Re\{\frac{w_{1,m}[n]}{p[n]}\},
\end{aligned} \tag{6.6}$$

$$\begin{aligned}
&= G_0\Re\{h_{1,m}[n]\}\Re\{h_{0,m}[n]\} + G_0\Re\{h_{1,m}[n]\}\Re\{\frac{w_{0,m}[n]}{p[n]}\} \\
&\quad - G_0\Im\{\hat{h}_m[n]\}\Im\{h_{0,m}[n]\} + G_0\Im\{\hat{h}_m[n]\}\Im\{\frac{w_{0,m}[n]}{p[n]}\} + \Re\{\frac{w_{1,m}[n]}{p[n]}\},
\end{aligned} \tag{6.7}$$

where G_0 is the gain from the amplify-and-forward relay and $w_{k,m}[n] \sim \mathcal{CN}(0, N_0/2)$ is the noise sample at the k^{th} node. The power in each of the terms in (6.7) can be given as $G_0^2\rho_0L_2$, $G_0^2\rho_1/\gamma_0$, and $1/\gamma_1$, where $\gamma_k = p^2[n]/N_{0k}$ is the CNR with at the k^{th} node. As K increases the terms expand with more path-loss coefficients, but there is always the last noise term with power $1/\gamma_K$. Assuming independence between the channel samples and noise samples, the estimated channel distribution (6.7) becomes an increasingly complicated expression of (6.5) convolved with a Gaussian distribution $\sim \mathcal{N}(0, 1/\gamma_K)$. This

expression is given as

$$f_{\mathbb{R}\{\hat{h}_m[n]\}}(h) = \frac{1}{\pi G_0 \sqrt{\rho_0 \rho_1}} J'_0 \left(\frac{|h|}{G_0 \sqrt{\rho_0 \rho_1}} \right) * \frac{1}{\pi G_0 \sqrt{\rho_1 / \gamma_k}} J'_0 \left(\frac{|h|}{G_0 \sqrt{\rho_1 \gamma_k}} \right) * \frac{1}{\sqrt{2\pi N_0}} e^{-\frac{p^2[n]h^2}{N_0}}, \quad (6.8)$$

which is quite complex. If $K = 0$ (there is only a single amplify-and-forward relay), the distribution has a Gaussian distribution. As mentioned, when $\rho_i > \rho_j$ the channel can appear Gaussian. The other case is when the third term in (6.8) dominates the distribution. This can happen if the product of the losses are large enough and the gain is relatively small, as the power of the last noise term will dominate and this channel will look like a set of IID AWGN samples. This is unique to amplify-and-forward relays, as the losses accumulate as a product as they are relayed, which can weaken the overall channel relative to the noise, if the gains are not large enough. This scenario counter-intuitive to the goals of the amplify-and-forward attacker, as it essentially means the signal is getting weaker, relative to noise. This could prevent the relaying from working to begin with.

To perform the amplify-and-forward relay attack detection, the cumulative distribution function (CDF) of the channel estimates and the autocorrelation in time and frequency are used to determine if an analog relay exists between the prover and verifier through a hypothesis (normality) test. To perform the test, the verifier uses the channel estimates $\hat{\mathbf{H}}_i$ from (6.2) and (6.3), and compares that against an expected normal distribution if the channel-to-noise ratio (CNR) is high, or looks at the coherence estimates if the CNR is low (noise dominates). Since the decode-and-forward test must pass before the amplify-and-forward test occurs (per the order of the protocol), the verifier can assume that $\hat{\mathbf{H}}_1 \approx \hat{\mathbf{H}}_2 + \mathbf{W}$, and average them to slightly reduce the effects of noise

$$\hat{\mathbf{H}} \triangleq \frac{\hat{\mathbf{H}}_1 + \hat{\mathbf{H}}_2}{2} \quad (6.9)$$

To first determine if an amplify-and-forward relay attack is occurring, the estimate of the coherence time $\hat{\tau}_c$ and the coherence bandwidth $\hat{b}_c$ are observed. If both of these values are below a threshold, the channel is changing quickly due to a relay channel dominated by AWGN, which is assumed to occur due to the relays, as shown in (6.4). This means the channel is under an amplify-and-forward relay attack and fails the test.

To first filter the results for the third scenario (strong noise), Equation 6.1.4 is used, which estimates the coherence in time and frequency, and is used on $\hat{H}_i$. This is done by simply using the first delay in the autocorrelation $C_{\hat{\mathbf{h}}_n}[l]$ (rows) for time and $C_{\hat{\mathbf{h}}_m}[j]$

Algorithm 1 Coherence Estimation

```

1:  $C_{\hat{\mathbf{h}}_n}[l] \leftarrow 1/N \sum_{n=0}^{N-1} F\{|\hat{\mathbf{h}}_n|^2\}$ 
2:  $C_{\hat{\mathbf{h}}_m}[l] \leftarrow 1/M \sum_{m=0}^{M-1} F\{|\hat{\mathbf{h}}_m|^2\}$ 
3:
4:  $\hat{B}_c \leftarrow 0$ 
5: while  $C_{\hat{\mathbf{h}}_n}[\hat{B}_c] < \alpha/N \sum_{l=0}^{N-1} C_{\hat{\mathbf{h}}_n}[l]$  do
6:    $\hat{B}_c \leftarrow \hat{B}_c + 1$ 
7: end while
8:
9:  $\hat{\tau}_c \leftarrow 0$ 
10: while  $C_{\hat{\mathbf{h}}_m}[\hat{\tau}_c] < \alpha/M \sum_{j=0}^{M-1} C_{\hat{\mathbf{h}}_m}[j]$  do
11:    $\hat{\tau}_c \leftarrow \hat{\tau}_c + 1$ 
12: end while
13:
14: return  $\hat{b}_c$ 
15: return  $\hat{\tau}_c$ 

```

(columns) for frequency, that is below their respective means. If these estimates are small, then it is assumed that the channel is extremely noisy and this is occurring due to a relay.

If AWGN does not dominate the channel, the normality check is used to check the channel distribution. A decimation (i.e., subsampling) is performed to decorrelate and normalize the channel samples into an uncorrelated random process. The coherence bandwidth and time are converted into decimation indices

$$N_d = \hat{B}_c \tag{6.10}$$

$$M_d = \hat{\tau}_c \tag{6.11}$$

The result of this process is a mapping of the channel frequency-time samples with unknown correlation, to a decimated matrix of uncorrelated channel frequency samples and reduced time correlation, $\mathbb{C}^{N \times M} \rightarrow \mathbb{C}^{\lfloor N/N_d \rfloor \times \lfloor M/M_d \rfloor}$, $\hat{\mathbf{H}} \mapsto \hat{\mathbf{H}}_d$. This operation can be visualized in Figure 6.3.

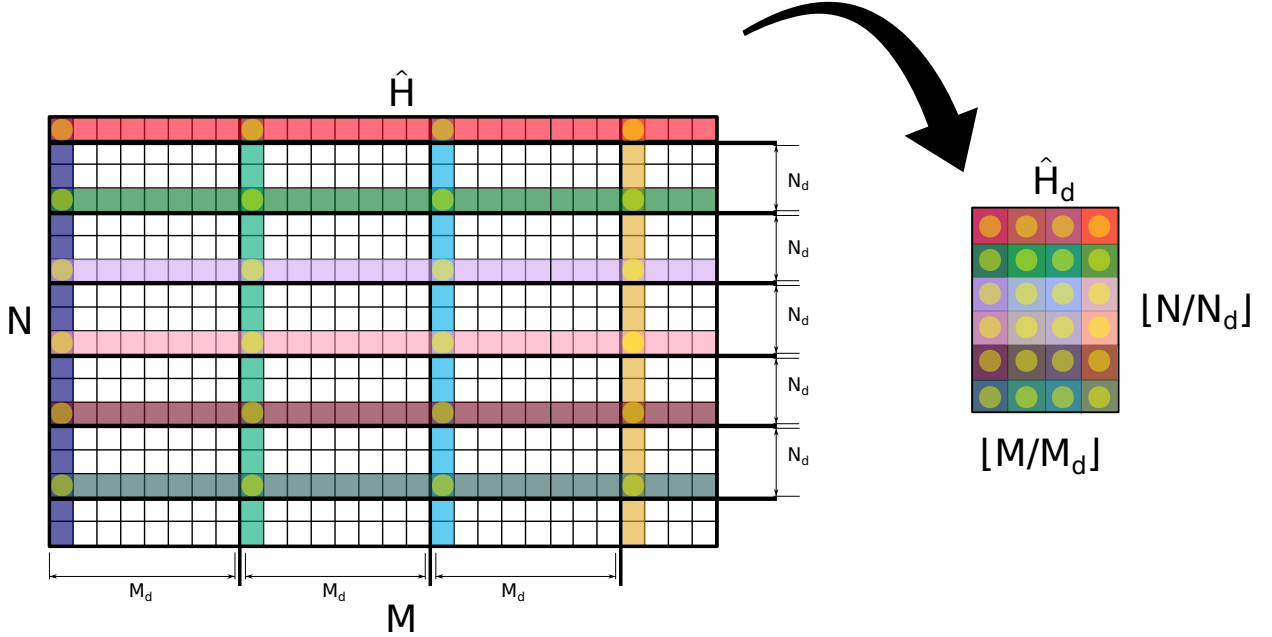


Figure 6.3: The mapping of correlated time-frequency channel to an uncorrelated channel through decimation [128]

After the channel has been decimated, the verifier normalizes the channel using the sample variance, yielding the channel matrix $\hat{\mathbf{H}}_{norm}$.

Following the established assumptions, if the communications are not under an amplify-and-forward relay attack, then $\hat{\mathbf{H}}_{norm} \stackrel{\text{IID}}{\sim} \mathcal{N}(0, 1)$. The number of values depends on the size of the normalized matrix. This will depend on the coherence time, coherence bandwidth, subcarriers, the OFDM symbol rate, and the number of exchanges.

To check if the result passes a normality check, the Shapiro-Wilk (S-W) test is selected utilizing Royston's approximations [143]. The S-W test is normality test that is used in this step to determine if a set of random experiments (uncorrelated channel samples) comes from a normal distribution or not. At a high-level, the S-W test observes the ordered statistic of the experiments to determine the error between the binned distribution and the normal distribution. Since each sample in $\hat{\mathbf{H}}_{norm}$ is ideally independent after the decimation, they can be vectorized as IID samples for the S-W test, where the length of the vector is given by $\lfloor N/N_d \rfloor \lfloor M/M_d \rfloor$. If the p-value from the test is above the selected significance level, then the channel is not considered to be under an amplify-and-forward relay attack.

6.1.5 Step 5: Challenge-Response Authentication

This protocol also implicitly provides challenge-response authentication. There are three scenarios for which an attacker can attempt false authentication. The first is when the attacker forwards the legitimate relay. This would be detected and rejected in the amplify-and-forward detection phase. The second and third scenarios occur when the attacker sends a random response or when the attacker replays a previous legitimate authenticated message M_1 (6.1). In both scenarios, if the attacker does not have the secret session key, then when the verifier performs the decryption in the reveal phase, it will decrypt a uniformly random and uncorrelated samples for $\mathbf{p}'_1, \mathbf{R}'_1 \in \mathbb{D} \subset \mathbb{C}$, assuming a proper encryption algorithm (e.g., AES) was used in the reveal phase. The set $\mathbb{D}$ represents the data type (e.g., Double or Float) of the transmitted symbols. The set of possible pilots $\mathbf{p}_2$ is a subset of $\mathbb{D}$, since the powers are limited to E_s . Similarly, $\mathbf{R}_2$ exists in an even smaller subset, as it includes the pathloss on E_s . This means that the respective norm and Frobenius norm of a uniform $\mathbf{p}'_1$ and $\mathbf{R}'_1$ will likely be much larger than $\mathbf{p}_2$ and $\mathbf{R}_2$. This leads to $\hat{\mathbf{H}}_1$ dominating $\hat{\mathbf{H}}$ in (6.9), which results in $\hat{\mathbf{H}}$ have a mostly uniform distribution from the failed decryption. This will result in a failure during the normality check. Thus the protocol inherently provides challenge-response authentication, as it requires current secret knowledge from the secret key and channel for the response message (6.9), which they cannot know under the assumptions of the channels and their distributions.

6.2 Feedback Size and Complexity

The number of bits that need to be transmitted from the prover to the verifier is given by

$$\begin{aligned} \text{Bits Transmitted} &= 2LNM + 2LN \\ &= 2LN(M + 1) \end{aligned} \tag{6.12}$$

where N and M are the number of subcarriers and number of OFDM pilot exchanges, respectively. For example, if $L = 16$ bits, $N = 64$, and $M = 100$, then the bits transmitted from the prover to the verifier is ≈ 25.9 KB. When compared to Protocol 1 (25.6 KB), this protocol has to transfer an extra column of data to reveal the pilots. While marginal, this means that Protocol 3 uses strictly more data than Protocol 1, even if the exchange size is the same. Furthermore, the normality testing requires many independent samples, which can dramatically increase the number of exchanges required, if the coherence time is large. On the other hand, the other protocols require very little transmissions, as only a single transmission is needed.

The worst case running time of this protocol is mostly given by $\Omega(M^2)$ ($M > N$), which is the running time of the decimation process, as the $N \times M$ $\hat{\mathbf{H}}$ matrix is traversed for the normality test.

6.3 Simulation Experimentation

This section looks at the Monte-Carlo results from the simulations. The performance metrics that observed are the false-positive rate and the false-negative rate of the ability to detect an amplify-and-forward relay attack. The simulations were performed in MATLAB with the same overall system as Protocol 1 in Section 4.3. Each experiment was repeated 1000 times at each SNR.

The parameters for the amplify-and-forward simulations are shown in Table 6.2. The simulation is limited to $K = 1$, $\rho_0 = \rho_1$, and $G_0 = 1$. This focuses on a single relay. The significance level for the p-value is chosen as 0.01 that is commonly acceptable for cryptographic purposes [68]. The threshold for the coherence time and bandwidth are selected based on realistic channel conditions [144]. A coherence bandwidth smaller than $b_{c_{Th}} = 500$ KHz is unlikely in an indoor or outdoor setting and a coherence time of $\tau_{c_{Th}} = 5$ ms at 2.4 GHz is unlikely.

Table 6.2: AF Simulation parameters [128]

	Parameter	Value
1	Symbol Duration (μs)	1
2	Carrier Frequency (GHz)	2.4
3	RMS Delay Spread (ns)	250
4	Distance (m)	1
5	CNR (dB)	-
6	Number of Exchanges (#)	100000
7	Number of Pilots (#)	64
8	Speed (m/s)	1
9	Power-Delay Profile	Exponential
10	Sig-level	0.01
11	$b_{c_{Th}}$ (MHz)	1
12	$\tau_{c_{Th}}$ (ms)	2

The results for the FPR (4.33) and FNR (4.34) can be seen in Figure 6.4 and Figure 6.5, respectively.

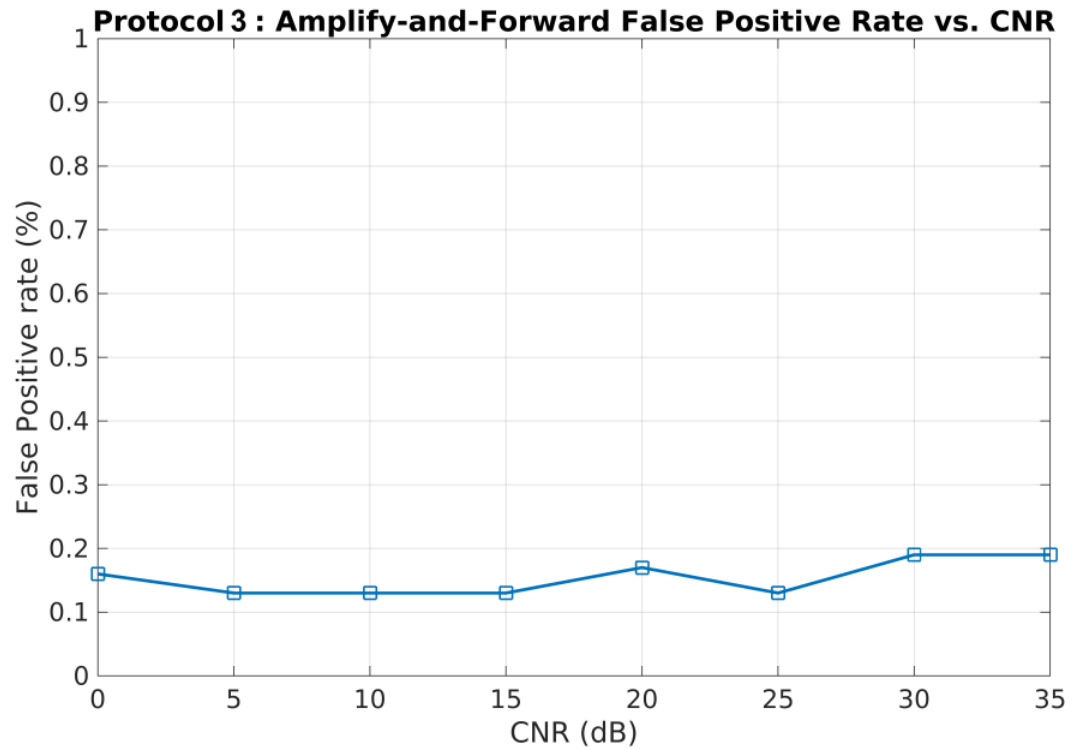


Figure 6.4: False-positive rate for detecting amplify-and-forward relay attack versus CNR [128]

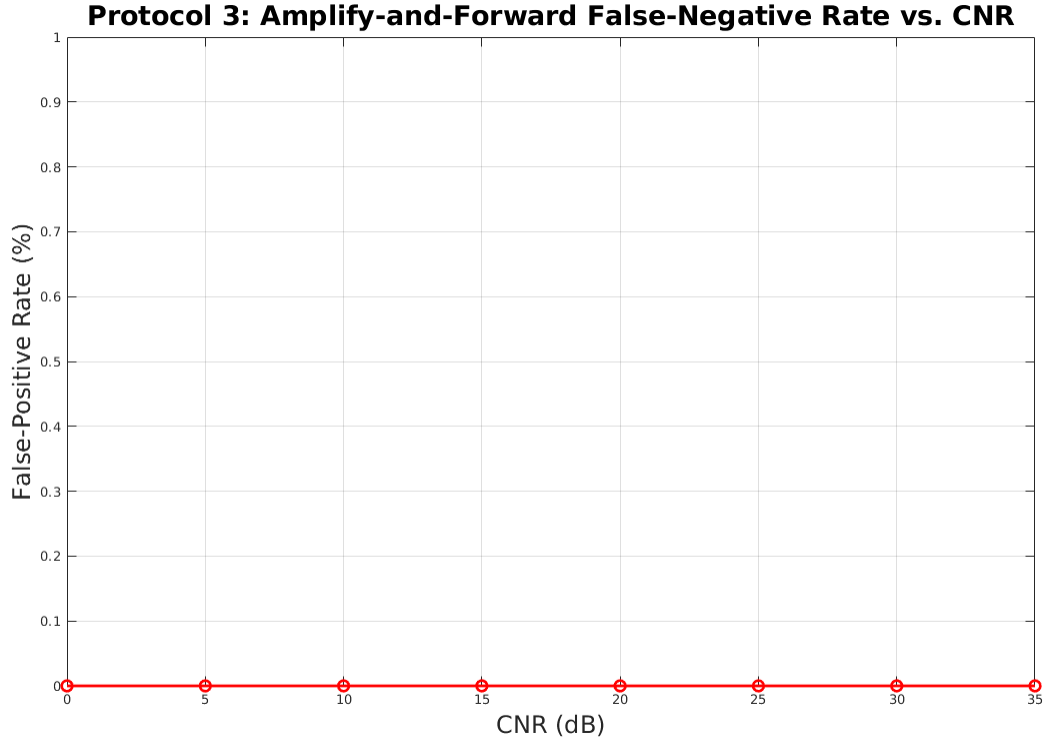


Figure 6.5: false-negative rate for detecting amplify-and-forward relay attack versus CNR

The results show that the FPR and FNR are relatively constant over the different noise powers, with an average of 15.4 % and 0 %, respectively. The FPR appears to be noise independent, as more noise does not affect the performance. A possible reason for this occurrence could be due to the fact that more noise causes a channel that has Gaussian distribution (i.e., no amplify-and-forward relay attack is occurring) to continue to have Gaussian distribution. This is because noise has a Gaussian distribution and it is being added to the uncorrelated Gaussian channel. Overall, the non-zero FPR may imply that the S-W test is giving a knowingly false results. It is possible that more samples should be required or a better normality test could be utilized. It may also be a result of suboptimal thresholds being selected, which could be explored in the future. On the other hand, the FNR has good overall performance, as the S-W test appears to be sensitive to non-Gaussian channels at all the noise powers.

Chapter 7

One-Way Amplify-and-Forward Relay Detection Protocol under Active and Passive Attacks using a Deep Neural Network

An important observations from Protocol 3 (Chapter 6) is that detecting an amplify-and-forward relay attack using a channel based approach can be done through the observation of some features of the channel. For example, Protocol 3 looked at the distribution of the channel as it gets cascaded through multiple transmissions by a relay. Comparing these features to an established baseline in a channel with no amplify-and-forward relay attack can allow a verifier to detect a relay attack. The selection of the optimal feature becomes an important problem in using the channel to detect amplify-and-forward relay attacks. Instead of spending time performing trial-and-error on different features to analytically and experimental identify performance, using a deep neural network allows for abstracting away the optimal channel feature and lets the neural network optimize on its own in the training process.

One of the issues with neural networks is overfitting of training data. While there are many ways a neural network can reduce overfitting, gathering robust trainig data sets is generally considered one of the most important requirements. When considering a wireless communication channel, which would be the input to the neural network, it is quite difficult in real life to gather diverse data sets of different channels experimentally. There are many parameters, including the delay spread, SNR, number of relays, Rician factor, gains, etc.

To compensate, statistically generated synthetic data (i.e., simulations) can be used to generate highly variable data sets, which can be mixed with real data for training. When considering past work on neural networks for detecting relay attacks (discussed in Section 2.4), a focus was given towards looking at device identification using a neural network. The issue with this methodology is that there is no robust field of knowledge for generating synthetic data for device non-idealities. Different devices have different components, with different profiles and are made of different materials, etc. It may be possible to hyperfocus on a single device, but training on many devices for practical uses is likely not possible, or at the very least not easy. On the other hand, statistical analysis of wireless communication channels is a robust field, making the generation of synthetic data straightforward. Using a neural network on a channel to identify a relay attack becomes a binary classification, which is simpler for a neural network to optimize, then the one-hot classification for device identification.

This chapter proposes the use of a deep neural network trained on synthetic data for detecting amplify-and-forward relay attacks. This protocol also leverages the keyed filter introduced in Protocol 2 (Chapter 5) to protect against active relay attacks. When compared to Protocol 3 (Chapter 6), this protocol does not require large pilot exchanges to perform real time feature evaluation, as the deep neural network offloads the feature evaluation to the training process. A single keyed filtered pilot is sent, which allows for the obscured channel estimation that prevents active attacks, like in Protocol 2 (Chapter 5). The protocol can be seen as a divergence from Protocol 2, where the verifier utilizes the channel estimate that it has in step 1 of Protocol 2 as the input to a neural network. This allows for the cascading of Protocol 2 and this protocol, which can provide both decode-and-forward and amplify-and-forward detection capabilities simultaneously. Simulation results are used as preliminary to evaluate this protocol.

7.1 Protocol Details

For this protocol, the following terminology is defined for use in the protocol:

- CK is an 128-bit encryption key generated from a symmetric-key K using a KDF and is only known by the prover and verifier,
- IK is an 128-bit authentication key generated from a symmetric-key K using a KDF only known by the prover and verifier,

- $\text{Enc}_{CK}\{\cdot\}$, $\mathbb{C}^N \rightarrow \mathbb{C}^N$, is a secure symmetric-key encryption that takes an input symbol and generates an encrypted output symbol using CK ,
- $\text{MAC}_{IK}\{\cdot\}$, $\mathbb{C}^N \rightarrow \mathbb{C}^N$, is a secure message authentication code that takes an input symbol and generates a tag output symbol using IK ,
- $\mathbf{p} \in \mathbb{C}^N$ is pilot vector, which contains the frequency sample and is publicly known,
- $\mathbf{h}_{xy,m} \in \mathbb{C}^N$ is an FIR filter representing the channel from y to x at the m^{th} time instance,
- $\mathbf{h}_k \in \mathbb{C}^N$ is an FIR filter defined as the keyed filter that is generated using CK and an encryption function to generate pseudorandom bitstream that is then quantized,
- $\mathbf{W}_m$ is a matrix of AWGN samples $\sim \mathcal{N}(0, N_0)$.

The protocol operates like a mixture of Protocol 2 (Chapter 5) and a normal challenge-response authentication protocol. From Protocol 2, this protocol utilizes the random filters to obstruct the real channel from the relay attacker. This prevents active relay attacks. Instead of using the challenge filter and the MSE, the protocol utilizes an offline model from a deep neural network (DNN) to make the binary classification.

Table 7.1: Steps of the Channel Based Relay Attack Detection Protocol

step	Details
1	Prover initiates protocol by transmitting request and a nonce $\mathbf{x}_{req}$ filtered through a random filter $\mathbf{h}_{K_1}$.
2	Verifier confirms request and generates a nonce $\mathbf{x}_{n_2}$ and a filter $\mathbf{h}_{K_2}$ using CK and $\mathbf{x}_{n_2}$. The nonce is transmitted through the filter to the prover
3	Prover uses the received nonce $\mathbf{x}_{n_2}$ and CK to generate $\mathbf{h}_{K_2}$ and equalizes the received pilots and estimates $\mathbf{h}_{pv,0}$ as $\tilde{\mathbf{h}}_{pv,0}$. Prover encrypts $\tilde{\mathbf{h}}_{pv,0}$, generates a MAC using CK and IK , and transmits both.
4	Verifier uses $\tilde{\mathbf{h}}_{p,0}$ with a DNN to determine if a relay attack is happening or not. With a sigmoid function as the activation function, if the output is ≤ 0.5 , then no attack is happening. If it is > 0.5 , then a relay attack is happening.

7.1.1 Step 1: Initiation

In the first step, the prover initiates the protocol by sending a request symbol transmission $\mathbf{x}_{req}$ to the verifier. The first transmission with a pilot must occur from the prover to the verifier, as the verifier must not reveal any channel state information to a potential relay before the usage of a random keyed filter. The symbol $\mathbf{x}_{req}$ is a fixed OFDM data symbol in the frequency domain.

The prover also filters the transmission using a random filter $\mathbf{h}_{K_1}$. This filter is necessary if the channels are reciprocal. Otherwise, the attacker can estimate $\mathbf{h}_{pv,m}$ by assuming $\mathbf{h}_{pv,0} \approx \mathbf{h}_{vp,0}$. This is the exact same initiation as Section 5.1.1 in Protocol 2.

7.1.2 Step 2: Nonce Transmission

After the verifier receives a request, it will check that the demodulated symbol matches the expected request $\mathbf{x}_{req}$. If it does, then the verifier will generate a random nonce that has ≥ 128 -bit length, where the length is known to both parties beforehand. The nonce will be used to provide the challenge in the challenge-response authentication. Specifically, the nonce is the seed or initial value (IV) utilized for the stream cipher encryption and MAC generation in subsequent steps. The nonce is transmitted in an OFDM symbol $\mathbf{x}_n$.

The second goal of this step is for the verifier to generate a random keyed filter ($\mathbf{h}_{K_2}$) using the encryption key (CK) and the nonce. The purpose of the keyed filter is to prevent the relay from recovering the channel from the verifier to relay. Again, this step is the same as Figure 5.4 in Protocol 2. The keyed filter can be generated by the prover and removed, but the attacker cannot do the same, as they do not know the secret keys.

7.1.3 Step 3: Channel Feedback

In this step, the prover provides feedback to the verifier of the estimated channel $\tilde{\mathbf{h}}_{p,0}$. It is critical that the feedback is encrypted using a strong stream cipher and authenticated using a MAC. Encryption is required to prevent the attacker from eavesdropping the data. With no encryption, the attacker can perform an active filter, which results in the channel appearing like no relay attack is occurring. The MAC is applied for precautionary reasons to prevent the attacker from modifying the estimate.

7.1.4 Step 4: Amplify-and-Forward Relay Attack Detection using a DNN

To evaluate if a relay attack is occurring, the relay decrypts and authenticates the prover's channel $\tilde{\mathbf{h}}_{p,0}$, which should have the keyed filter $\mathbf{h}_{K_2}$ equalized. Afterwards, the channel is serialized, where the amplitude $|\tilde{\mathbf{h}}_{p,0}|$ and phase vector $\angle \tilde{\mathbf{h}}_{p,0}$ are put into the DNN model for binary classification. As mentioned, if the output is ≤ 0.5 , then no relay attack is occurring. If it is > 0.5 , then a relay attack is occurring. The amplitude is scaled to be within $[0,1]$ to ensure that the pathloss is irrelevant in the evaluation.

The neural network used has a depth of 5, with 750 neurons at each hidden layer. Between each layer dropout is used with a dropout percent of 40%. For evaluation, the input layer was 128 Float32 subcarriers from the estimated channel. This would a parameter count of 2,350,501 $((128 \times 750 + 750) + 4(750^2 + 750) + (751))$. The activation function is a sigmoid. The loss function is the binary cross entropy, and the stochastic gradient descent method uses ADAM, with a learning rate of 0.0001. Batching is also used, with random batch sizes of 32 with batch normalization. For training, an epoch count of 100 was used. A graphical example of the neural network can be seen in Figure 7.1.

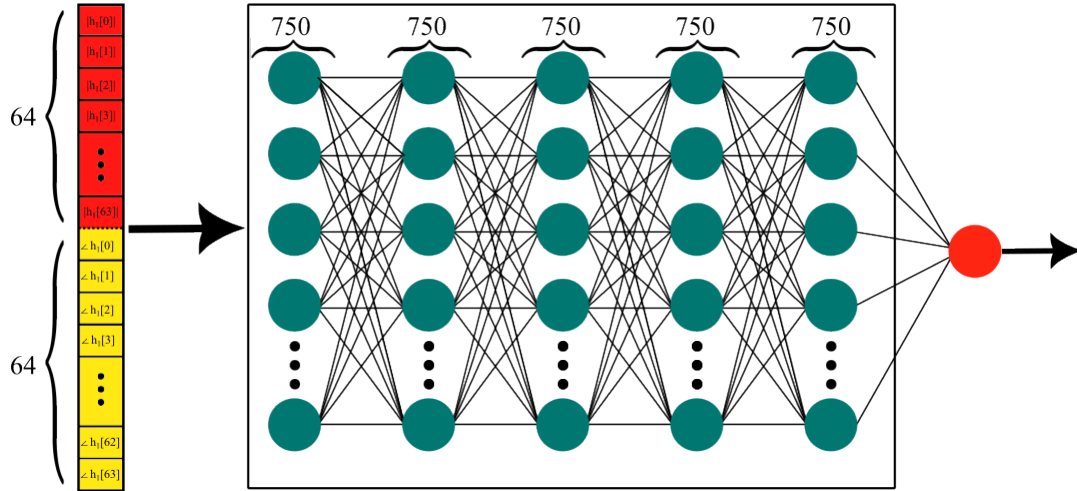


Figure 7.1: Basic visualization of the neural network used in Protocol 4

7.2 Simulation Experimentation

To create and evaluate the protocol, a simulation was used to generate synthetic data and to do the challenge testing. Training was done on an Nvidia RTX 4070 Ti Super graphic card with 16 GB of vRAM using the Flux library in the Julia programming language. The computer has an Intel i9-10900F CPU with 64 GB of RAM. Training data was generated by simulating channels. The channels were generated randomly by selecting from a set including the: CNR (5, 10, 15, 20, 25, 30) dB, a delay spread of (150, 200, 250, 300, 350, 400, 450, 500) ns, Rician factor (-100, 3) dB, and relay gain of 30 dB. The protocol was simulated 1,000,000 times to get the training data, represented as the scaled Float32 inputs. The protocol was simulated randomly under a $K = \{0, 1\}$ amplify-and-forward relay attack (i.e., no relay attack or relay attack). The velocity of the system was not considered, as only a single exchange is performed with the symbol duration of 1 us, which is much faster than any coherence time. The sampling rate was $f_s = 20$ MSPS, which also corresponds to a 20 MHz bandwidth. Each channel is padded with zeros to have a tap length of 64 samples, to ensure consistency when doing the comparison. Like the input vector, the training data is scaled to be within $[0, 1]$, to ensure that the pathloss has no impact on the evaluation. The training data was relatively small, with a total size of 512 MB. This represents a set of 1,000,000 simulated channels, consisting of 500,00 non-amplify-and-forward channels, and 500,00 amplify-and-forward channels. The simulation challenge was done at various CNRs, with 1000 FPR and FNR tests done for each one, respectively.

The FPR (4.33) and FNR (4.34) were evaluated using the training model. The FPR results on simulated pilot exchanges, can be seen in Figure 7.2.

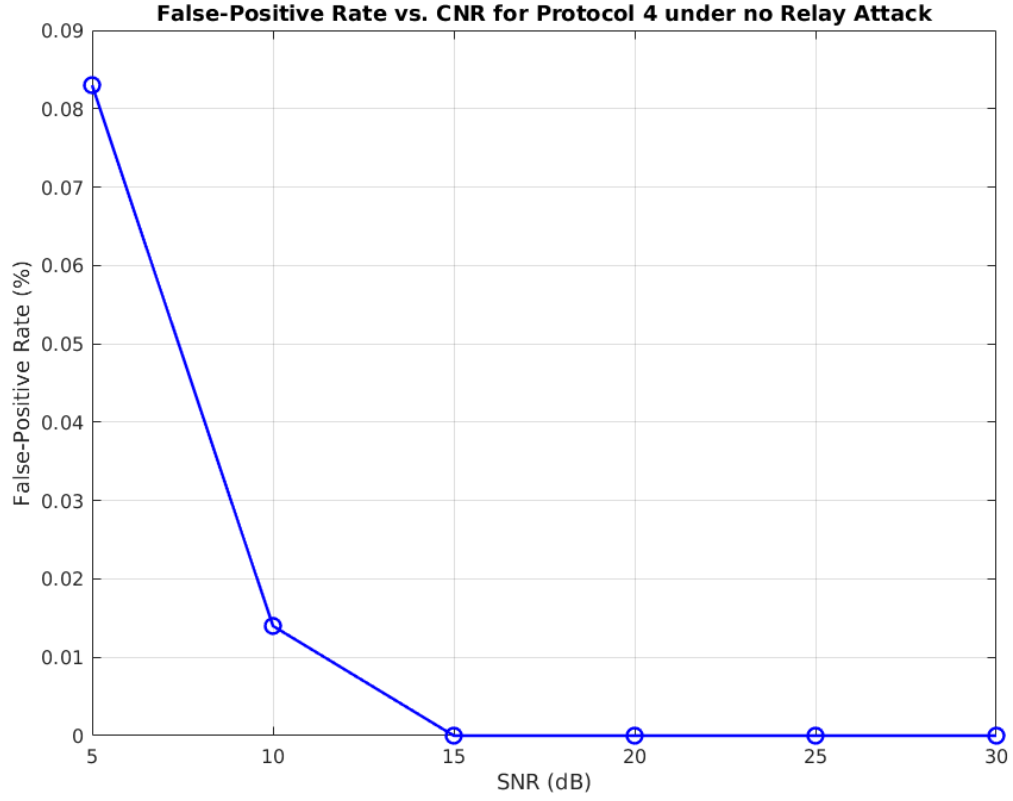


Figure 7.2: False-Positive rate for detecting amplify-and-forward relay attack versus CNR

Note, the scale on the FPR only rises to 9 % at 5 dB CNR. This shows that the DNN works well at detecting relay attacks and is better than the result seen in Protocol 3 (6.4).

Similarly, FNR results on simulated pilot exchanges, can be seen in Figure 7.3.

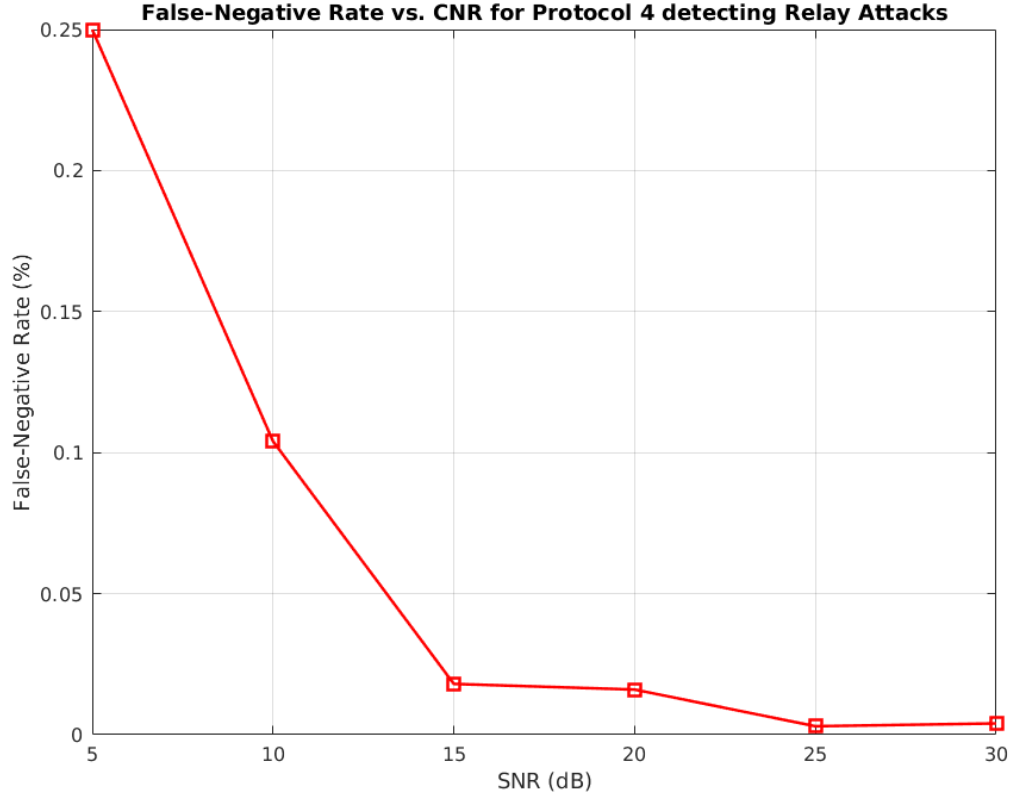


Figure 7.3: False-Negative rate for detecting amplify-and-forward relay attack versus CNR

These results are worse than the FPR results, with an FNR of 25 % at 5 dB CNR. Higher CNR inputs yield better results, and is possible more training data is needed to accomodate the noise, or more epochs are needed in the training phase.

When comparing the results to those seen in Protocol 3 (6.5), Protocol 4 generally achieves better performance, except at 5 dB CNR for the FNR. The protocol does not appear to be noise agnostic, as opposed to Protocol 3, which had consistent performance for ever CNR tested. This is expected, as the given CNRs do not significantly affect the distribution (for Protocol 3), but can have a significant effect on how a channel compares to the training data. Overall, it is hard to say which features of the channel have an effect without further weight analysis, which will be left to future work.

Chapter 8

Near Field Relay Attack Detection¹

All the protocols presented so far in this thesis make the assumption that the channel has a sufficiently low coherence bandwidth (i.e., a large RMS delay spread τ_m), such that the channel introduces sufficient randomness that some aspect of the random process can be utilized to identify relay attacks. This model is derived from uncorrelated multipaths that cause effects such as uncorrelated scattering, uniform scattering, spatial decorrelation, and Rayleigh distributions. In the far field this is a reasonable assumption to make, where all the fading channel effects can occur, as detailed in Section 2.3.2. In reality, not all relay attacks operate in the far field. As discussed in detail in Section 3.1, many relay attacks occur on HF RFID and NFC devices (e.g., credit cards, key fobs, etc.), where challenge-response authentication protocols are widely used, as seen in Figure 8.1. These communication systems are dominated by the equivalent of a strong single tap LOS channel, where mutual inductive coupling is used for both communications and device powering. Ideally, a proposed protocol should also work in the near field to detect relay attacks. This chapter looks at the specifics on why this problem is more difficult and also a proposed modified protocol that can detect passive relay attacks in the near field.

¹This is an accepted manuscript in IEEE RFID-TA 2024. R. Abubaker and G. Gong, “Evaluating Channel Based Relay Attack Detection in the near field” [158]

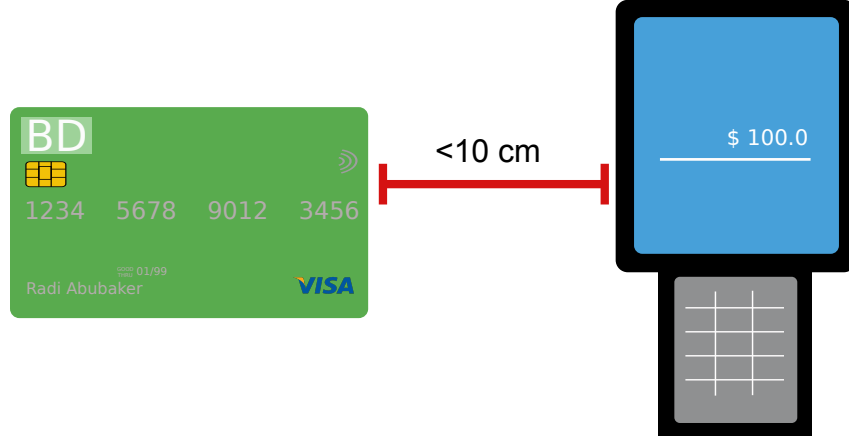


Figure 8.1: Example of near field challenge-response authentication with a HF RFID credit card and a payment terminal

8.1 Near Field Channel Model

A HF RFID or NFC system uses mutual inductance to perform communications. The system model can be perfectly represented through a circuit, where the channel model is a single tap with an amplitude and phase.

The basis of the proposed protocol for relay detection in the near field is the adaption of the CBRAD proposed in Chapter 5, where the assumption is that any HF RFID CFR from device y to device x is given by

$$D\{\mathbf{h}_{xy,m}\} = ke^{j\theta_{yx}} \mathbf{I} \quad (8.1)$$

$$= ke^{j\theta_{xy}} \mathbf{I}, \quad (8.2)$$

where $\{k \in \mathbb{R} | 0 \leq k \leq 1\}$ is the mutual inductance coupling coefficient and the phase $\theta_{yx} = \theta_{xy}$ comes from the phase effects (i.e., matching) of the equivalent circuit. The coupling coefficient is the magnitude of the CFR. From the circuit matching, it can also be assumed that the channel is reciprocal. The model assumes induction is being utilized, but this channel is equivalent to a far field LOS, where the Rician factor $K \gg 20$ dB. In this interpretation of the problem, the coupling coefficient is equivalent to pathloss, and the phase from the circuit is equivalent to the phase due to the delay and AoA. In other

words,

$$k = \sqrt{\rho} \quad (8.3)$$

$$= \frac{\lambda}{2\pi d}, \quad (8.4)$$

where ρ comes from the Friis free-space equation (2.3) and is utilized since the channel is assumed to be a LOS channel with little or no significant multipaths that would yield a different qualitative equation. Similarly, the phase can be represented as,

$$\begin{aligned} \theta_{xy} &= 2\pi (0.5 \sin(\theta_{\text{AoA}}) - \tau_m 2f_c) \\ &= \theta_{yx}, \end{aligned} \quad (8.5)$$

where the first term in (8.5) comes from the phase of the angle of arrival and the second term comes from the delay. Using this model to represent a HF RFID channel allows for the use of a wireless communication system as a simplification, without the need for the creation of an inductive system. The important factor is to ensure that the devices are close enough, such that the communications is LOS. As mentioned, this requires a Rician factor $K \gg 20$ dB. The LOS model given by (8.1) is constant for each subcarrier, where the IDFT results in a single tap,

$$\mathbf{F}^{-1}\{\mathbf{h}_{xy,m}\} = (ke^{j\theta_{xy}}, 0, \dots, 0) \quad (8.6)$$

The major issue that arises in this model is that it is impossible to utilize the channel based solutions to obscure the wireless channel, as in Chapter 4, Chapter 5, or Chapter 6, since the CFR is constant for all frequencies (i.e., there is variability in the phase or amplitude). This introduces a problems which prevent these protocols from being useful:

1. The constant LOS channel given in (8.6) does not have the Rayleigh distribution or temporal correlation properties found in the uncorrelated delay or uniform scattering channels. This prevents Protocol 1 (Chapter 4) from being able to work, resulting never being able to detect a relay attack (100% false-negative).
2. Similarly, the constant channel will never have a Gaussian distribution, unless it is below the noise floor. This prevents Protocol 3 (Chapter 6) from being able to work, resulting in a constant detection of relay attacks (100% false-positive).
3. The constant LOS channel makes it trivial for a relay attacker to perform an active relay since there is essentially no obscuring of the channels as the DFT of (8.6) has a constant amplitude and phase. This also makes it possible to directly attack every protocol presented, regardless of the use of obscuring filter like the keyed or challenge filters.

If the attacker understands that the channel is near field (e.g., HF RFID/NFC) or is a channel that has significant LOS component, then they can easily attack the proposed protocols. To simplify the problem this thesis will only propose a novel protocol related to the passive decode-and-forward relay attack problem.

8.2 Modified One-Way Decode-and-Forward Relay Detection Protocol for Passive Near Field Detection

This section details the motivation and design of the proposed CBRAD protocol. All communications in this protocol are OFDM baseband symbol transmissions, like in the previous protocols. The protocol assumes there is constant temporal coherence. While reciprocity is assumed to exist in this channel model, the protocol can accommodate non-reciprocal channels. The protocol also does not assume frequency-selective fading or a Rayleigh or Rician distribution of the channel, which might not occur in the near field since multipath effects and plane wave modeling might not hold. To compensate, a random filter is proposed to simulate frequency-selective.

The following terminology is defined for use in the protocol:

- CK is an 128-bit encryption key generated from a symmetric-key K using a KDF and is only known by the prover and verifier,
- IK is an 128-bit authentication key generated from a symmetric-key K using a KDF only known by the prover and verifier,
- $\text{Enc}_{CK}\{\cdot\}, \mathbb{C}^N \rightarrow \mathbb{C}^N$, is a secure symmetric-key encryption that takes an input symbol and generates an encrypted output symbol using CK ,
- $\text{MAC}_{IK}\{\cdot\}, \mathbb{C}^N \rightarrow \mathbb{C}^N$, is a secure message authentication code that takes an input symbol and generates a tag output symbol using IK ,
- $\mathbf{p} \in \mathbb{C}^N$ is pilot vector, which contains the frequency sample and is publicly known,
- $\mathbf{h}_{xy,m} \in \mathbb{C}^N$ is an FIR filter representing the channel from y to x at the m^{th} time instance,
- $\mathbf{h}_{ch} \in \mathbb{C}^N$ is an FIR filter defined as the challenge filter that is randomly generated complex sequence by the verifier,

- $\mathbf{W}_m$ is a matrix of AWGN samples $\sim \mathcal{N}(0, N_0)$.

The details of the CBRAD protocol can be seen in Table 8.1 with the visualization of the steps in Figure 8.2.

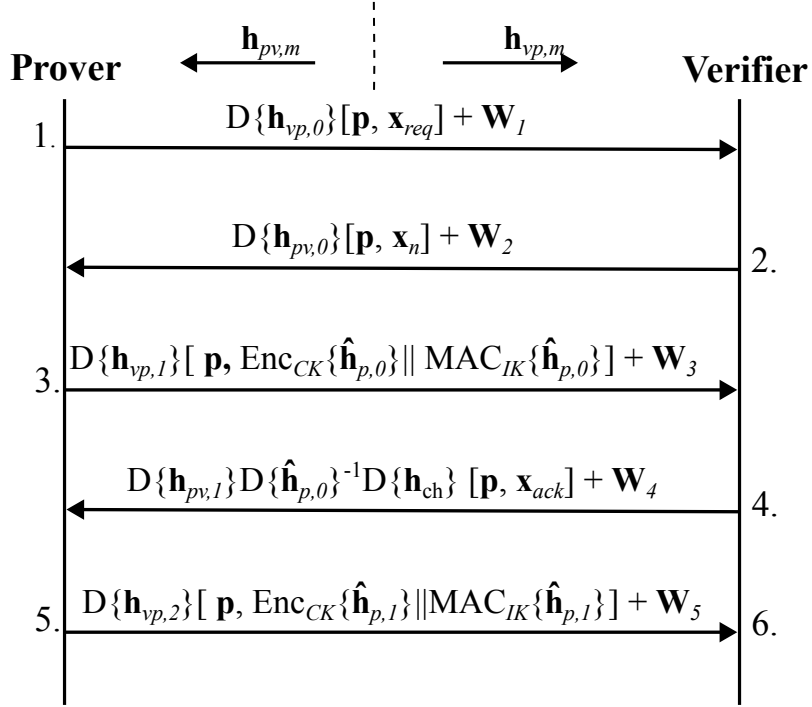


Figure 8.2: Message exchange of the Channel Based Relay Attack Detection Protocol [158]

8.2.1 Step 1: Initiation

In the first step, the prover initiates the protocol by sending a request symbol transmission $\mathbf{x}_{req}$ to the verifier. The first transmission with a pilot must occur from the prover to the verifier, as the verifier must not reveal any channel state information to a potential relay before the usage of a random keyed filter. The symbol $\mathbf{x}_{req}$ is a fixed OFDM data symbol in the frequency domain. This step differs from the step 1 in Section 5.1.1 as there is no attempt to protect the channel from observation against an active relay attacker.

Table 8.1: Steps of the Channel Based Relay Attack Detection Protocol [158]

step	Details
1	Prover initiates protocol by transmitting request, $\mathbf{x}_{req}$
2	Verifier confirms request, generates a nonce $\mathbf{x}_n$. The nonce is transmitted to the prover
3	Prover uses the received nonce $\mathbf{x}_n$ and received pilots to estimate $\mathbf{h}_{pv,0}$. Prover encrypts $\hat{\mathbf{h}}_{p,0}$, generates a MAC, and transmits both
4	Verifier generates a random challenge filter $\mathbf{h}_{ch}$ and transmits an acknowledgement $\mathbf{x}_{ack}$ through this filter while applying equalization using $\hat{\mathbf{h}}_{p,0}$
5	Prover estimates the challenge filter $\mathbf{h}_{ch}$, as $\hat{\mathbf{h}}_{p,1}$, encrypts it, and transmits it with a MAC
6	Verifier compares $\mathbf{h}_{ch}$ and $\hat{\mathbf{h}}_{p,1}$ using the MSE. If it is above a threshold Th , then a relay attack is occurring and authentication fails, otherwise no relay attack is occurring and authentication passes

8.2.2 Step 2: Nonce Transmission

After the verifier receives a request, it will check that the demodulated symbol matches the expected request $\mathbf{x}_{req}$. If it does, then the verifier will generate a random nonce that has ≥ 128 -bit length, where the length is known to both parties beforehand. The nonce will be used to provide the challenge in the challenge-response authentication. Specifically, the nonce is the seed utilized for the stream cipher encryption and MAC generation in subsequent steps. The nonce is transmitted in an OFDM symbol $\mathbf{x}_n$.

Unlike step 2 in Protocol 2 (Section 5.1.2), the nonce is used exclusively as an IV for the key generation as there is no use of a keyed filter, as mentioned in Section 8.1.

8.2.3 Step 3: Channel Feedback

In this step, the prover provides feedback to the verifier of the estimated channel $\tilde{\mathbf{h}}_{p,0}$. It is critical that the feedback is encrypted using a strong stream cipher and authenticated using a MAC. Encryption is required to prevent the attacker from eavesdropping the data. With no encryption, the attacker can perform an active filter, which results in the channel

appearing like no relay attack is occurring. We show this in Section 5.1.3. The MAC is applied for precautionary reasons to prevent the attacker from modifying the estimate.

8.2.4 Step 4: Relay Challenge

In this step, after decrypting and verifying the feedback, the verifier uses $(\tilde{\mathbf{h}}_{p,0})$ to challenge the prover to confirm that it is directly talking to the verifier across a single channel, and not a decode-and-forward relay. Figure 8.3 shows the difference in step 4 when the decode-and-forward relay attack occurs and does not occur.

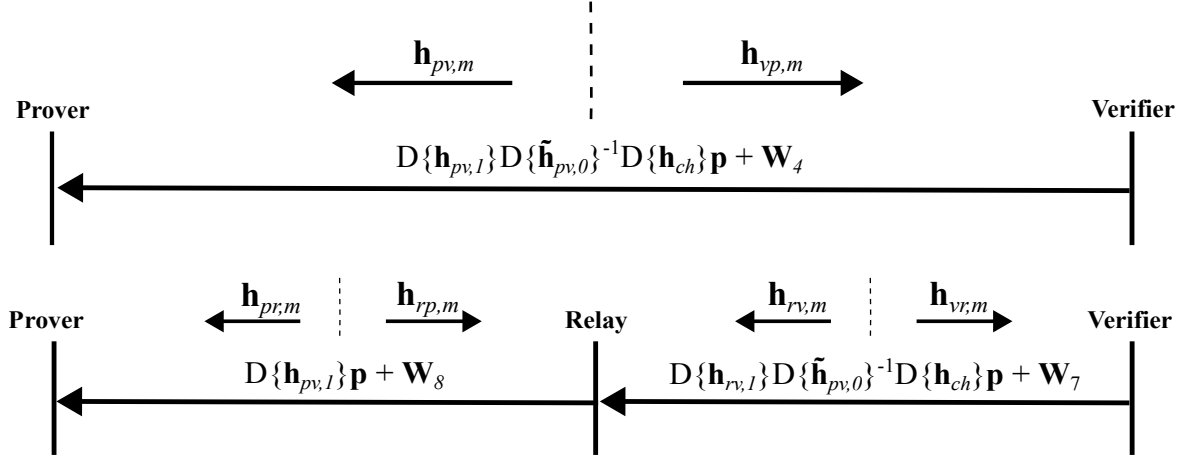


Figure 8.3: step 4: Without a decode-and-forward relay attack (top) and under one (bottom) [158]

When a decode-and-forward relay attack is not occurring, the verifier filters the transmission to equalize the channel from the verifier to the prover. If the steps have happened fast enough, then the estimate $\tilde{\mathbf{h}}_{p,0}$ should be close to $\mathbf{h}_{pv,1}$, thus

$$\begin{aligned}\tilde{\mathbf{h}}_{p,1} &= D\{\hat{\mathbf{h}}_{pv,1}\}D\{\tilde{\mathbf{h}}_{p,0}\}^{-1}\hat{\mathbf{h}}_{ch} \\ &\approx \hat{\mathbf{h}}_{ch}\end{aligned}\tag{8.7}$$

which comes from the assumption that $D\{\mathbf{h}_{pv,1}\}D\{\tilde{\mathbf{h}}_{p,0}\}^{-1} \approx \mathbf{I}$. This means that the prover can estimate a channel that is ideally close to the challenge $\mathbf{h}_{ch}$ that the verifier has generated.

If a decode-and-forward relay attack is occurring, then the prover will not receive the challenge channel ($\mathbf{h}_{ch}$)

$$\tilde{\mathbf{h}}_{p,1} = \hat{\mathbf{h}}_{pr,1} \quad (8.8)$$

which will have a large MSE with $\mathbf{h}_{ch}$, since the channels are uncorrelated/independent. The relay will see the channel

$$\hat{\mathbf{h}}_{r,1} = \mathbf{D}\{\hat{\mathbf{h}}_{rv,1}\}\mathbf{D}\{\tilde{\mathbf{h}}_{p,0}\}^{-1}\hat{\mathbf{h}}_{ch} \quad (8.9)$$

where $\mathbf{h}_{rv,1}$ and $\tilde{\mathbf{h}}_{p,0}$ are uncorrelated, which should obscure $\mathbf{h}_{ch}$ from the relay. If the attacker knew what $\mathbf{h}_{vr,0}$ was, it would be possible for them to estimate the proper filter to apply an active relay attack to the prover.

8.2.5 Step 5: Prover Generation

In step 5, the prover repeats the processes in step 2 (Section 8.2.2), where they provide the encrypted and authenticated channel estimate to the verifier, so the final authentication can occur. As mentioned, if no decode-and-forward relay attack is occurring, the prover will estimate $\hat{\mathbf{h}}_{ch}$ (8.7). If a decode-and-forward relay attack is occurring they will estimate $\hat{\mathbf{h}}_{pr,1}$ (8.9). Encryption and authentication are applied as a precaution, though they likely aren't necessary as only the prover that has the channel $\tilde{\mathbf{h}}_{p,0}$ should be able to recover $\hat{\mathbf{h}}_{ch}$. The size of the transmission is the same as in Protocol 2 step 2 (Equation 5.18).

8.2.6 Step 6: Decode-and-Forward Relay Detection with Authentication

In step 6, after decrypting and verifying the feedback, the verifier checks that the received feedback $\hat{\mathbf{h}}_{p,1}$ has a low MSE with the transmitted challenge ($\mathbf{h}_{ch}$). The MSE of $\hat{\mathbf{h}}_{p,1}$ and $\mathbf{h}_{ch}$ is measured as

$$\text{MSE} = \frac{1}{N} \left(\hat{\mathbf{h}}_{p,1} - \mathbf{h}_{ch} \right)^H \left(\hat{\mathbf{h}}_{p,1} - \mathbf{h}_{ch} \right) \quad (8.10)$$

If $\text{MSE} \leq Th$, then the verifier concludes that the prover has (1) the secret key (2) no decode-and-forward relay is occurring. The threshold Th is selected heuristically, with higher threshold values increasing the false-positive rate, while lower the false-negative rate, for a given signal-to-noise ratio (SNR).

8.2.7 Security Analysis

In general, the security is the same as subsection 5.2.1. If the devices are communicating directly to each other, then $\mathbf{h}_{ch} \approx \mathbf{h}_{pv,1}$, where $\hat{\mathbf{h}}_{p,1} \approx \mathbf{h}_{pv,1}$ is an estimate of $\mathbf{h}_{pv,1}$. The verifier uses the MSE to evaluate both the challenge-response authentication and to determine if a relay attack is occurring,

$$\text{MSE} = \frac{1}{N} \left(\hat{\mathbf{h}}_{p,1} - \mathbf{h}_{ch} \right)^H \left(\hat{\mathbf{h}}_{p,1} - \mathbf{h}_{ch} \right). \quad (8.11)$$

Challenge-response authentication is provided from the encryption and authentication response in step 3 (8.2.3). If the prover does have CK and IK from the original symmetric-key, then the verifier will receive and decrypt a random $\hat{\mathbf{h}}_{p,1}$, resulting in an extremely large MSE.

The security of the protocol against relay attacks comes from the assumption that in step 4, an attacker would demodulate and decrypt the pilot and acknowledgement $[\mathbf{p}, \mathbf{x}_{ack}]$, removing the effects of the challenge filter $\mathbf{h}_{ch}$, causing the prover to estimate something different. This will result in a large MSE for the verifier. The random filter is generated through a pseudorandom bit generation, conversion to complex numbers, then interpolation, as detailed in subsection 5.2.1.

8.3 Hardware Experimentation

In this section hardware experiments are performed on Libre and Pluto SDRs to evaluate the practical performance of the proposed protocol for detecting relay attacks in the near field and far field. The performance criterion under evaluation are the FPR and FNR. To control the SDRs implementing the verifier and relay remotely, these devices are each connected to a Raspberry Pi 5, remote desktop, and a battery pack. To control the signal-to-noise ratio (SNR), the signal and noise power are measured to get an estimate of the SNR in the experiment, absent any interference with a 23 dB PA, and the SDR transmission gain is scaled back to achieve the desired SNR. Each SNR experiment repeated 30-50 times to collect the FNR and FPR. The transmission frequency used is 915 MHz, which has a wavelength of 0.33 m. At this frequency, the near field is considered to be any distance ≤ 2 m and the far field to be any distance > 2 m. The communication system implemented is 64 subcarrier with QPSK-OFDM transmission utilizing a 10 MHz bandwidth, with a 25 % cyclic-prefix. The pilot $\mathbf{p} = \text{F}\{\text{FZC}(8)\}$ is the DFT of a Frank-Zadoff-Chu (FZC) sequence. Channel estimation utilizes the least square estimator

(LSE) and symbol estimation using maximum-likelihood estimation (MLE). Details of the hardware implementation are given in Chapter 9. Specifically, the SDR platform, OFDM system, waterfilling, channel estimation, equalization, decode-and-forward relay attack, and synchronization processes are detailed. To verify that the relay is working during for the protocol, a challenge-response authentication is simultaneously relayed. The experiment setup can be seen in Figure 8.4.

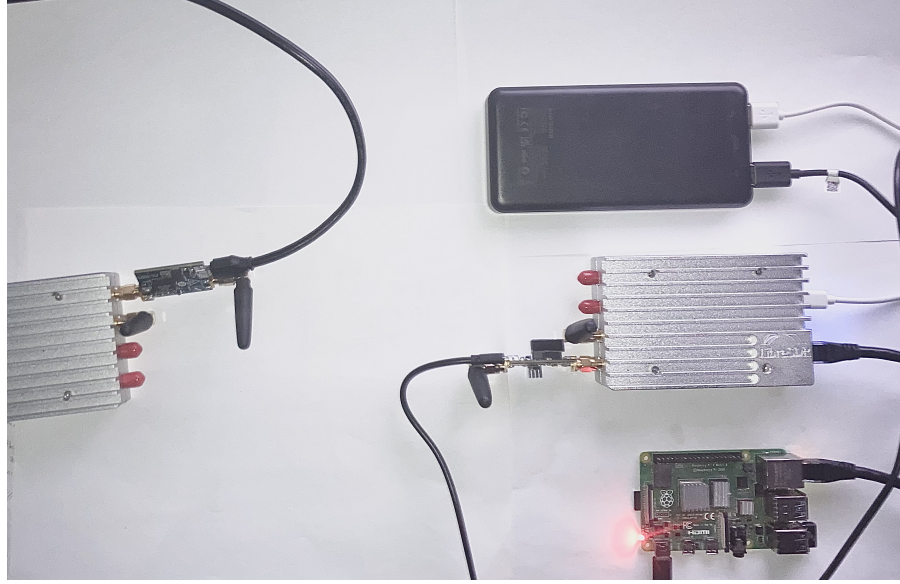


Figure 8.4: Experiment setup [158]

8.3.1 False-Positive Experiment

The FPR is defined in (4.33). The experiment is performed with the prover and verifier at distances of 1 and 4 m with line-of-sight (LOS), over a SNR range of 5–30 dB at the verifier. The results of the experiment can be seen in Figure 8.5.

Note that the far field results only goes to 20 dB. The results show that even in the near field (1 m), the devices are capable of distinguishing relay attacks well. The independent variable that has the largest effect is the SNR, not the distance. The SNR can negatively effect performance by introducing symbol errors to the MLE and channel error to the LSE. Any symbol errors will result in a false-positive and channel error can increase the MSE, potentially resulting in a false-positive.

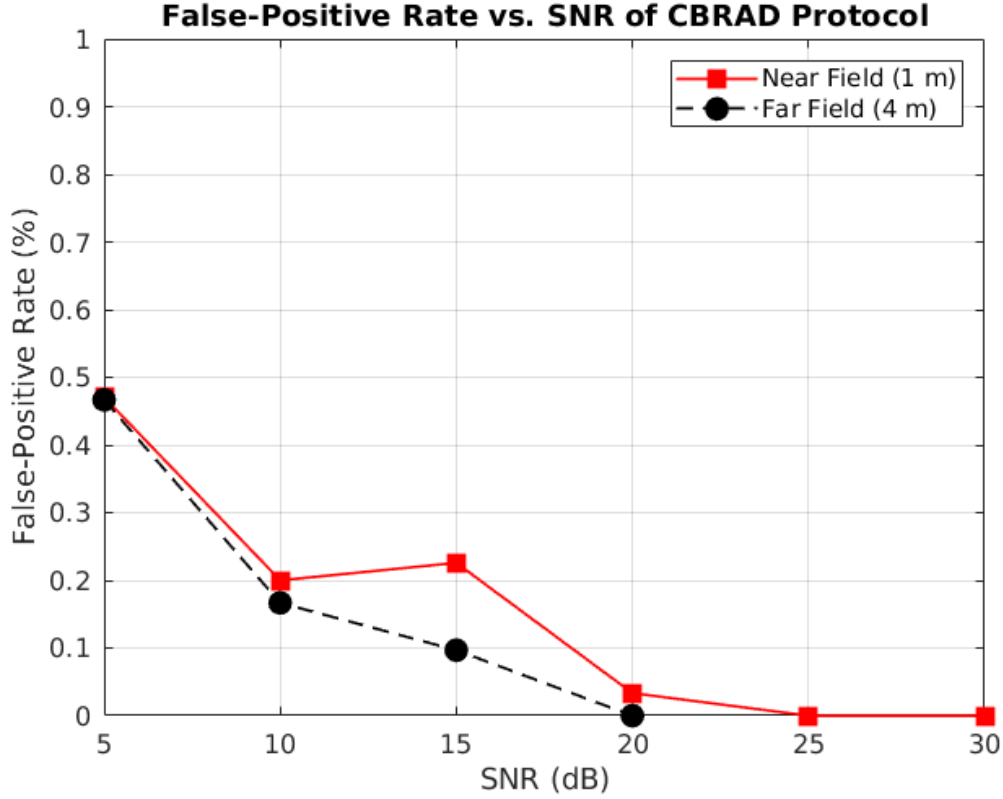


Figure 8.5: False-Positive Rate versus SNR at various distances between the prover and verifier [158]

8.3.2 False-Negative Experiment

The FNR is defined in (4.34). The experiment is performed with the relay and verifier at distances of 1 and 4 m with LOS and the prover constantly 1 m away with LOS from the relay, over an SNR range of 5–20 dB at the verifier. The results of the experiment can be seen in Figure 8.6.

The results are better than that for the FPR, with good performance in the near field and far field. Since any error will increase the likelihood of detecting a relay attack, SNR does not have the same negative impact as it did for the FPR. The main performance loss was from the selection of a nonoptimal threshold for the MSE in step 6. For comparison, it is difficult to find in literature false-negative rates that actually perform a relay attack to evaluate the FNR, especially with varying SNRs. The latest work by M. Fomichev *et al.*

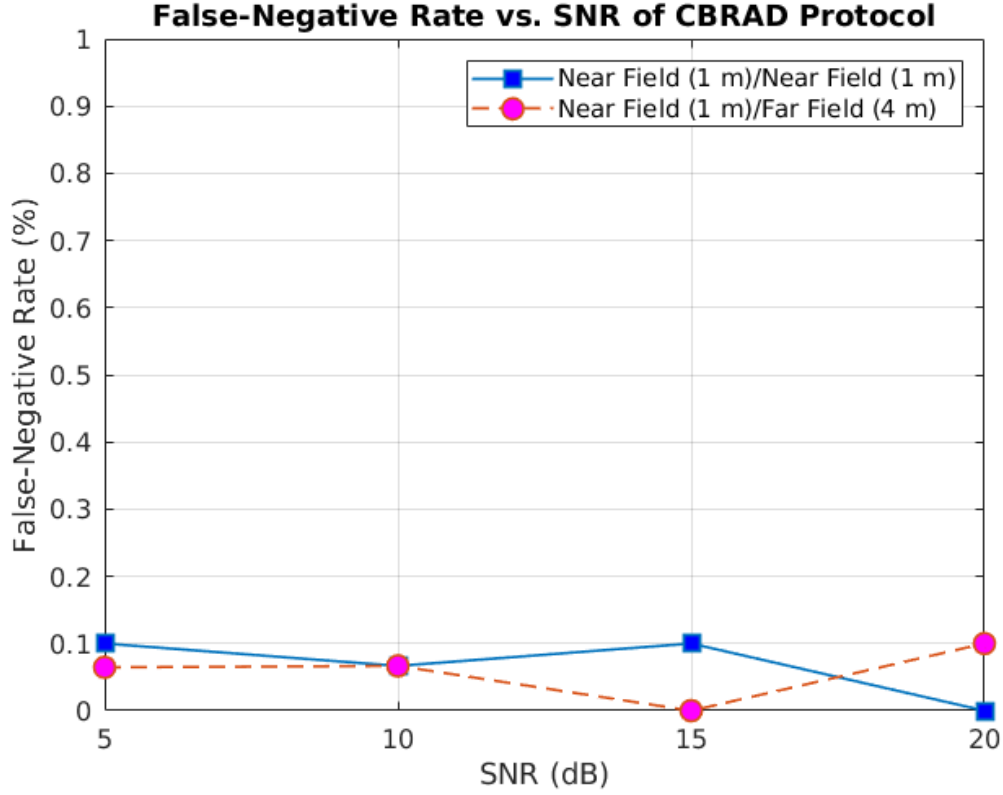


Figure 8.6: False-Negative Rate versus SNR in the near field and far field under a relay attack [158]

achieves a 4 % error for distance measuring [115].

Compared to the results in Section 5.2.3, where the devices were kept at a larger distance and solely in the far field, the near field performance seems to be worse for the FPR, which is likely caused by the fixed position it was in. This causes any poor performance in the LOS term to have a consistent effect at every subcarrier. On the other hand, the far field performance can benefit from an averaging from the error over all the multipaths. The FNR results are quite similar to the far field result, which lends positively to the proposed processes for detecting relay attacks. Any error is increased by worse noise, which does not have a negative effect on the FNR.

Chapter 9

Hardware Implementation

This chapter looks at the hardware implementation of the challenge-response authentication, relay attack, and the implementation of select protocols on software-defined radios required to achieve reliable OFDM communications. Each protocol and attack implementation is novel for software-defined radios in literature, and is one of the major contributions of this thesis. The base of the implementation is an OFDM communication protocol, which will also be detailed. A hardware overview of the devices used for the implementation is also given. A focus is placed on the embedded software implementation on a modified Analog Devices ADALM-Pluto SDR and the algorithms necessary to implement the protocols.

9.1 Motivation

When selecting hardware to test the relay attack, there are limited options to choose between. For channel based protocols, like for physical-layer key generation, the most common hardware platform selected in literature is a WiFi (IEEE 802.11n) COTS network interface card (NIC) that can extract channel information from the preamble [57, 69, 113]. This information is typically discarded at the physical-layer, before the packet is passed to the link-layer. Custom drivers, like those made by D. Halperin *et al.*, are required to access this information [120]. While these systems are easy to use, they have several limitations. The first major issue is that you are only able to extract information using these tools. While it is possible to forge the header and data of WiFi packets to perform data exchanges and to choose the transmission modes (e.g., bandwidth, throughput, etc.), there are no COTS platforms that allow for the altering of the physical-layer preamble or any other aspect of the signal that is outside the standard specifications, which is usually

the IEEE 802.11n standard [69]. For example, this makes it impossible to evaluate a protocol that deviates in any way from what could be implemented with IEEE 802.11n. Changing the pilots or applying any filter to the signals is not possible. In context of a relay attack, it is only possible to perform passive decode-and-forward relaying. Since binary data at the link-layer and channel information from the physical-layer is the only information accessible; it is not possible to relay an analog signal. In particular, noise is stripped and it is not possible to re-apply the channel effects through an FIR filter. For the IEEE 802.11n methodology, another issue is the limited resolution, as samples only contain 8-bit signed real and imaginary components $([-128, +127])$. The NIC may also have to interact with an OS, which can add significant stochastic delay to any protocol. In general, other platforms for Bluetooth and LTE have similar limitations, where only the bits in the packet can be forged and some channel information can be extracted (i.e., no signal level control), which does not make it possible to implement or perform amplify-and-forward or active relay relay attacks.

To circumvent these constraints on COTS device, it is necessary to pick a hardware platform where the whole signal can be manipulated. Other than an application specific device, an SDR is the only other general purpose device that is capable of implementing the protocols and attacks presented in this thesis. This chapter looks at how this done, with a focus on the details of the novel implementation. The specific contributions of this chapter include:

1. The first SDR implementation in literature of a decode-and-forward and amplify-and-forward relay attack,
2. A novel pilot generation method utilizing a generalized Milewski structure,
3. A new timing synchronization algorithm specifically for an embedded SDR implementation for research,
4. A novel waterfilling and adaptive modulation algorithm that considers security from eavesdroppers,
5. A look at the performance overview of the Analog Devices ADALM-Pluto SDR.

9.2 Pluto SDR

The basic details of an SDR are described in Section 2.5. The SDR platform used for the implementation in this thesis is an Analog Devices ADALM-Pluto SDR, colloquially called

the Pluto SDR. The specifications of the processing components in the Pluto SDR are described in Table 9.1. The Pluto SDR has a fully open source design, with the software and hardware design being openly available. This includes hardware description for the FPGA, drivers to control the components, and schematic and bill of material to recreate the Pluto SDR.

Table 9.1: Major processing components in the Analog Devices ADALM-Pluto SDR

Component	Manufacturer	Part ID	Specification
FPGA	Xilinx	Zynq Z-7010	28K LE / 17,600 LUTs
Processor	ARM	Cortex-A7	1×500 MHz, L1-32 KB, L2-512 KB
RAM	Micron	-	DDR3 512 MB@525 MHz
Memory	Micron	-	32 MB Flash
Comp. Interface	Microchip	Transceiver	USB 2.0
OS Driver	Analog Devices	IIO	v0.25

The Zynq Z-7010 system-on-chip (SoC) integrates all these components, with the FPGA and processor having the high-speed Advanced Extensible Interface (AXI) that can perform direct memory access (DMA) between them. The Pluto SDR runs an embedded Linux environment built using buildroot, which allows for the installation of a Linux OS and many libraries to run programs and self compiled code [121]. The major limitation with the Pluto SDR is that the OS is stored on the 32 MB storage and runs on memory, after being loaded from the storage. This prevents persistence and limits the programs that can be installed and their capabilities. For example, collecting and storing large datasets and models is not possible exclusively on the Pluto SDR. Like most SDRs, the Pluto SDR has a USB 2.0 interface that can connect to a computer to exchange data with. Using the Industrial Input/Output (IIO) structure, data can be exchanged back-and-forth between the Pluto SDR and the computer it is connected to. The radio frontend of the Pluto SDR utilizes the 1×1 Analog Devices AD9363 zero intermediate frequency (ZIF) integrated transceiver. The major aspects of the radio can be seen in Table 9.2.

Table 9.2: Specifications of radio transceiver on the ADALM-Pluto SDR [122]

Specification	Value	Unit
Min. f_c	325	MHz
Max f_c	3800	MHz
Samp. Rate	≤ 61.44	MSPS
Tx/Rx Bandwidth	0.2 – 20	MHz
DAC/ADC Reso.	12	bits
PA Gain	-89.75 – 0	dB
LNA Gain	-1 – 71	dB
Noise Figure	2.5 – 3.3	dB
IIP2	40 – 45	dBm
IIP3	-18 – -14	dBm
EVM	-34	dB

The transmit power profile of the Pluto SDR is not guaranteed, so for all transmissions power amplifiers (PAs) were used to ensure a sufficient power range.

9.2.1 Industrial Input/Ouput

To process the I/Q samples from the radio frontend of the Pluto SDR, a developer can either use the FPGA directly for signal processing or they can use the Industrial IO (IIO) framework to extract samples to process in hardware. IIO is an open source package of drivers and a framework for creating new drivers that allows for the controlling of input/output (IO) devices through memory maps that are not built into the Linux kernel. For example, ADCs, DACs, sensors, and actuators can be controlled through the IIO framework. Analog Devices (AD), the original manufacturer and developer of the Pluto SDR, utilize IIO for directly interfacing every component on the Pluto SDR, except for the FPGA. If the Pluto SDR is to be programmed in software, IIO commands are how the different components are interfaced. This includes setting the bandwidth, carrier frequency, sampling rate, transmit/receiver gains, input/output filters, and extracting and placing I/Q samples on the buffers for receiving and transmitting, respectively. IIO itself can be controlled through linux commands, but further abstractions through open source software libraries allow the controlling of IIO through most programming languages, including C,

Python, Julia, and MATLAB. Through these libraries it is possible to do real-time embedded digital signal processing on the Pluto SDR. IIO also has the ability to pass commands through the kernel via Internet Protocol (IP) packets. This allows remote digital signal processing on an external computer through an Ethernet over USB connection to pass IP packets containing IIO data packets. The Pluto SDR structure with IIO is shown in Figure 9.1. Compared to the single core 500 MHz ARM processor on the Pluto SDR,

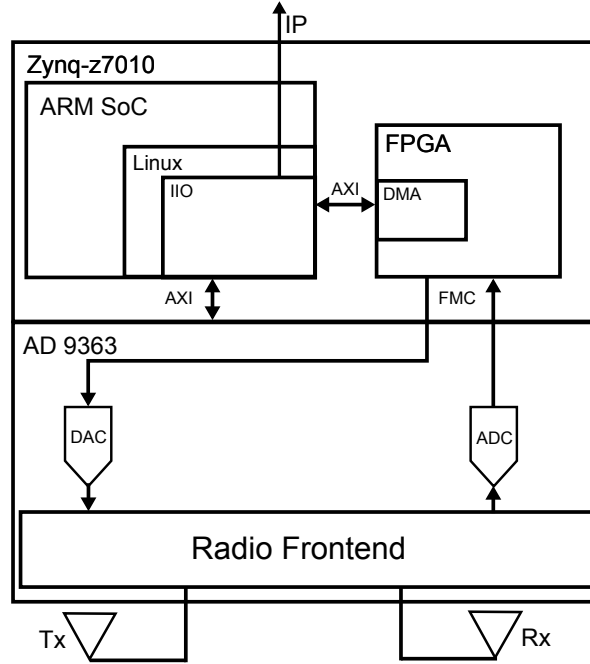


Figure 9.1: Pluto SDR structure with IIO

an external modern computer, with faster speeds, more processor cores, and architectural improves, should be able to process digital signals significantly faster than the Pluto SDR, but the network link between the Pluto SDR and external computer acts as a significant bottleneck. In general, USB 2.0 has a theoretical maximum bidirectional bitrate of 480 Mb/s. Each I and Q sample is 12 bits 2s-complement signed integer, for a total size of 24 bits per sample. Since memory is allocated in chunks of bytes, in theory 3 bytes would be sufficient for each time sample. Unfortunately the current IIO drivers for Pluto SDR packages each I and Q portion of the time sample separately, requiring 16 bits (2 bytes) for each 12 bit sample, where the 4 least-significant bits (LSB) are empty. The structure of each I or Q sample can be seen in Figure 9.2.

Considering that the transport layer is utilizing the Transmission Control Protocol

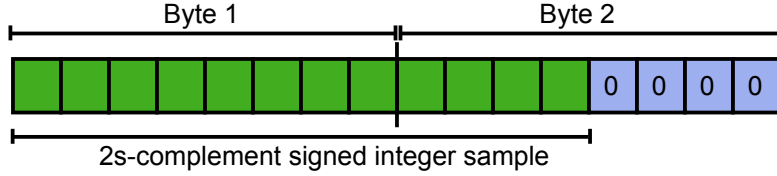


Figure 9.2: Pluto SDR structure with IIO

(TCP) to link the IP packets to the program on the computer, a typical 1500 Byte link with IPv4 address will need 20 bytes for the TCP and IPv4 packet header, respectively. This leaves 1460 Bytes for the sample data in each TCP/IP packet

$$\begin{aligned} \frac{\# \text{ Samples}}{\text{pkt}} &= \left\lfloor \frac{1460 \times 8}{16 + 16} \right\rfloor \\ &= 365 \text{ samples/pkt.} \end{aligned} \quad (9.1)$$

Given the maximum theoretical bitrate of USB 2.0, the maximum receiver sampling rate of a Pluto SDR to an external computer is

$$\begin{aligned} \max_{f_s} &= \frac{480 \times 10^6}{1500 \times 8} \times 365 \\ &= 14.6 \text{ MSPS.} \end{aligned} \quad (9.2)$$

The maximum sampling rate possible is only 14.6 MSPS over USB 2.0, despite the fact that the maximum sampling rate of AD 9393 is 61.44 MSPS, as seen in Table 9.2. Given the Nyquist rate requirement for no ISI (i.e., $B \leq f_s/2$), the maximum bandwidth that can be captured is only 7.8 MHz, compared to the maximum theoretical bandwidth of 30.72 MHz, that the Pluto SDR is capable of. When considering the protocols proposed in this thesis, a bandwidth $B \geq 15$ MHz is required to have frequency-selectivity. In practice, USB 2.0 rarely operates at its maximum speed and whatever the actual throughput is, it must be shared between all USB 2.0 devices connected to the computer on a per demand basis. To evaluate the maximum throughput of the Pluto SDR a packet exchange over TCP/IP IIO via USB 2.0 is performed to simulate receiving samples by transmitting 1.51 GiB from the Pluto SDR to a remote computer. The result can be seen in Figure 9.3

```
rr@rr-MS-7C75: $ iio_readdev -u ip:192.168.3.1 -b 100000 cf-ad9361-lpc | pv > /dev/null
[ 51GiB 0:01:14 [20.9MiB/s] [ <=> ]
```

Figure 9.3: The throughput measured through a TCP/IP packet exchange over IIO on a Pluto SDR and USB 2.0 on a Linux remote computer

The maximum throughput possible is only 20.9 MiB/s (175.32 Mb/s), which is much lower than the theoretical 480 Mb/s (57.22 MiB/s). The real maximum sampling frequency is

$$\begin{aligned} \max_{f_s} &= \frac{175.32 \times 10^6}{1500 \times 8} \times 365 \\ &= 5.33 \text{ MSPS}, \end{aligned} \tag{9.3}$$

which only corresponds to $B \leq 2.67$ MHz, given the Nyquist rate. This is a very small bandwidth and is much smaller than the bandwidth from the theoretical maximum Pluto SDR bandwidth of 30.72 MHz.

The version of IIO running on the Pluto SDR is v0.25. By default, IIO reserves a receiver and transmitter buffer structure on the Pluto SDR to utilize $B = 4$ buffers of arbitrary size Q samples on the main memory shared by the FPGA and SoC on the Zynq-z7010. The receiver and transmitter chain each have 4 buffers, respectively, which fills in a round-robin fashion, which can be seen in Figure 9.4. Buffers are processed in whole blocks of Q samples by an IIO call, which performs a memory copy from the reserved buffers, to another location in memory.

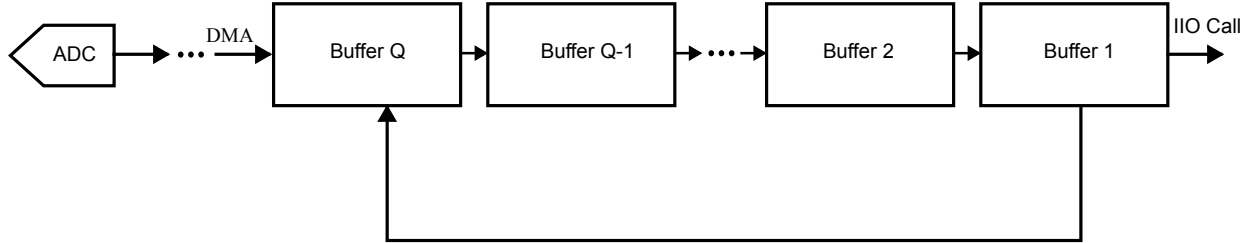


Figure 9.4: Example of the receiver buffer queue on the Pluto SDR

When receiving samples from the ADC, these buffers are filled automatically as one single large queue through DMA, but as mentioned only one buffer can be transferred to another location in memory at a time. When transmitting, the buffers are only filled through explicit command on the buffers, before being transferred through DMA to the DAC for transmission. Since the transmitter chain is controlled by the user, it is only the receiver chain (which is constantly sampling the wireless channel) that operates under a queue model. If the SDR cannot process incoming samples faster than the rate they are entering the buffer, then an unstable queue will form, which results in underflow errors (i.e., missed samples are dropped instead of overwriting in memory). Missing samples can

mean that the radio can completely or partially miss a symbol transmission from another device. The SDR structure as a C/C/1 queue can be seen in Figure 9.5.

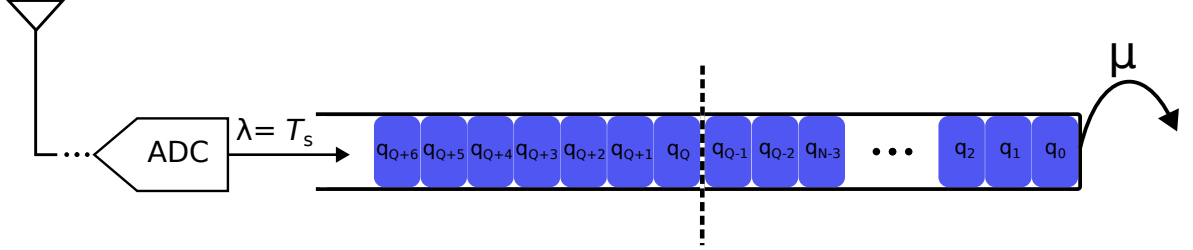


Figure 9.5: Example of the receiver buffer queue on the Pluto SDR [123]

The size of a buffer is Q samples, though there can be $B \geq 1$ buffers in series in the SDR, for a total queue size of BQ . The utilization of the buffer is $\rho = f_s/\mu$, where f_s is the sampling rate and μ is the rate samples are processed by the embedded processor on the SDR. If $\rho > 1 \rightarrow f_s > \mu$, then the queue is unstable and results in dropped samples. Little's rule gives the mean number of samples in the buffer $L_q = \rho BQ$. Since the buffer is capped at BQ , this can be used to get the mean percentage of dropped samples

$$\begin{aligned} \% \text{ Dropped} &= \frac{L_q - BQ}{BQ} \\ &= \rho - 1 \\ &= \frac{f_s}{\mu} - 1 \end{aligned} \tag{9.4}$$

$$\begin{aligned} &= \frac{f_s}{Q/(T_{proc} + T_c)} - 1 \\ &= \frac{f_s(T_{proc} + T_c)}{Q} - 1, \end{aligned} \tag{9.5}$$

where (9.5) comes from the approximation that $\mu = Q/(T_{proc} + T_c)$ is the time it takes to process the buffer (T_p) and the time to use the IIO call to copy the buffer (T_c). To ensure no samples are dropped, the f_s can be lowered, but this proportionally results in a lower bandwidth, which limits the portion of the channel that can be observed (i.e., high bandwidth from a fast interface). An alternate solution is to increase μ , by developing faster methods to process a buffer. It is possible to speed up the processing by increasing Q , but T_p has a monotonic relationship to Q , as any processing performed will have a running time $\Omega(Q)$.

Using a fast external computer with better performance than the Pluto SDR for signal processing is one way to massively increase μ . The main limitation is the low bandwidth from (9.3) and the processing limitation from the added time of T_c needed to encapsulate, transmit, receive, and decapsulate a TCP/IP packet over USB 2.0. Given the limitations in the sampling rate to the computer, as shown in (9.3) and the limited processing from T_c , the only alternative is to implement the protocols on the FPGA or exclusively on the Pluto SDR with an embedded IIO implementation. In this thesis, all the implementations are done embedded on the Pluto SDR utilizing the IIO library on C (libiio) [126].

9.2.2 Communication Setup

Since the Pluto SDR has an open source design, other companies have produced slightly modified version of the Pluto SDR with upgraded components to improve the performance and price. One version is called the Libre SDR. A comparison of the major hardware components of the Libre SDR compared to the Pluto SDR is shown in Table 9.3. The major downside of the Libre SDR is the lack of guarantee and warranty that comes with buying the Pluto SDR. Analog Devices is a reputable company that likely has an established manufacturer process to test and validate the quality of the products and its components, while the Libre SDR is an open source adaption made by anonymous developers.

Table 9.3: Major processing components in the Analog Devices ADALM-Pluto SDR

Component	Pluto SDR	Libre SDR
FPGA	28K LE / 17,600 LUTs	85K LE / 53,200 LUTs
Processor	1 × 500 MHz	2 × 1150 MHz
RAM	DDR3 512 MB@525 MHz	DDR3 1024 MB@750 MHz
Memory	32MB Flash	32MB Flash / microSD
Comp. Interface	USB 2.0	1 Gbit/s Ethernet
OS Driver	IIO v0.25	IIO v0.25

In general, the Libre SDR has twice the processing and memory capabilities of the Pluto SDR. The Libre SDR has the same radio frontend (AD 9363) as the Pluto SDR. With the Gigabit ethernet connection, the Libre SDR can have a faster throughput compared to USB 2.0 between an external computer, which can be seen in Figure 9.6.

```
rr@rr-MS-7C75:~$ iio readdev -u ip:192.168.70.60 -b 100000 cf-ad9361-lpc | pv > /dev/null
1007MiB 0:00:10 100MiB/s <=>
```

Figure 9.6: The throughput measured through a TCP/IP packet exchange over IIO on a Libre SDR and Gigabit Ethernet on a Linux remote computer

The throughput of 100 MiB/s (838.851 Mb/s) is $4.875\times$ than the Pluto SDR performance 20.9 MiB/s. This performance yields a maximum sampling rate

$$\begin{aligned} \max_{f_s} &= \frac{838.851 \times 10^6}{1500 \times 8} \times 365 \\ &= 25.5 \text{ MSPS}, \end{aligned} \tag{9.6}$$

One of the minor adjustments that provides this boost is firmware optimizations that utilize LVDS signalling on the Libre SDR, compared to the CMOS signalling on the Pluto SDR, for the exchange of data between the AD9363 and the FPGA. Furthermore, some optimization were performed to remove unnecessary data in the encapsulations [127].

To remotely control a Pluto SDR or Libre SDR a remote connection is established through Raspberry Pi 5 single board computers . Each Raspberry Pi 5 has 8 GB of RAM and is capable of being controlled using a Virtual Network Computing (VNC) using Raspberry Pi Connect over the Internet. The Pluto SDR can connect through USB 2.0 to the Raspberry Pi, and the Libre SDR can connect through the 1 Gbps ethernet connection. Since the SDR implementations are embedded, the Raspberry Pi acts purely as a controller. Since the Pluto SDR runs using an embedded Linux OS and the USB 2.0 or ethernet connection creates a local network with the Raspberry Pi, the SDRs are controlled through an secure shell (SSH) connection. With all both devices being powered by a portable battery and the Raspberry Pi able to connect to the Internet using WiFi or LTE, either SDR is highly mobile and controllable from any computer. A diagram of this setup can be seen in Figure 9.7.

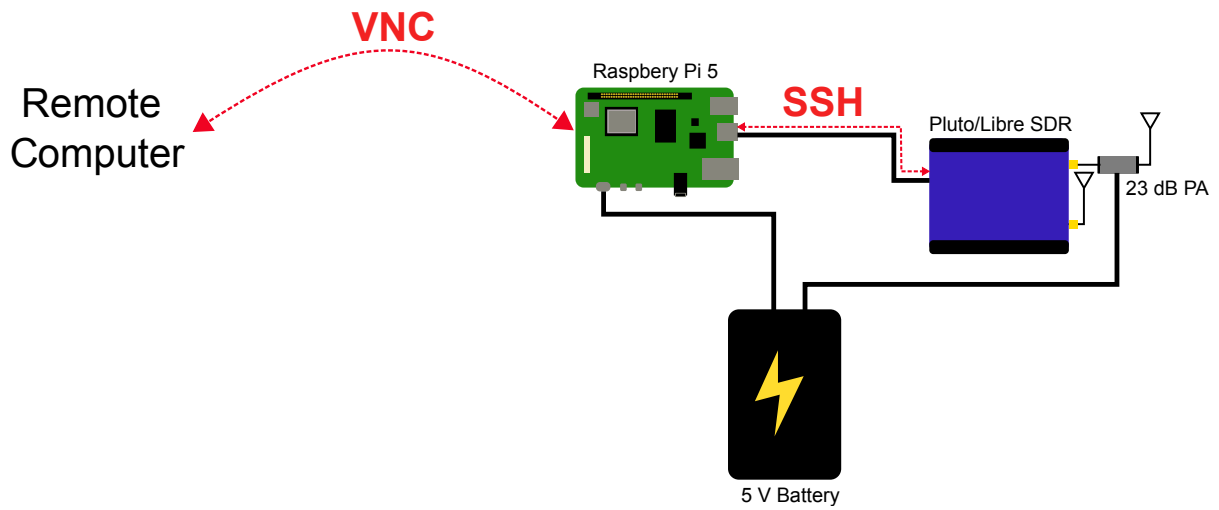


Figure 9.7: SDR setup to control and power remotely

A physical example of the setup with the Libre SDR can be seen in Figure 9.8

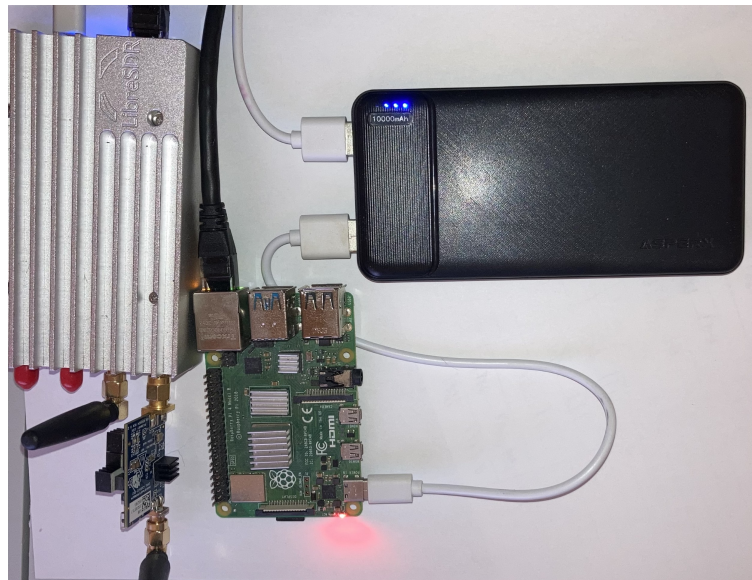


Figure 9.8: SDR setup with physical devices

To extend the distance of communications a 23 dB power amplifier must be used. To scale back the power to ensure regulator compliance and to manage the SNR, the internal

transmit gain of the Pluto or Libre SDR is adjusted through the IIO controls before any experimentation.

9.3 OFDM System

The communication protocol used throughout the hardware experiments in Chapter 5, Chapter 7, and Chapter 8 is an OFDM system. The system uses multiple steps in the signal processing chain to ensure proper demodulation of the pilots and data symbols, which is required to implement the relay detection protocols on the prover and verifier and the relay attacker. The same processes occur for the prover, verifier, and the different relays. The processes proposed are typical in most OFDM communication systems, so the general structure is not novel, but to make these processes function in real-time on an embedded SDR required novel changes, which are presented in subsequent sections. The transmitter structure for the OFDM system is shown in Figure 9.9

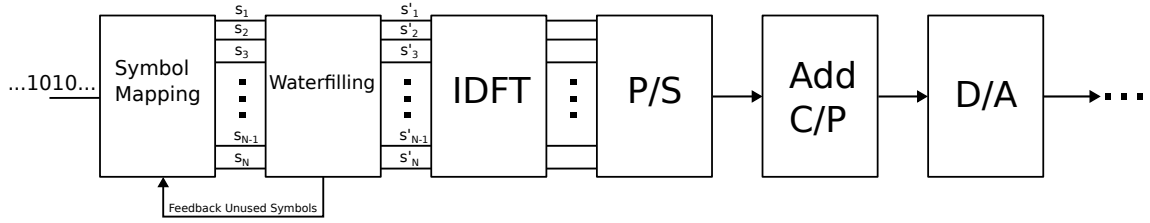


Figure 9.9: OFDM Transmitter Chain

This structure is similar to the OFDM transmitter structure in Figure 2.9, with the additional waterfilling process required to properly allocate power. On the other hand, the receiver structure is much more complex and is shown in Figure 9.10.

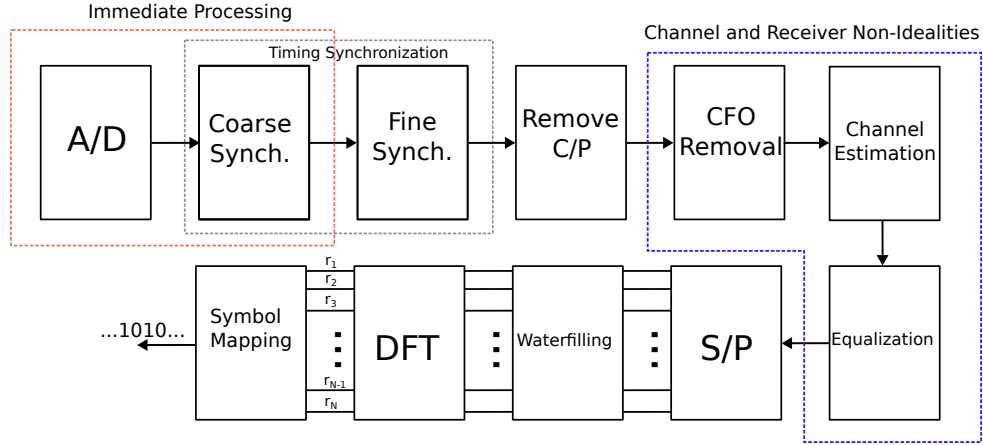


Figure 9.10: OFDM Receiver Chain

The receiver structure is more complex to process due to the synchronization and non-idealities that exist in a real transmission system, which is required to have reliable communications. Each of these processes are designed and programmed using IIO and the C programming language on the Pluto and Libre SDR for this thesis. As submission of this thesis, on the Internet this will be the only implementation for this SDR platform that does reliable OFDM transmissions that will be made publicly available.

Given the processing and timing limitations, the portion of the receiver structure that requires immediate processing (buffer transferring from the analog-to-digital converter (ADC) and coarse synchronization) are the bottlenecks for achieving fast sampling in (9.5). This will be discussed in the subsequent sections in this chapter.

For the OFDM system all transmissions have the same structure, shown in Figure 9.11.

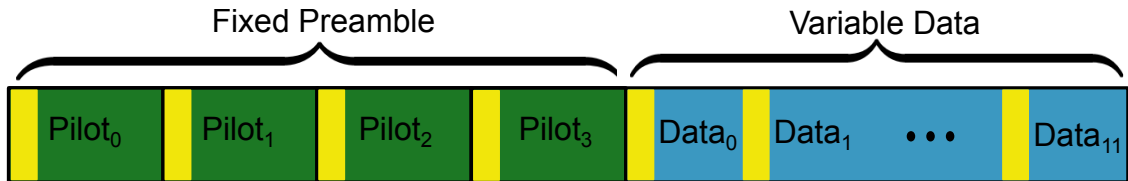


Figure 9.11: OFDM Transmission Structure

The transmissions always contain 4 identical pilots in the preamble, which are used for synchronization, CFO removal, channel estimation, and equalization. The data has

a variable length of 0-12 blocks, where each block represent an OFDM symbols. Each symbol is 128 samples, where 64 subcarriers are used for QPSK symbols (128-bits per symbol). This yields a maximum of 1536 bits per transmission. The pilots vary depending on the experiment, but they always occupy the same subcarriers as the OFDM symbols for equalization. The pilot is generated by a 64-bit sequence that is interpolated $2\times$ using a Sinc pulse-shape, yielding the overlapping 128 subcarriers with the OFDM symbol. Each pilot and data also has a cyclic pre-fix, represented by the yellow in Figure 9.11.

9.4 Pilot Selection: Construction of Frequency Distance Sequence Sets¹

As discussed in Section 9.2.1, when performing an OFDM symbol transmission with an embedded SDR, the processing delay must be limited during the receiving step of the system, to ensure that the no samples are missed from the transmission. From (9.5), this can be achieved by lowering the sampling frequency (rate) f_s , but this comes at a cost of a reduced bandwidth for transmissions. The two primary goals for pilot selection are:

1. Timing synchronization for the signal acquisition and identification of the starting sample,
2. For a given f_s , full bandwidth channel estimation.

Selecting the appropriate pilot is important for optimizing the timing synchronization, which is the long step, which adds time to (9.5).

For timing synchronization, the average power of a received signal can be a fast method for performing coarse time synchronization. If the received power is greater than some deviation from the noise power, then a pilot can be identified and the protocol can proceed. When compared to correlation based pilot synchronization, calculating the average power has a linear running time $O(N)$, compared to a Fourier transform detection with $O(N \ln(N))$. To linearly improve the processing time, the average power can be measured through a subsampling (decimation) of the received signal. Subsampling risks missing samples with high instantaneous powers, which can minimize the evaluated average power, resulting in a missed pilot within a larger buffer. A pilot with a large peak-to-average power ratio (PAPR) is more likely to be missed, especially if the decimation rate is high.

¹This section comes from a published paper by R. Abubaker and G. Gong [123]

In contrast, lowering the PAPR reduces the chance that a pilot will be missed. This yields a sub-requirement, that for signal acquisition, the pilot needs to have a low PAPR. After coarse synchronization, the received symbol can be saved for fine time synchronization. Fine timing synchronization requires good autocorrelation, with a perfect pilot having a zero aperiodic autocorrelation (ZAAC). Non-ZAC can result in timing errors, especially in a noisy or fading environment. Selecting a preferred sequence to generate a pilot symbol that has a low PAPR for decimated average power detection, ZAC or near ZAC for fine symbol synchronization, and a bandwidth that covers the whole sampling rate is required to design practical physical-layer security based protocols.

In this section, a novel process for generating perfect sequences to use as pilot is proposed, which is a generalization of the Milewski construction method [129]. The proposed pilots are used in the preamble of all the OFDM symbols transmitted in the hardware implementation (Figure 9.11), for timing synchronization and channel estimation, resulting in a block-type arrangement. In general, pilot selection is a relatively well explored field, as practical OFDM systems like WiFi and 4G/5G have needed to perform equalization and synchronization to combat frequency-selective fading. This subsection gives a proposal of a novel method that should match the performance of established sequences, and allow for the generation of new Golay complementary pairs (GCP) and new perfect sequences, that may not have been previously been known. The rest of the section details the goals for pilot generation, the new generation method, and evaluation of the performance of the generated pilot sequences.

9.4.1 Pilot Sequence Properties

When designing pilots sequences, there are certain properties that are ideal for estimating and utilizing the channel, as well as for timing synchronization. For a sequence $\mathbf{a} = (a[0], a[1], \dots, a[N-1])$ the properties of interest are:

1. Zero Aperiodic Autocorrelation (ZAAC), i.e., perfect sequences, where the Aperiodic Autocorrelation (AAC) is defined as:

$$\begin{aligned} AC_{\mathbf{a}}[\tau] &= \sum_n a[n]a[n+\tau]^* \\ &= 0, \text{ when } \tau \neq 0 \end{aligned} \tag{9.7}$$

Pilots selected with a ZAAC allow for effective coarse synchronization and fine synchronization of transmitted symbols, either on their own, or used as a preamble for

transmitted data symbols. It is important to note that the AAC differs from the autocorrelation, the same way the convolution differs from the cyclic convolution. It is assumed in the AAC that samples outside of the sample range are 0 and do not contribute to the correlation. In practice, the AAC will be applied as an aperiodic cross-correlation, as a signal is being searched for within a longer sequence for pilot timing synchronization.

2. Low Peak-to-Average Power ratio (PAPR): A sequence $a[n]$ has a PAPR defined by,

$$PAPR(\mathbf{a}) = \frac{\max_{0 \leq n \leq N-1} |a[n]|^2}{1/N \sum_{n=0}^{N-1} |a[n]|^2} \quad (9.8)$$

For a given average input power, a low PAPR is generally desirable for any transmitted signal because it allows for the use of efficient power amplifiers, without causing significant distortion from the large input peaks. Other considerations can include poor gain control performance. Since pilots are selected a priori, they should ideally be chosen to have a low PAPR. A low PAPR also improves coarse synchronization when utilizing the decimated average power.

3. Low Frequency Spacing: A frequency m of sequence $\mathbf{a}$ with corresponding DFT $\hat{\mathbf{a}} = F\{\mathbf{a}\}$ has a frequency spacing defined as

$$FS_{\mathbf{a}}[m] = \min_{d>0, d \in \mathbb{Z}} d, \text{ s.t. } |\hat{a}[m + d \bmod N]| \neq 0. \quad (9.9)$$

The frequency spacing can be interpreted as the distance between frequency k and the next non-zero powered frequency larger than it. Low frequency spacing ensures that the samples in the frequency domain are not too spread apart. If they are spread beyond the coherence bandwidth of the communication channel, the pilot signal will not be able to accurately estimate the coherence bandwidth. To ensure low frequency spacing both N and the sampling rate should be large and the frequency spacing should be equidistant and small as defined by

$$FS_{\mathbf{a}}[m] = q \ll N, \forall m.$$

When this occurs, the nonzero spectral points sample the full bandwidth of the channel, for a given sampling rate. This ensures that the pilot sequence can capture the coherence bandwidth of the channel, which is important for deriving randomness in physical-layer security. Let S be a set consisting of L sequences with equal frequency spacing. We define this frequency spacing as the parameter $q_{subchannel}$. An interleaving process that is a mapping of all the sequences in S to a sequence, where this

sequence has a resulting frequency spacing referred to as $q_{fullchannel}$. Ideally, the frequency spacing with $q_{subchannel} > 1$ will be desired for getting uncorrelated samples for physical-layer security. At the same time, all the nonzero frequencies should cover the entire bandwidth, i.e., $q_{fullchannel} = 1$, which means every frequency is getting sampled.

Due to the Fourier transform identity for $F\{\delta\}$ (where δ is the delta sequence), the time-frequency duality, and Wiener-Khinchin Theorem, finding sequences that are wide in frequency and time is non-trivial and generally contradictory. Below, a generalized Milewski construction to generate pilot sequences is introduced, which can provide sequences that satisfies all of the above properties.

Generalized Milewski construction

Considering the frequency domain requirement, if N is large, then the frequency spacing should not be large, which can prevent multiple sample values from being within the coherence bandwidth. Assuming an equidistant frequency spacing, the numbers of samples in the frequency domain needs to be large enough for randomness considerations as well (i.e., span over more than one coherence bandwidth), since consecutive samples may be correlated. A large total bandwidth is also needed to guarantee independence in order to provide randomness to be utilized in physical-layer security uses, like in Protocol 1.

We will take the approach starting from frequency domain. Given $\ell, q \in \mathbb{N}$, and

$$\hat{\mathbf{a}} = (\hat{a}[0], \hat{a}[1], \dots, \hat{a}[\ell - 1]) \in C^\ell,$$

we define a $q \times (\ell q)$ array, denoted as $\hat{A}$ by adding $q - 1$ zeros between any two neighboring elements in $\hat{a}[m]$. In detail, let $\hat{r}_i[m]$ be the i th row vector of $\hat{A}$

$$\begin{aligned} \hat{\mathbf{r}}_0 &:= (\hat{a}[0] \quad 0 \quad \overset{q-1}{0 \ 0 \ \dots \ 0} \quad \dots \ 0 \quad \dots \ \hat{a}[\ell-1] \quad 0 \quad \overset{q-1}{0 \ 0 \ \dots \ 0} \quad \dots \ 0) \\ \hat{\mathbf{r}}_1 &:= (0 \quad \hat{a}[0] \quad 0 \ 0 \ \dots \ 0 \quad \dots \ 0 \quad \hat{a}[\ell-1] \quad 0 \ 0 \ \dots \ 0) \\ &\vdots \\ \hat{\mathbf{r}}_{q-1} &:= (0 \quad 0 \quad 0 \ \dots \ 0 \quad \hat{a}[0] \quad \dots \ 0 \quad 0 \quad 0 \ \dots \ \hat{a}[\ell-1]). \end{aligned} \tag{9.10}$$

In other words, define

$$\hat{A} := \begin{pmatrix} \hat{\mathbf{r}}_0 \\ \hat{\mathbf{r}}_1 \\ \vdots \\ \hat{\mathbf{r}}_{q-1} \end{pmatrix} = (\hat{r}_i[m])_{q \times (\ell q)}, \tag{9.11}$$

where the i^{th} row corresponds to the vector $\hat{\mathbf{r}}_i$.

Property 1.

$$\hat{\mathbf{r}}_i = R^i(\hat{\mathbf{r}}_0), 0 < i < q,$$

where R is the right-shift operator.

In this way, each row vector has a fixed frequency spacing, which can be combined all the row vectors together, where the nonzero frequency response covers the entire bandwidth. To understand what this looks like in the time domain for the frequency response sequences.

Let $\mathbf{a}$ be obtained by applying the IDFT to $\hat{\mathbf{a}}$, i.e., $\mathbf{a} = F^{-1}\{\hat{\mathbf{a}}\}$, and $\mathbf{r}_i = F^{-1}\{\hat{\mathbf{r}}_i\}$ $0 \leq i < q$. Let

$$A := \begin{pmatrix} \mathbf{r}_0 \\ \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_{q-1} \end{pmatrix} = (r_{i,n})_{q \times (\ell q)}.$$

Define this as the IDFT of the matrix $\hat{A}$. The sequence set $\Sigma = \{\mathbf{r}_i | 0 \leq i < q\}$ an (ℓ, q) -frequency distance sequence set (FDSS). The sequence $\mathbf{a}$ is called a *seed sequence* of Σ in the time domain, and $\hat{\mathbf{a}}$ a *seed sequence in the frequency domain*.

9.4.2 Representation and correlation of frequency distance sequence sets

The results of the following Lemmas 1 and 2 are known [130]. For completeness, they are shown as they are important properties for the generalized Milewski construction.

Lemma 1. *With the above notation, the elements in $\mathbf{r}_i = (r_i[0], r_i[1], \dots, r_i[\ell q - 1])$, $0 \leq i < q$ are given by*

$$\begin{aligned} r_0 &= a[n] \\ r_i[n] &= r_0[n] \omega_{\ell q}^{in} = a[n] \omega_{\ell q}^{in}, 0 \leq n < \ell q, 0 \leq i < q \end{aligned}$$

where $\omega_N = e^{\frac{j2\pi}{N}}$ is the N^{th} root of unity and n is reduced by modulo ℓ in $a[n]$, since the sequence has a period (length) ℓ .

Proof. For $i = 0$, thus

$$\begin{aligned}
r_0[n] &= F^{-1}\{\hat{\mathbf{r}}_0\} = \frac{1}{\ell q} \sum_{m=0}^{\ell q-1} \hat{r}_0[m] \omega_{\ell q}^{mn} \\
&= \frac{1}{\ell} \sum_{j=0}^{\ell-1} \hat{a}[j] \omega_{\ell q}^{jnq} = \frac{1}{\ell} \sum_{j=0}^{\ell-1} \hat{a}[j] \omega_{\ell}^{jn} \\
&= a[n], 0 \leq n < \ell q.
\end{aligned}$$

For $i > 0$, from Property 1,

$$\mathbf{r}_i = F^{-1}\{\hat{\mathbf{r}}_i\} = F^{-1}\{R^i(\hat{\mathbf{r}}_0)\}.$$

Together with the time-frequency relation,

$$r_i[n] = r_0[n] \omega_{\ell q}^{in}, 0 \leq n < \ell q.$$

Applying the case $i = 0$, consequently,

$$r_i[n] = a[n] \omega_{\ell q}^{in}, 0 \leq n < \ell q.$$

□

From Theorem 1, the entries in A are given by

$$A = \begin{pmatrix} a[0] & \cdots & a[\ell-1] & \cdots & a[0] & \cdots & a[\ell-1] \\ a[0] & \cdots & a[\ell-1] \omega_{\ell q}^{\ell-1} & \cdots & a[0] \omega_{\ell q}^{\ell q-\ell} & \cdots & a[\ell-1] \omega_{\ell q}^{\ell q-1} \\ \vdots & & & & & & \\ a[0] & \cdots & a[L-1] \omega_{\ell q}^{(q-1)(\ell-1)} & \cdots & a[0] \omega_{\ell q}^{(q-1)(\ell q-\ell)} & \cdots & a[\ell-1] \omega_{\ell q}^{(q-1)(\ell q-1)} \end{pmatrix}$$

Example 1. Let $\ell = 4$, $q = 3$, $a[n] = (1, 1, 1, -1)$, a perfect seed sequence, then

$$A = \begin{pmatrix} 1 & 1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & -1 \\ 1 & \omega_{12} & \omega_{12}^2 & -\omega_{12}^3 & \omega_{12}^4 & \omega_{12}^5 & \omega_{12}^6 & -\omega_{12}^7 & \omega_{12}^8 & \omega_{12}^9 & \omega_{12}^{10} & -\omega_{12}^{11} \\ 1 & \omega_{12}^2 & \omega_{12}^4 & -\omega_{12}^6 & \omega_{12}^8 & \omega_{12}^{10} & \omega_{12}^{12} & -\omega_{12}^2 & \omega_{12}^4 & \omega_{12}^6 & \omega_{12}^8 & -\omega_{12}^{10} \end{pmatrix}$$

and

$$\hat{A} = \begin{pmatrix} \hat{a}[0] & 0 & 0 & \hat{a}[1] & 0 & 0 & \hat{a}[2] & 0 & 0 & \hat{a}[3] & 0 & 0 \\ 0 & \hat{a}[0] & 0 & 0 & \hat{a}[1] & 0 & 0 & \hat{a}[2] & 0 & 0 & \hat{a}[3] & 0 \\ 0 & 0 & \hat{a}[0] & 0 & 0 & \hat{a}[1] & 0 & 0 & \hat{a}[2] & 0 & 0 & \hat{a}[3] \end{pmatrix}.$$

Since $\hat{a}[m] = 2(1, j, 1, -j)$ where $j = \omega_4$:

$$\hat{A} = \left(\begin{array}{ccc|ccc|ccc} 2 & 0 & 0 & 2j & 0 & 0 & 2 & 0 & 0 \\ 0 & 2 & 0 & 0 & 2j & 0 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 & 0 & 2j & 0 & 0 & 2 \end{array} \middle| \begin{array}{ccc} -2j & 0 & 0 \\ 0 & -2j & 0 \\ 0 & 0 & -j \end{array} \right).$$

Lemma 2. Let $\mathbf{u} = (u[0], \dots, u[N-1])$ and $\mathbf{v} = (v[0], \dots, v[N-1])$ be two vectors in $\mathbb{C}^N$ and their respective DFTs be $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$. Then

$$F^{-1}\{D\{\hat{\mathbf{u}}\}\hat{\mathbf{v}}^*\}[n] = C_{\mathbf{u},\mathbf{v}}[N-n]$$

If $\mathbf{u} = \mathbf{v}$,

$$F^{-1}[D\{\hat{\mathbf{u}}\}\hat{\mathbf{u}}^*][n] = C_{\mathbf{u}}[N-n].$$

Proof.

$$\begin{aligned} \hat{u}[m] &= \sum_{n=0}^{N-1} u[n] \omega_N^{-nm} \text{ and } \hat{v}[n] = \sum_{n=0}^{N-1} v[n] \omega_N^{-nm} \\ \Rightarrow \\ \hat{u}[m] \cdot \hat{v}[m]^* &= \sum_{0 \leq n, n' < N} u[n] v[n']^* \omega_N^{-k(n-n')} \\ &= \sum_{\tau=0}^{N-1} \sum_{t=0}^{N-1} u[n] v[n-\tau]^* \omega_N^{-m\tau}, \quad (\tau = n - n') \\ &= \sum_{\tau=0}^{N-1} C_{\mathbf{u},\mathbf{v}}[-\tau] \omega_N^{-m\tau}. \end{aligned}$$

□

Theorem 1. The FDSS Σ is mutually orthogonal, i.e., any pair of the sequences in Σ is uncorrelated:

$$C_{\mathbf{r}_i, \mathbf{r}_j}[\tau] = 0, \quad \forall i \neq j, 0 \leq i, j < q, \forall \tau.$$

Proof. According to the definition of $\hat{\mathbf{r}}_i$, we have

$$\hat{\mathbf{r}}_i \cdot \hat{\mathbf{r}}_j^* = \overbrace{0 \cdots 0}^{\ell q} \quad \forall i \neq j.$$

Applying Lemma 2, the cross-correlation function of $\mathbf{r}_i$ and $\mathbf{r}_j$ is a zero function.

□

Remark 1. Note that this property is independent of the autocorrelation of the seed sequence $\mathbf{a}$. It only depends on the structure. In other words, they are mutually orthogonal in the frequency domain, so too in the time domain.

Lemma 3. *The autocorrelation functions of $\mathbf{r}_i$ s are given below*

$$C_{\mathbf{r}_0}[\tau] = qC_{\mathbf{a}}[\tau] \quad (9.12)$$

and

$$C_{\mathbf{r}_i}[\tau] = C_{\mathbf{r}_0}[\tau]\omega^{-i\tau} = qC_{\mathbf{a}}[\tau]\omega^{-i\tau}, 0 \leq i < \ell q. \quad (9.13)$$

Proof.

$$C_{\mathbf{r}_0}[\tau] = \sum_{j=0}^{\ell q-1} r_0[n]r_0[n+\tau]^* = \sum_{j=0}^{\ell q-1} a[n]a[n+\tau]^* = q \sum_{j=0}^{\ell-1} a[n]a[n+\tau]^* = qC_{\mathbf{a}}[\tau].$$

For $i > 0$, we have

$$C_{\mathbf{r}_i}[\tau] = \sum_{n=0}^{\ell q-1} a[n]\omega_{\ell q}^{in}a[n+\tau]^*\omega_{\ell q}^{-i(n+\tau)} = \sum_{n=0}^{\ell q-1} a[n]a[n+\tau]^*\omega_{\ell q}^{-in} = qC_{\mathbf{a}}[\tau]\omega_{\ell q}^{-in}.$$

□

Theorem 2. *The frequency distance set $\Sigma = \{\mathbf{r}_0, \dots, \mathbf{r}_{q-1}\}$ is a periodic complementary sequence set (CSS) if and only if $\mathbf{a}$ is perfect.*

Proof. For $\tau \neq 0$, using Lemma 3, we have

$$\sigma := \sum_{i=0}^{q-1} C_{\mathbf{r}_i}[\tau] = qC_{\mathbf{a}}[\tau] \sum_{i=0}^{q-1} \omega_{\ell q}^{-i\tau}. \quad (9.14)$$

Case 1. $\tau \equiv 0 \pmod{\ell}$. In this case, we have $\tau = e\ell, 0 < e < q$. Then

$$\sigma = q\|\mathbf{a}\|^2 \sum_{i=0}^{q-1} \omega_q^{-ie} = 0 \quad (\text{because } e \not\equiv 0 \pmod{q})$$

where $\|\mathbf{a}\| = \sqrt{|a[0]|^2 + |a[1]|^2 + \dots + |a[\ell-1]|^2}$.

Case 2. $\tau \not\equiv 0 \pmod{\ell}$. We can $\tau = e\ell + d, 0 < e < q, 0 < d < \ell$. Note that $\sum_{i=0}^{q-1} \omega_{\ell q}^{-i\tau} \neq 0$, where $\sum_{i=0}^{q-1} \omega_{\ell q}^{id} = 0$ if and only if $\ell \mid d$ (Euler's identity of the primitive root of unity). Since $C_{\mathbf{a}}[\tau] = C_{\mathbf{a}}[d] = 0$, then $\sigma = 0$ if and only if $\mathbf{a}$ is perfect.

The frequency distance set Σ is a CSS if and only if $\mathbf{a}$ is perfect.

□

New perfect sequences

Let $(PA)^T$ be the transpose of PA where P is a $q \times q$ permutation matrix. Define an interleaved sequence, denoted as $\mathbf{u} = \{u[n]\}_{n \geq 0}$ with period ℓq^2 from $(PA)^T$ whose elements are given by

$$u[j\ell q + i] := a[j \bmod \ell] \omega_{\ell q}^{\pi(i)j}, 0 \leq i < q, 0 \leq j < \ell q \quad (9.15)$$

where π is a permutation of $\mathbb{Z}_q$ from the permutation matrix P . In other words, $\mathbf{u}$ is obtained from $(PA)^T$ by concatenating each row of the matrix into a single sequence. The sequence $\mathbf{u}$ is referred to as an (ℓ, q) -FDSS interleaved sequence or a frequency distance sequence for short.

Example 2. Using the A from Example 1 and with $P = I$, the resulting matrix is

$$A^T = \begin{pmatrix} a[0] & a[0] & a[0] \\ a[1] & a[1]\omega_{12} & a[1]\omega_{12}^2 \\ a[2] & a[2]\omega_{12}^2 & a[2]\omega_{12}^4 \\ a[3] & a[3]\omega_{12}^3 & a[3]\omega_{12}^6 \\ a[0] & a[0]\omega_{12}^4 & a[0]\omega_{12}^8 \\ a[1] & a[1]\omega_{12}^5 & a[1]\omega_{12}^{10} \\ a[2] & a[2]\omega_{12}^6 & a[2] \\ a[3] & a[3]\omega_{12}^7 & a[3]\omega_{12}^2 \\ a[0] & a[0]\omega_{12}^8 & a[0]\omega_{12}^4 \\ a[1] & a[1]\omega_{12}^9 & a[1]\omega_{12}^6 \\ a[2] & a[2]\omega_{12}^{10} & a[2]\omega_{12}^8 \\ a[3] & a[3]\omega_{12}^{11} & a[3]\omega_{12}^{10} \end{pmatrix}$$

and the corresponding interleaved sequence is a $(4,3)$ -FDSS sequence $\mathbf{u}$ and is obtained by concatenating the matrix row-by-row, which has ZAC.

$$\begin{aligned} \mathbf{u} &= (\overbrace{a[0], a[0], a[0]}^{\text{row 1}}, \overbrace{a[1], a[1]\omega_{12}, a[1]\omega_{12}^2}^{\text{row 2}}, \dots, \overbrace{a[2], a[2]\omega_{12}^2, a[2]\omega_{12}^{10}}^{\text{row 11}}, \overbrace{a[3], a[3]\omega_{12}^{11}, a[3]\omega_{12}^{10}}^{\text{row 12}}) \\ &= (\overbrace{1, 1, 1}^{\text{row 1}}, \overbrace{\omega_{12}, \omega_{12}^2}^{\text{row 2}}, \dots, \overbrace{1, \omega_{12}^2, \omega_{12}^{10}}^{\text{row 11}}, \overbrace{-1, -\omega_{12}^{11}, -\omega_{12}^{10}}^{\text{row 12}}) \end{aligned}$$

Theorem 3. With the above notation, the following assertions are true.

1. The autocorrelation of $\mathbf{u}$ is given by

$$C_{\mathbf{u}}[\tau] = \begin{cases} 0, & \tau \not\equiv 0 \pmod{q}, \\ 0, & \tau = dq, d = n\ell, 0 < t < q \\ qC_{\mathbf{a}}[j]\delta, & \tau = dq, 0 < d < \ell q, d = n\ell + j, \\ & 0 < j < \ell, 0 \leq n < q \end{cases}$$

where

$$\delta = \sum_{i=0}^{q-1} \omega_{\ell q}^{-id}.$$

2. $\mathbf{u}$ is perfect if and only if $\mathbf{a}$ is perfect.

Proof. For simplicity, the following proof is completed using the identity matrix for permutation ($P = I$). The proof is similar when P is a general $q \times q$ permutation matrix.

Case 1. 1) $\tau \not\equiv 0 \pmod{q} \implies \tau = dq + e, 0 < e < q, 0 \leq d < \ell q$. Since this is an interleaved sequence, the relationship of (e, d) in the computation of autocorrelation of $\mathbf{u}$ is shown below.

$$A^T = \begin{pmatrix} r_{0,0} & \cdots & r_{0,e} & \cdots & r_{0,q-1} \\ \vdots & & & & \\ r_{d,0} & \cdots & r_{d,e} & \cdots & r_{d,q-1} \\ \vdots & & & & \\ r_{\ell q-1,0} & \cdots & r_{\ell q-1,e} & \cdots & r_{\ell q-1,q-1} \end{pmatrix}$$

Thus, the autocorrelation of $\mathbf{u}$ is transferred to the sum of the cross-correlation of $\mathbf{r}_i$ and $\mathbf{r}_{i+e}$, as detailed below.

$$C_{\mathbf{u}}[\tau] = \sum_{n=0}^{\ell q^2-1} u[n]u[n+\tau]^* = \sum_{i=0}^{q-1} C_{\mathbf{r}_i, \mathbf{r}_{i+e}}[d].$$

From Lemma 1, $C_{\mathbf{u}}[\tau] = 0$.

Case 2. $\tau = dq, 0 < d < \ell q$. We have

$$C_{\mathbf{u}}[\tau] = \sum_{i=0}^{q-1} C_{\mathbf{r}_i}[d] = qC_{\mathbf{a}}[d] \sum_{i=0}^{q-1} \omega_{\ell q}^{-id},$$

where the last identify comes from Lemma 2.

2) $d = n\ell, 0 < n < q$. In this case, $C_{\mathbf{a}}[d] = \|\mathbf{a}\|^2$. Consequently

$$C_{\mathbf{u}}[\tau] = q\|\mathbf{a}\|^2 \sum_{i=0}^{q-1} \omega_{\ell q}^{-in\ell} = q\|\mathbf{a}\|^2 \sum_{i=0}^{q-1} \omega_q^{-in} = 0,$$

where the last identity comes from Property 1, because $n \not\equiv 0 \pmod{\ell}$. Now, the second case has been established in the first assertion.

3) $d = n\ell + j, 0 \leq n < q, 0 < j < \ell$. In this case,

$$C_{\mathbf{a}}[d] = C_{\mathbf{a}}[j].$$

So

$$C_{\mathbf{u}}(\tau) = qC_{\mathbf{a}}[j] \sum_{i=0}^{q-1} \omega_{\ell q}^{-id}. \quad (9.16)$$

The proof is complete for the first assertion.

Note that

$$\omega_{\ell q}^{-id} \neq 0 \text{ if } d \not\equiv 0 \pmod{\ell}.$$

Otherwise, $\ell \mid d$ which is a contradiction. Hence, for those $\tau = (t\ell + j)q, 0 \leq t < q, 0 < j < \ell$, together with (9.16), we have

$$C_{\mathbf{u}}[\tau] = 0 \Leftrightarrow C_{\mathbf{a}}[j] = 0.$$

Thus, the second assertion is established. $\square$

Summarizing the properties of the proposed $(\ell, q, \mathbf{a})$ -FDSS in Table 9.4.

Remark 2. In Milewski's original construction [129], he considers the case where $P = I$ (identity matrix), required $q = \ell^k$, and only briefly outlined that if $\mathbf{a}$ is perfect, then $\mathbf{u}$ is perfect, not the reverse part. In later work by Mow [131], he established the results shown in Table 9.4 with the condition that P is the identity, ℓ is a square free number, and $\mathbf{a}$ is a Frank-Zadoff-Chu sequence, is a sufficient condition, but did not establish any required conditions.

Table 9.4: Correlation properties of (ℓ, q) -FDSS

(ℓ, q) -FDSS for any positive integers ℓ, q and the seed $\hat{\mathbf{a}} = \{\hat{a}[m]\}_{m=0}^{\ell-1}$	
$\hat{A}_{q \times \ell q} \xleftrightarrow{DFT} A_{q \times \ell q}$	FDSS, the set consisting of the row vectors of A is mutual orthogonal.
	The FDSS is a CSS of size q if and only if the seed sequence $\mathbf{a}$ is perfect.
$\mathbf{u} = \{u[t]\}_{t=0}^{\ell q^2-1}$	Interleaved from the transpose of PA where P is a $q \times q$ permutation matrix, i.e., $(PA)^T$ is a perfect sequence if and only if the seed sequence $\mathbf{a}$ is perfect.

More compact representation of A

This subsection presents a compact representation of A from the frequency distance set in terms of the Kronecker delta product of vectors and matrix.

Let $W_{\ell q, \ell} = (1, \omega_{\ell q}, \dots, \omega_{\ell q}^{\ell-1})$, we define a power of M as a term-wise power.

$$W_{\ell q, \ell}^i := (1, \omega_{\ell q}^i, \dots, \omega_{\ell q}^{i(\ell-1)}), 0 \leq i < q$$

where $W_{\ell q}^0 = (1, \dots, 1)$. Note that

$$D\{\mathbf{a}\}W_{\ell q, \ell} = (a_0, a_1\omega_{\ell q}, \dots, a_{\ell-1}\omega_{\ell q}^{\ell-1}).$$

With the following DFT matrix of order q .

$$DFT_q = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_q & \omega_q^2 & \dots & \omega_q^{q-1} \\ \vdots & & & & \\ 1 & \omega_q^{q-1} & \omega_q^{2(q-1)} & \dots & \omega_q^{(q-1)^2} \end{pmatrix} = \begin{pmatrix} W_q^0 \\ W_q \\ \vdots \\ W_q^{q-1} \end{pmatrix}$$

where $W_q = W_{q, q}$.

Property 2. *With the above notation, we have*

$$\mathbf{r}_i = W_q^i \otimes (D\{\mathbf{a}\}W_{\ell q, \ell}^i), 0 \leq i < q,$$

where $\otimes$ is the Kroeneker product.

9.4.3 Pilot detection sequence application

To show the practicality of the sequence generation from the generalized Milewski construction, this subsection shows that when signals with good PAPR are selected as seed sequences, they can provide us near ZAAC signals. Three example seed sequences are observed: Frank-Zadoff-Chu (FZC) sequences [154, 153]), Binary-Phase Shift Keying Golay Complementary Pairs (BPSK-GCP), and Quaternary-Phase Shift Keying Golay Complementary Pairs (QPSK-GCP)[155]. These seed sequences do not have ZAAC on their own, but are selected as seeds, as they have a near ZAAC on the FDSS.

For the pilots a (4,4)-FDSS interleaved sequence is generated from each seed. This size was selected so the interleaved sequence is not large when pulse-shaped, otherwise the transmission time would be too long, defeating the purpose of desiring short RTT for pilots. The seed sequences can be observed in Table 9.5.

Table 9.5: Seed sequences

Seed	$n = 0$	$n = 1$	$n = 2$	$n = 3$
FZC	1	$1/\sqrt{2} + 1/\sqrt{2}j$	-1	$1/\sqrt{2} + 1/\sqrt{2}j$
BPSK-GCP	$-j$	j	j	j
QPSK-GCP	-1	$-j$	$-j$	-1

A naive pilot sequence is included, which contains a repetition of each QPSK symbol as reference to compare our sequences against. The pilot is interpolated with rate = 8, so the sequence has the same length (256) as the Milewski constructions. The sequence was generated by repeating the seed sequence, modulo 4, up to 32 subcarrier, such that interpolation matches the signal length of the FDSS interleaved sequences. The QPSK symbols are given by

$$\hat{a} = (1 + j, 1 - j, -1 + j, -1 - j). \quad (9.17)$$

This sequence was also selected due to it's full bandwidth coverage ($C = 1$) and simple implementation with OFDM.

From Table 9.5 it is clear that the PAPR of each sequence must be 1, as the PAPR of the seeds are all 1, and the root of unity from the Milewski construction has a magnitude of 1. In practice, the transmitted sequence is not the direct interleaved sequence, but a pulse-shaped (interpolated) sequence, which can have a different PAPR, depending on the choice of filter. The PAPR of the interpolated (rate = 4) sequence, including the PAPR of the in-phase and quadrature components, can be seen in Table 9.6.

Table 9.6: PAPR of the interpolated sequences

Seed	PAPR	I-PAPR	Q-PAPR
FZC	2.544	3.171	4.197
BPSK-GCP	4.244	4.209	6.172
QPSK-GCP	4.348	5.094	4.184
QPSK-OFDM	8.258	8.533	10.894

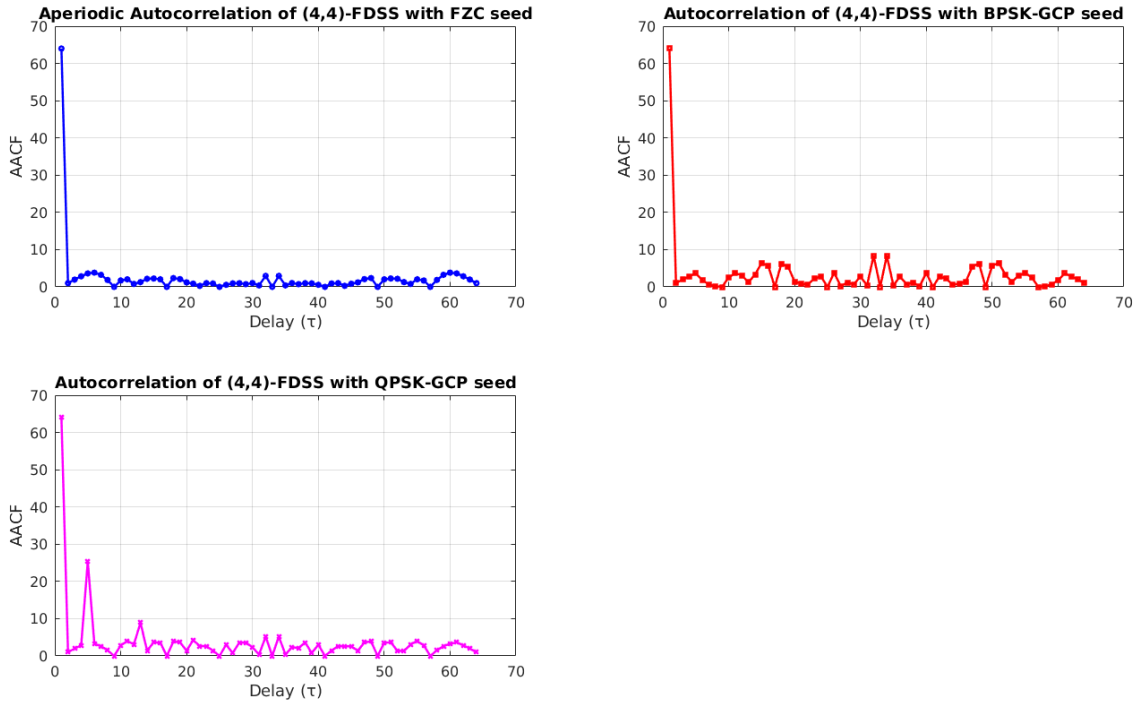


Figure 9.12: Aperiodic Autocorrelation Functions of the FDSS interleaved sequence with a Milewski Construction with different seeds (1) FZC (2) BPSK-GCP (3) QPSK-GCP [123]

It can be seen that the PAPR universally increases when pulse-shaping occurs, but the PAPRs are still much lower than a naive OFDM pilot, which has identical frequency properties as the sequences. Note that the PAPR on the sequences are not equal to the PAPR of those interpolated sequences.

The resulting AACF of the FDSS interleaved sequence $\mathbf{u}$ can be seen in Figure 9.12.

While these AACFs are not by-definition perfect, they are near perfect, where the zero-autocorrelation is $\approx 10\times$ greater than any other correlation value. This will ensure that the sequence is easily picked up during a fine synchronization step or coarse synchronization, if desired.

From Theorem 1, regardless of the seed selected the FDSS interleaved sequence must consist of mutual orthogonal components in the frequency domain. When interleaved, the linearity of the DFT ensures that the transmitted pilot will cover the total bandwidth that the orthogonal set covers. For the selected seeds, the signals cover the whole frequency domain (take the DFT), so the FDSS interleaved sequence also covers the whole frequency domain when deinterleaved at the received side, with a frequency spacing of $q_{fullchannel} = 1$. This is opposed to the frequency spacing of $q_{subchannel} = 4$ from the original FDSS Σ .

Theorem 2 established that the FDSS is a periodic complementary set and Theorem 3 showed that the FDSS interleaved sequence is a perfect sequence when the seed is perfect, for example, the FZC sequence used in the experiments. Those properties will be further used in the fine synchronization process, which we did not explore in the experiment, since we only conduct the coarse synchronization experiments.

For the performed experiments, good PAPR, near ZAAC, and full frequency bandwidth (with minimal frequency spacing) are all achieved with the proposed pilots generated from the FDSS interleaved sequences, using the seeds shown in Table 9.5.

9.4.4 Channel Feedback

One of the major differences in utility between physical-layer key generation and physical-layer relay attack detection is the existence of established secret keys, which is an observation first established in this thesis. Whether through a public key or symmetric-key scheme, verifiers and provers can share any channel estimate that they get with the confidence that the shared channel estimates are secure (i.e., authentic, confidential, and have integrity). This allows for the correction of non-ideal effects in channels, as well as the ability for devices to perform secure subcarrier power allocation. If the channel feedback is not secure, there are several attacks that can happen. The first attack is an active relay attack. Even if the keyed filters are used to mask the real channel, an attacker could recover the channel state information, if either party reveals its estimated channels. In 3 out of the 4 main protocols proposed in this thesis, channel feedback is utilized to reduce the error caused by noise. The One-Way Decode-and-Forward CBRAD that can detect active and passive relay attacks (5, the feedback is used not only to reduce the effects of channel non-reciprocity, but it is also used to reduce the effects of noise.

9.5 Timing Synchronization

Timing synchronization looks at finding the exact time that a received symbol starts. Given a buffer $\mathbf{q} = (q[0], q[1], \dots, q[Q - 1])$ with Q samples and a signal $r[n]$ with size $N \ll Q$ within it,

$$q[n] = \begin{cases} 0, & n < \tau_n \\ r[n - \tau_n], & \tau_n \leq n \leq \tau_n + N \\ 0, & \tau_n + N < n, \end{cases}$$

where $\tau_n \sim U(0, Q - N)$ and AWGN is ignored for simplicity. The goal of timing synchronization is to identify τ_n from $q[n]$, so $r[n]$ can be extracted. For an SDR that is processing signals digitally, there are two timing synchronization errors that can occur: intra-sample and inter-sample timing synchronization error. The intra-sample error, is also known as the sample timing offset (STO) and occurs when the sampling of the analog OFDM symbol from the ADC is inbetween the original sampling point of the source digital signal. An example of this error can be seen in Figure 9.13, where ε_Δ is the intra-sampling timing error.

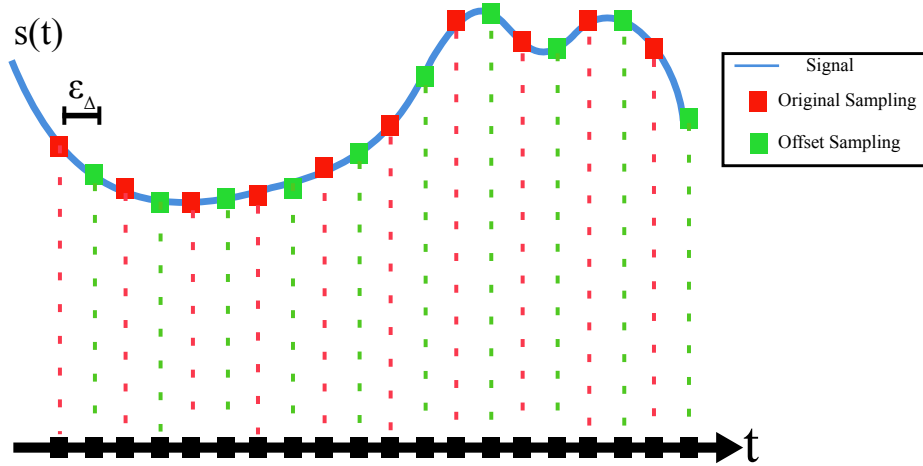


Figure 9.13: Intra-sampling timing error when sampling an analog signal from an original discrete signal

For simplicity, assume there is no AWGN for a received analog OFDM symbol (before the application of any cyclic-fix and preamble) $s(t)$ and corresponding sampled signal $\mathbf{s} = (s[0], s[1], \dots, s[N - 1])$, where T is the symbol duration and T_s is the sample period. The symbol comes from the application of symbols in the frequency domain, where $\tilde{\mathbf{s}} =$

$F\{\mathbf{s}\} = (\tilde{\mathbf{s}})$. The resulting intra-sampling timing error results in the following effect for the received subcarrier symbol $\mathbf{r} = (r[0], r[1], \dots, r[N-1])$:

$$\begin{aligned}
r[n] &= r(t) \Big|_{t=nT_s + \frac{\varepsilon_\Delta}{T_s}} \\
&= \sum_{m=0}^{N-1} \tilde{s}[m] e^{\frac{j2\pi(n + \frac{\varepsilon_\Delta}{T_s})m}{N}} \\
&= \sum_{m=0}^{N-1} \tilde{s}[m] e^{\frac{j2\pi nm + j2\pi \frac{\varepsilon_\Delta}{T_s} m}{N}} \\
&= \sum_{m=0}^{N-1} \tilde{s}[m] e^{\frac{j2\pi nm}{N}} e^{\frac{j2\pi \frac{\varepsilon_\Delta}{T_s} m}{N}} \\
&= \sum_{m=0}^{N-1} \left(\tilde{s}[m] e^{\frac{j2\pi \frac{\varepsilon_\Delta}{T_s} m}{N}} \right) e^{\frac{j2\pi nm}{N}}, \tag{9.18}
\end{aligned}$$

where the term ε_Δ/T_s represents the discrete intra-sample error. The result from (9.18) shows that intra-sample timing error results in a per subcarrier phase error with a frequency given by $2\pi(\varepsilon_\Delta/T_s)/N$. Since every OFDM symbol contains pilots, the error can be absorbed into the equalization process that removes the effects of the channel. An example of the STO effect can be seen in Figure 9.14, which shown a QPSK-OFDM transmission between 2 Pluto SDRs, where 4 QPSK symbols are received on 20 neighbouring subcarriers. The effect of the phaseshift over different subcarriers can be seen.

Inter-sampling timing error occurs if the estimated position $\hat{\tau}_n$ of the signal $\mathbf{r}$ is missed. There are two cases where errors can occur:

1. The estimated start overshoots the real start: $\hat{\tau}_n > \tau_n$,
2. The estimated start undershoots the real start: $\hat{\tau}_n < \tau_n$,

In the first case ($\hat{\tau}_n > \tau_n$), the resulting estimate of $\hat{r}[n]$ becomes:

$$\begin{aligned}
\hat{r}[n] &= \begin{cases} r[n - (\hat{\tau}_n - \tau_n)], & n < N - (\hat{\tau}_n - \tau_n) \\ 0, & n \geq N - (\hat{\tau}_n - \tau_n) \end{cases} \\
&= \begin{cases} r[n + \varepsilon_\tau], & n < N + \varepsilon_\tau \\ 0, & n \geq N + \varepsilon_\tau, \end{cases} \tag{9.19}
\end{aligned}$$

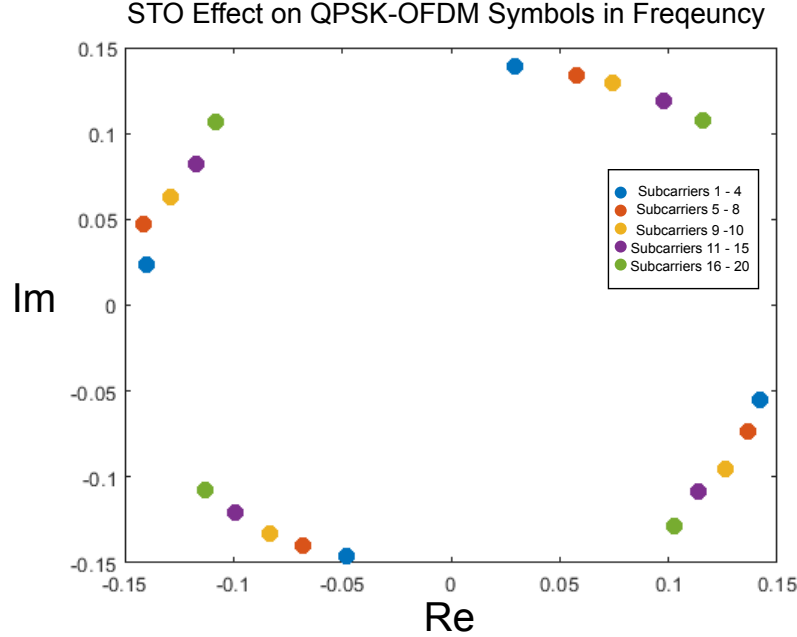


Figure 9.14: STO offset comparing 4 QPSK symbols received on neighbouring subcarriers (20 QPSK total) in a single OFDM symbols between 2 Pluto SDRs in LOS

where $\varepsilon_\tau = \tau_n - \hat{\tau}_n$ is the inter-sample timing error. In the second case ($\hat{\tau}_n < \tau_n$), the resulting estimate of $\hat{r}[n]$ becomes:

$$\hat{r}[n] = \begin{cases} 0, & n < -\varepsilon_\tau \\ r[n - \varepsilon_\tau], & n \geq N - \varepsilon_\tau \end{cases} \quad (9.20)$$

Since the intra-sampling timing error can be dealt with through equalization, the main implementation is focused on creating an algorithm with a good running timing, that minimizes the inter-sample timing error ε_τ on the Pluto SDR. A secondary goal for the timing synchronization process is to make a binary determination if the $\mathbf{q}$ even contains the signal $\mathbf{r}$ within it (i.e., “coarse timing synchronization”) a subset of timing synchronization), so interference or AWGN is not misinterpreted as the signal. To meet these two goals: identifying if $\mathbf{r}$ is within $\mathbf{q}$ and then minimizing ε_{τ_n} , the synchronization process usually follows a two step process of coarse timing synchronization, followed by fine timing synchronization.

A novel timing synchronization algorithm is proposed to minimize the error ε_τ , while performing coarse timing synchronization in an efficient running time. The algorithm is

designed with the specific goal of working on the embedded SDRs. Timing synchronization is a well developed area, as most communication systems require it for demodulation. The algorithm proposed is specifically for implementing a timing synchronization method that provides fast coarse timing synchronization, which is required to prevent an unstable queue from forming on the embedded SDR, even if it is not necessarily the optimal timing synchronization method. This is a major requirement for evaluating the protocol implementations. The algorithm utilizes pilots symbols in the preamble of the OFDM transmissions to both provide timing synchronization and equalization in later steps. An overview of the algorithm state machine can be seen in Figure 9.15

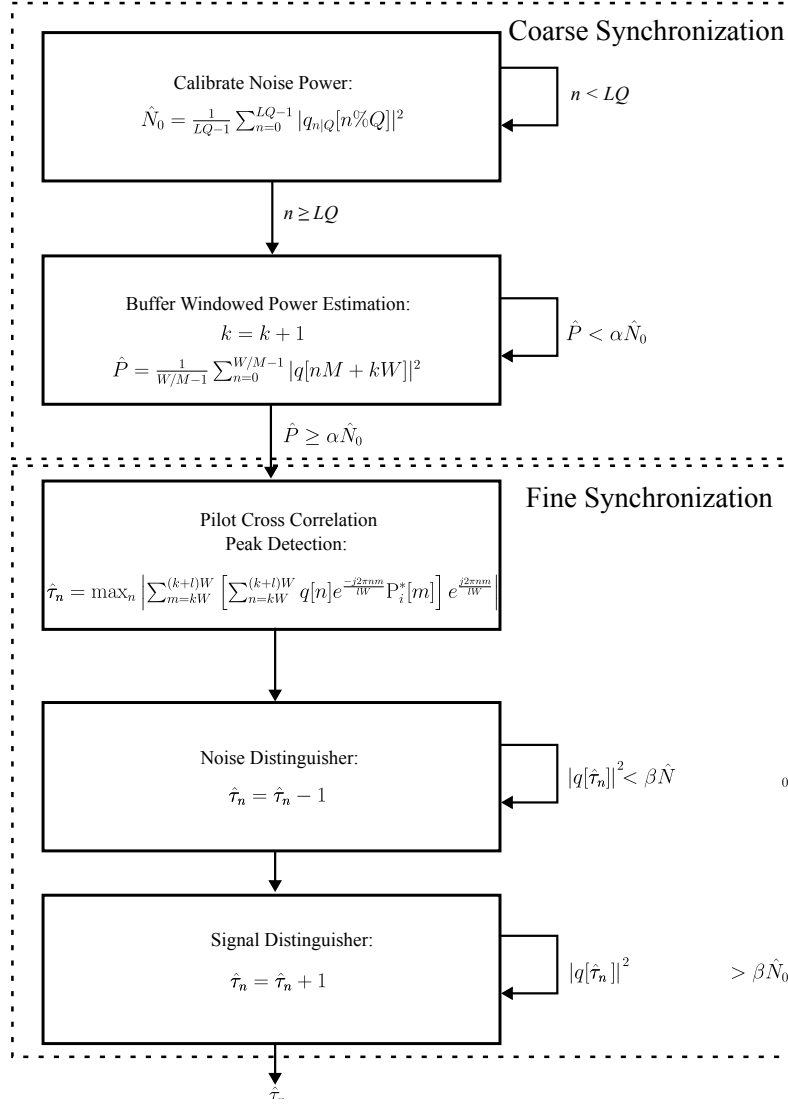


Figure 9.15: State Machine for the timing synchronization algorithm utilized on the embedded SDR for protocol implementation

A general overview of the steps of the algorithm can be seen in Table 9.8, where the algorithm is separated in two sections: coarse timing synchronization in the first two states and fine timing synchronization in the last three states. The details of the algorithm are presented in the subsequent subsections.

Table 9.7: Steps of the timing synchronization algorithm

step	Details
1. Calibrate Noise Power	Estimate the noise power $\hat{N}_0$ from L empty buffers before any further processing
2. Buffer Windowed Power Estimation	Divide an incoming buffer of size Q into sub-blocks of size W . If any block has an average power larger than $\alpha\hat{N}_0$, then extract that block for fine synchronization
3. Pilot Cross Corr. Peak Detection	Take the maximum of the absolute value of the cross corr. of the block as a rough estimate of $\hat{\tau}_n$
4. Signal Distinguisher	Use the signal power to walk back sample by sample, until noise is detected
5. Noise Distinguisher	Use the noise power to walk forward sample by sample, until the signal is detected.

9.5.1 Coarse Timing Synchronization

Coarse Synchronization allows the radio to identify that a symbol has been received in a larger buffer filled with noise samples, from a stream of buffers. This is sometimes called “signal acquisition”. Once a symbol has been identified in a buffer, the whole buffer can be stored in memory for later extraction of the pilot and data using fine timing synchronization, followed by a return transmission to continue the communications. A common method to perform coarse synchronization in an OFDM symbol transmission is to utilize the sliding window autocorrelation and to have two identical pilots appear consecutively in a single symbol. The autocorrelation will result in two peaks: one for $\tau = 0$ due to noise and a second at $\tau = T_p \times f_s$, where T_p is the pilot symbol duration in seconds. The sliding window autocorrelation can identify the pilot type and can estimate the timing offset, but it is not as accurate as using the cross-correlation, which is typically used for fine synchronization. For a noisy signal that is wide-sense stationary (WSS), the fastest way to estimate the autocorrelation is to estimate the power-spectral density (PSD) using the FFT, followed by element wise squaring of the FFT, and an IFFT of the whole buffer (Wiener-Khinchin theorem). The sliding autocorrelation process has a

running time of $O(Q \ln W)$ from the performing of the FFT/IFFT, where W is the window size where the autocorrelation is performed, Q is the size of the buffer, and $Q \geq W$. In contrast, the cross-correlation without any coarse synchronization would have a running time $O(Q \ln Q)$. Sliding-window autocorrelation has the running time reduction compared to the cross-correlation, as it only requires the subtraction and addition of the last and first sample in the FFT as the windows slides. In general, while the FFT is considered to be fast, it can actually be relatively slow on an embedded processor, when compared to the rate at which buffers are filled from a high sampling rate ADC required to get a bandwidth that is multiple MHz.

An alternate and less precise method to perform coarse timing synchronization is to observe the average power over all Q samples in the buffer. If a threshold is selected that is greater than some multiple standard deviation of the noise power, then the receiving radio can safely assume that a buffer contains a some transmitted information. This average power thresholding is

$$\alpha \hat{N}_0 > P_{avg} = \frac{1}{Q} \sum_{n=0}^{Q-1} |q[n]|^2, \quad (9.21)$$

where $\hat{N}_0$ is the estimated noise power, and $\alpha > 2$. The running time of this process is $O(Q)$ operations, as it is checking every element in in the Q sized buffer. If certain assumptions are made, the average power thresholding can be sufficient to provide coarse synchronization for a specific communication protocol. These assumptions include,

1. At the frequency and bandwidth utilized, only the users involved with the protocol are transmitting (including interference),
2. All transmission are half-duplex, so only one party is transmitting at a given time and frequency.

These assumptions can be made if all communications occur in a controlled environment, particularly on the Industrial, Scientific, and Medical (ISM) frequencies bands that see minimal communications. The priority of a timing synchronization can then be broken into two general problem conditions:

1. Fast Coarse Timing Synchronization so no samples are lost from an unstable queue. This requires

$$f_s < \frac{Q}{T_{proc} + T_c},$$

from (9.5), where T_{proc} is the processing time and T_c is the copy time for the buffer to main memory. Given the fixed copy time (for a given buffer size), the T_{proc} must be minimized, otherwise an unstable queue can cause transmitted symbols to be missed. The coarse timing synchronization happens before a symbol is identified, so it must be able to check every incoming sample to ensure it finds a symbol.

2. Fine Timing synchronization does not have to be relatively fast, as once the a buffer passes coarse synchronization, it is assumed that no other transmission is expected by the receiver (half-duplex), until it transmits. The timing limitation for fine timing synchronization is the coherence time of the channel, which can be in the dozens or hundreds of milliseconds, compared to coarse synchronization having to complete in microseconds.

The utilization of the average power can have further linear performance optimization through the use of subsampling (decimation) of the sample buffer. For example, instead of looking at every sample, every M^{th} sample could be observed for the average power thresholding:

$$\alpha \hat{N}_0 > P_{avg} = \frac{1}{\lfloor Q/M \rfloor} \sum_{n=0}^{\lfloor Q/M-1 \rfloor} |q[nM]|^2, \quad (9.22)$$

The running time of this process is still $O(Q)$ operations, as it is a scalar linear search of the buffer size Q . Considering all the synchronization processed mentioned so far, a general overview of a timing synchronization process that incorporates them all can be seen in Figure 9.16. Each step is intended to narrow down the signal timing with progressively more

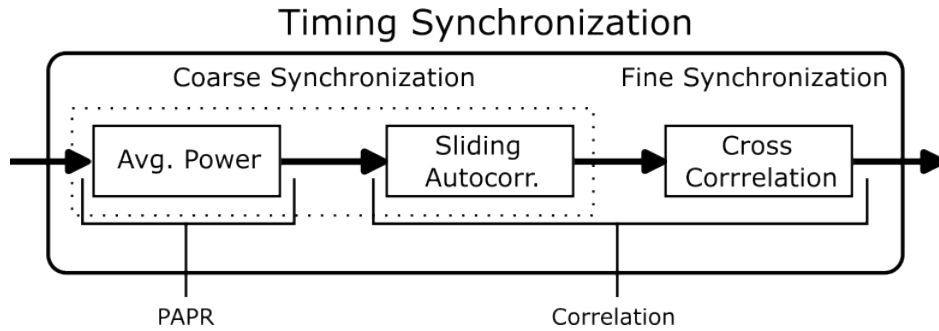


Figure 9.16: General synchronization process and which property of the signal is important for the process [123]

precise, but costly synchronization methods. Since further synchronization utilizes the correlation of the signal, it is important that the pilot has good autocorrelation properties.

A comparison of the coarse synchronization methods running on the embedded ARM Cortex A9 processor on the Pluto SDR is shown in Figure Figure 9.17.

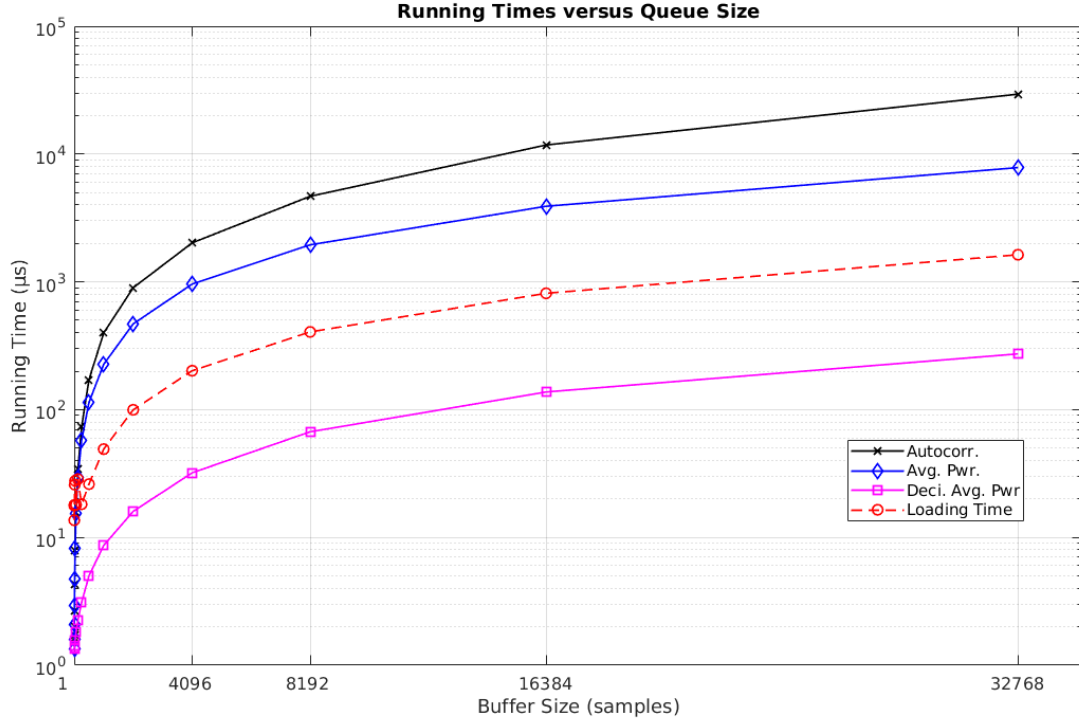


Figure 9.17: Running time in μs versus sample size of a single buffer in the queue with a sampling rate $f_s = 20$ MSPS, decimation rate = 30 [123]

It can be seen that the only method that has a service time consistently below the arrival time is the decimated average power; this is the only stable coarse synchronization method (i.e., does not result in dropped samples). It is also important to note that this does not include other basic processes, such as converting the samples to standardized data types, loading the samples into different memory positions, etc., which would add additional time to these coarse synchronization methods.

Given the assumptions that any time the average power is detected above a certain threshold the algorithm determines that the buffer signal $\mathbf{q}$ contains $\mathbf{r}$, since no other

signal is assumed to come from any other source. Instead of needing a secondary coarse synchronization stage (e.g., sliding autocorrelation) as shown in Figure 9.16, it is sufficient to go from average power detection, to fine timing synchronization with cross-correlation. Furthermore, since transmission are considered to be half-duplex, after the reception the identification of an expected amount a symbols (given by the protocol), the receiver can spend as much time as they want to perform processes (e.g., fine synchronization, demodulation, etc.), since no new transmission is expected. Samples can be dropped in the receiver queue, as they are assumed to only contain noise. The only limitation is that a response happens within the channel coherence time.

The proposed coarse timing synchronization algorithm has two steps, as indicated in Table 9.8:

Table 9.8: Steps of the coarse timing synchronization algorithm

step	Details
1. Calibrate Noise Power	Estimate the noise power $\hat{N}_0$ from L empty buffers of size Q , before any protocol occurs and before any signal coarse synchronization.
2. Buffer Windowed Power Estimation	Divide each incoming buffer of size Q into $\lceil Q/W \rceil$ sub-blocks of size W . The block that has an average power larger than $\alpha\hat{N}_0$ (from 1), then extract that W block and the neighbouring blocks for fine synchronization

A priori knowledge of the noise power $\hat{N}_0$ is required for multiple aspects of the communication system, including the MMSE channel estimator. It is assumed that $\hat{N}_0$ does not change throughout the communications, which is reasonable assumption to make for a fixed environment (i.e., ambient temperature), sampling rate, and device operation (e.g., same algorithms are used, amplifier power, etc.). A preset variable L indicates how many buffers are required to estimate $\hat{N}_0$, which keeps a running average that is updated for each buffer. WithThe algorithm for the average power algorithm is shown in Algorithm 2.

Given that LQ samples are used to calculate the average power, the distribution of $\hat{N}_0$ is given by the distribution of the sample variance [137]

$$\hat{N}_0 \sim \frac{N_0}{LQ} \chi^2(LQ). \quad (9.23)$$

Algorithm 2 Noise Power Calibration

```

1:  $\hat{N}_0 \leftarrow 0$ 
2:  $\ell \leftarrow 1$ 
3: while  $\ell < L$  do
4:    $\hat{N}_0 \leftarrow \frac{1}{\ell Q} (\ell Q \hat{N}_0 + \sum_{n=0}^{Q-1} |q_\ell[n]|^2)$ 
5:    $\ell \leftarrow \ell + 1$ 
6: end while
7: return  $\hat{N}_0$ 

```

The algorithm calibration happens immediately at start up of the devices, before any transmission occurs. Under the assumptions regarding the transmission environment, the calibration only picks up the noise power. In all the experiments, $L = 1000$ is selected. This results in LQ samples. A typical buffer size of $Q = 10240$, would give 10240000 samples. Given that the sample variance has a MSE of [136]

$$\text{MSE} = \frac{2N_0^2}{LQ},$$

where the resulting mean error is $\approx \pm 1/2262$ of N_0 . In terms of sampling, this is $\approx 1/47.6$. For a signal that is amplified to the peak for a 12-bit twos-complement (i.e., 2048), an SNR of 5 db (the lowest used in the experiments) results in an average noise sample of $2048/\sqrt{10} \approx 650$. This could yield a mean sample error of ≈ 14 samples, which is less than 1 % (0.68 %) of the maximum signal value. This algorithm has a running time of $O(Q)$ in the buffer size, when only considering operations

After $\hat{N}_0$ is estimated the devices continue with whichever protocol or transmission in their state machine. If a device is in a receive state it continuously processes buffer using the buffer windowed power estimation algorithm (step 2 in coarse synchronization). The algorithm works by dividing the buffer into blocks of size W , where $W < Q$. For example in all the experiments performed $W = 640$, which is $4 \times$ the length of the preamble. For a buffer size of $Q = 10240 = 4 \times 4 \times 640$, this results in 16 blocks being checked. Each block follows the same process as the noise estimation. The average power is measured in each block, until a block has power $> \alpha \hat{N}_0$. For a given buffer $\mathbf{q} = (q[0], q[1], \dots, q[Q-1])$, where the w^{th} block is $0 \leq w < \lfloor Q/W \rfloor$ and has an average power of given by

$$\hat{P}_w = \frac{1}{M} \sum_{n=0}^{\lfloor W/M \rfloor - 1} q[nM + kW],$$

where M is the sub-sampling (decimation) rate used to linearly improve performance. After noise calibration, the decimation is required to ensure that the sampling stable, as shown in Figure 9.17. If $\mathbf{r} = (0, 0, \dots, 0)$, then $\hat{P}_w \sim N_0/\lfloor W/M \rfloor \chi^2(\lfloor W/M \rfloor)$, which is a weaker sample variance compared to the noise power estimate (9.23). Given α and $\hat{N}_0$, the probability of getting a false positive (i.e., selecting a block to contain a received signal, when the block only contain noise) is given by

$$\mathrm{P} \left(\hat{P}_w > \alpha \hat{N}_0 | \mathbf{r} = (0, 0, \dots, 0) \right) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f_{\hat{P}_w, \hat{N}_0}(\hat{P}_w, \alpha \hat{N}_0) d\hat{P}_w d\hat{N}_0 \quad (9.24)$$

$$= \int_{-\infty}^{+\infty} f(\hat{P}_w) d\hat{P}_w \int_{-\infty}^{\frac{\hat{P}_w}{\alpha}} f_{\hat{N}_0}(\hat{P}_w) d\hat{N}_0 \quad (9.25)$$

$$= E_{\hat{P}_w} \left[F_{\hat{N}_0} \left(\frac{\hat{P}_w}{\alpha} \right) \right] \quad (9.26)$$

$$= E_{\hat{P}_w} \left[\frac{\gamma(\frac{LQ}{2}, \frac{LQ\hat{P}_w}{2\alpha\hat{N}_0})}{\Gamma(\frac{LQ}{2})} \right], \quad (9.27)$$

where f_X is the PDF of the given random variable. Line (9.25) comes from the independence of AWGN at each sample ($\mathrm{P}(\hat{P}_w, \hat{N}_0) = \mathrm{P}(\hat{P}_w)\mathrm{P}(\hat{N}_0)$). Line (9.27) comes from the CDF of the χ^2 distribution, where γ is the lower incomplete gamma function and Γ is the gamma function.

The probability of the false-negative (i.e., $\mathrm{P}(\hat{P}_w < \alpha \hat{N}_0 | \mathbf{r} \neq (0, 0, \dots, 0))$) depends on the length of $\mathbf{r}$ and the position $\tau_n \sim U(0, Q - N)$. To simplify the evaluation, assume that $\mathbf{r}$ consists of the repetition of the 4 pilots with cyclic extension (i.e., $N = 4 \times 160 = W$, as described in Section 9.3. Given the pilot selection, the PAPR of $\mathbf{r}$ is also considered to be 2.5 (Table 9.6), for simplicity. Considering these assumptions

$$\mathrm{P} \left(\hat{P}_w < \alpha \hat{N}_0 | \mathbf{r} = (r[0], r[1], \dots, r[N-1]) \right) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f_{\hat{P}_w, \hat{N}_0}(\hat{P}_w, \alpha \hat{N}_0) d\hat{P}_w d\hat{N}_0 \quad (9.28)$$

$$= \int_{-\infty}^{+\infty} f(\hat{P}_w) d\hat{P}_w \int_{-\infty}^{\frac{\hat{P}_w}{\alpha}} f_{\hat{N}_0}(\hat{P}_w) d\hat{N}_0 \quad (9.29)$$

If a block w has an average power that passes the threshold, then that block, the previous block, and the next 2 blocks are stored in main memory for future processing (total of 4 blocks). The selected block can be found in the range $0 \leq w \leq 15$, where 0

indicates the threshold passing in the first W samples and 15 the last W samples, with equal probability since τ_n has a uniform distribution. The reason that the previous block is selected is to ensure that no “straggling” occurs with the block. An example of this can be seen Figure 9.18, where block 1 would have likely been missed, but Block 2 would very likely be hit. To ensure that the first sample is not missed, the first 2 blocks must be extracted. Subsequent blocks are extracted to capture any OFDM symbols. For example, if a single OFDM symbol was sent, there would be 3 blocks extracted, out of the total 16. The sub blocks are what are used in the fine synchronization step. The full algorithm can be seen in algorithm 3.

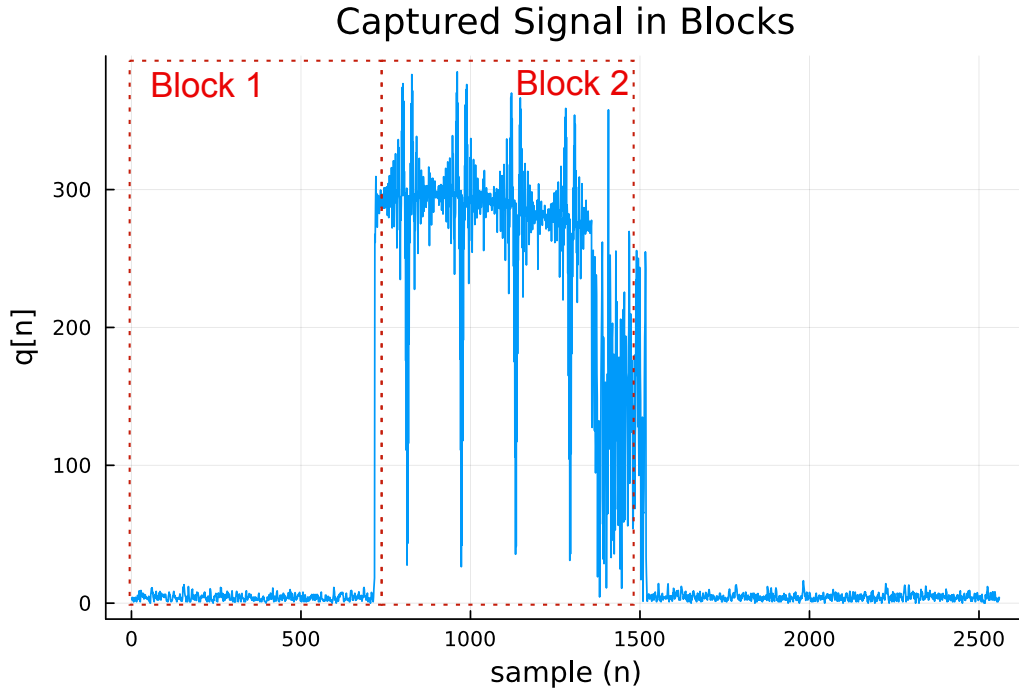


Figure 9.18: Example of a received transmission on a Libre SDR, where the portion of the received signal could be in 2 blocks

It is important to note that this can also happen between buffers. If the average power threshold fails for every block in a buffer, the last block is still saved in local memory. If the first block in the next buffer has an average power larger than the threshold, then the saved block is also passed. This corresponds to a situation where $w = 0$. It is also possible that the symbol is detected in one buffer but extends into the next one. This corresponds to $w \geq 12$. In this case it is required that the next buffer must also be loaded.

With this process, if $\mathbf{r}$ is in the blocks, then it is guaranteed that any extracted blocks $\mathbf{q}_{\text{coarse}}$, with size $6W$, has r start somewhere in the first 2 blocks of size $2W$ inside $\mathbf{q}_{\text{coarse}}$. For all the experiments $\alpha = 2$ was selected. Using thresholding of the average power in the buffer reveals if a signal is in the buffer, but it does not say anything about where in the buffer the signal may be.

Algorithm 3 Buffer Windowed Power Estimation

```

1:  $w \leftarrow 0$ 
2: while  $w < \lfloor Q/W \rfloor$  do
3:    $\hat{P}_w \leftarrow \frac{1}{\lfloor W/M \rfloor} \sum_{n=0}^{\lfloor W/M \rfloor - 1} |q_\ell[nM + w \lfloor Q/W \rfloor]|^2$ 
4:   if  $\hat{P}_w > \alpha \hat{N}_0$  then
5:     return  $w$ 
6:   end if
7:    $w \leftarrow w + 1$ 
8: end while
9: return No Symbol Detected

```

Simulation Results

To evaluate the performance of the coarse timing synchronization with a pilot sequence generated using the generalized Milewski construction, and to evaluate the performance of average power coarse timing synchronization, simulations were performed to gather the Monte-Carlo results. The simulations were performed in the Julia programming language on a desktop computer with an Intel i9-10900F processor and 64 GB of RAM. The full simulation code and archived results are openly available on GitHub [132]. Simulations were designed to recreate the processes seen by the SDR implementation. Buffers were simulated with a uniformly distributed random arrival times of pilots, with a mean arrival time being the desired RTT. Coarse synchronization of buffers occurs as they get filled (i.e., under an assumption of stability), which is assumed to only be the case using an average power coarse synchronization. The simulation was ran using the same sequences described in Table 9.5 with an interpolation rate of 4. The naive pilot is the pilot defined in (9.17). The threshold for detecting a pilot was selected to be twice the estimated noise power found in any $1/2$ -sector of a buffer. Given this threshold, only an SNR of 5 dB or higher is considered. The performance criterion being evaluated in the simulation is the false-negative rate (FNR) of the decimated average power coarse synchronization process

for a given signal-to-noise ratio (SNR). The FNR is defined as:

$$FNR = 1 - \frac{\text{Transmissions Detected}}{\text{Transmissions Sent}}, \quad (9.30)$$

where a 0 FNR means every pilot transmitted was detected from the coarse synchronization process at the receiver. The higher the FNR, the worse the performance is. For simplicity, in these evaluations the only consideration is an additive white Gaussian noise (AWGN) channel (i.e., no Dopplers effect or multipaths). The results from the simulation can be seen in Figure 9.19, where the results are compared to the naive OFDM pilots previously mentioned.

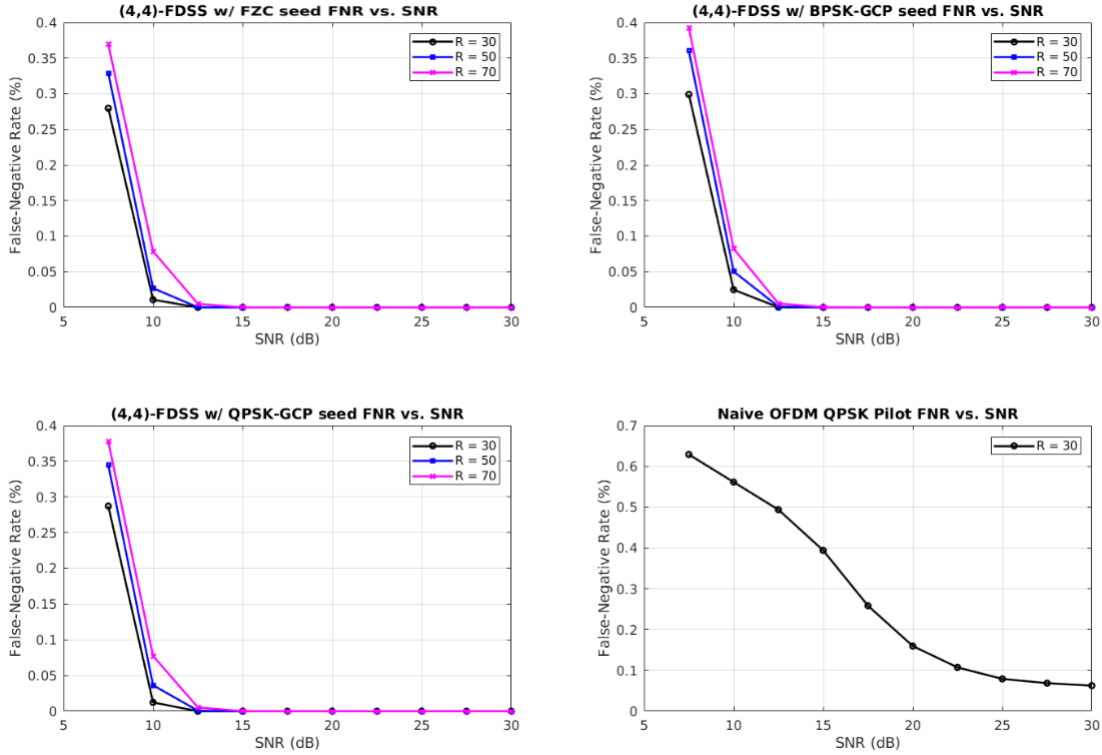


Figure 9.19: False-Negative Rates versus SNR for different FDSS interleaved sequences using various seed sequences: (1) FZC (2) BPSK-GCP (3) QPSK-GCP (4) OFDM QPSK [123]

Table 9.9: Experiment parameters for the SDRs

Parameter	Value
Carrier Freq. (MHz)	915
Sample Rate (MSPS)	20
Rx/Tx BW (MHz)	12.8
Rx. Gain (dB)	0
Tx. Gain (dB)	-
SNR (dB)	-

It can be observed that all of that for all the various seed sequences, the pilots have similar performance, regardless of the decimation rates, with only slight improvements as the decimation decreases. For higher SNRs, this could mean that we could gain significant increases in service rates by using as high of a decimation rate as possible, without concern for missing pilots. When comparing against the OFDM pilots, the other sequences have significantly better results. Even at a relatively high SNR of 20 dB and the lowest decimation rate ($R = 30$), the receiver is missing nearly 20% of the transmitted OFDM pilots.

Hardware Experiment Results

To corroborate the simulations results seen in Section 9.5.1, hardware experiments were also performed using two Pluto SDRs, where they were connected directly to each other through an SMA wire to ensure that the only channel consideration was AWGN, matching the simulation. Coarse synchronization is performed on the embedded processor on the receiver SDR. The parameters for the SDR can be seen in Table 9.9, where the transmitter gain is adjusted for each experiment to meet the desired SNR. A picture of the experimental setup can be seen in Figure 9.20.

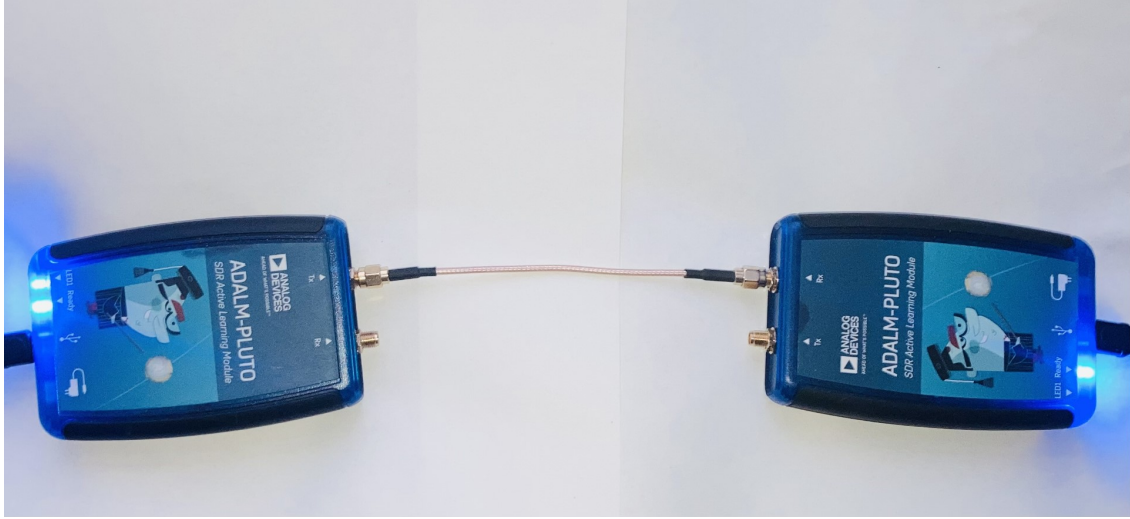


Figure 9.20: Analog Devices ADALM-Pluto SDRs connected with an SMA cable [123]

The results from the experiment can be seen in Figure 9.21, where we compared the OFDM QPSK pilot versus the FDSS interleaved sequence with an FZC seed. The use of the selected pilots, drastically improves the detection rate on the SDRs. When comparing the experimental results to the results in Figure 9.19, we can see that they are very close, with high correlation and a small offset for the FNR.

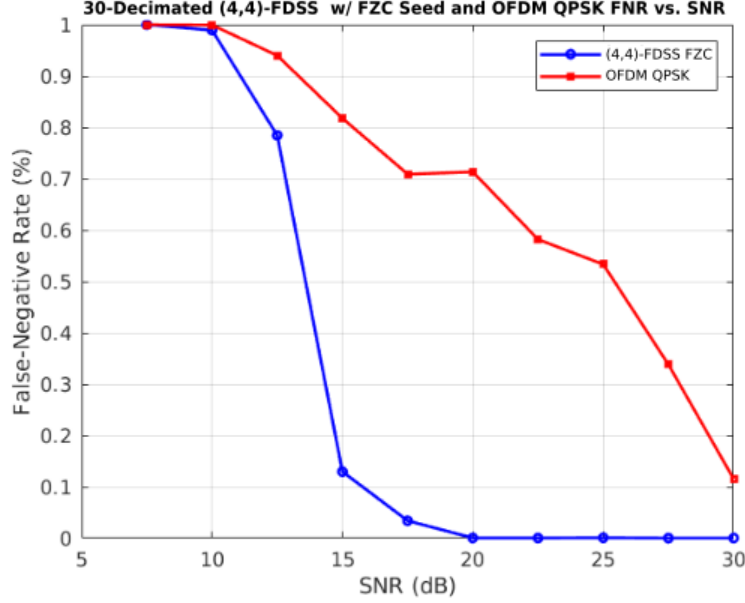


Figure 9.21: FDSS interleaved sequence comparison from coarse synchronization experiment on SDRs [123]

9.5.2 Fine Timing Synchronization

As mentioned in Section 9.5.1, the goal of coarse timing synchronization is to perform signal acquisition within a received buffer and to narrow down the location of the signal with the buffer to sub-window. Both of these steps are used to help reduce the size of the input signal for the subsequent processes and to quickly process incoming samples, without resulting in an unstable sample queue. The actual evaluation of $\hat{\tau}_n$ is performed in the fine timing synchronization process. This is achieved through the use of the peak of the cross-correlation, which is the standard for most block-type pilot systems, where the pilots are placed in the preamble, like in WiFi. Given a preamble that contains pilots with good signal properties, particularly correlation and PAPR, it is possible to perform timing synchronization to detect the start of a signal. To improve upon established methodologies that simply uses FZC sequences [133], the usage of the mean of the instantaneous signal power is proposed to quickly improve the synchronization error, particularly in a noisy environment where the cross-correlation may not work well. The proposed algorithm has a worst case running timing of $O(Q)$, though in reality is significantly smaller. The steps in the fine timing synchronization algorithm are detailed in Table 9.10.

Table 9.10: Steps of the timing synchronization algorithm

step	Details
1. Pilot Cross Corr. Peak Detection	Calculate the cross-correlation of the expected preamble with the blocks. Take the maximum of the absolute value of the cross-correlation of the block as a first estimate of $\hat{\tau}_n$.
2. Signal Distinguisher	Use the instantaneous signal power at $\hat{\tau}_n$ to walk back using a sliding window sample by sample from $\hat{\tau}_n$, until noise is detected (low power).
3. Noise Distinguisher	Use the instantaneous signal power at $\hat{\tau}_n$ to walk forward using a sliding window sample by sample from $\hat{\tau}_n$, until the preamble is detected (high power).

The pilot cross-correlation looks at the cross-correlation of the expected preamble pilots, with a length of $4N_p$, where N_p is the length of the pilot,

$$\mathbf{p}_{\text{pre}} = (\overbrace{p[0], \dots, p[N_p - 1]}^{\text{pilot 1}}, \overbrace{p[0], \dots, p[N_p - 1]}^{\text{pilot 2}}, \overbrace{p[0], \dots, p[N_p - 1]}^{\text{pilot 3}}, \overbrace{p[0], \dots, p[N_p - 1]}^{\text{pilot 4}}), \quad (9.31)$$

with the extracted block

$$\mathbf{q}_{\text{coarse}} = (q[(w - 1)W], q[(w - 1)W + 1], \dots, (w + 4)W - 1], \dots, (w + 6)W - 1],$$

which for simplicity will be considered to only span over a single buffer ($0 < w < 12$). This extracted buffer has a length of $6W$. For speeding up the calculation, the zero-padded cross-correlation is used so that the DFT can be used to calculate the AAC. The padded length is $6W + 4N_p - 1$, such that

$$\mathbf{p}_{\text{prepad}} = (\mathbf{p}_{\text{pre}}, \overbrace{0, 0, \dots, 0}^{6W-1}),$$

and

$$\mathbf{q}_{\text{coarsepad}} = (\mathbf{q}_{\text{coarse}}, \overbrace{0, 0, \dots, 0}^{4N_p-1})$$

where their respective DFTs are $\tilde{\mathbf{p}}_{\text{pre_pad}}$ and $\tilde{\mathbf{q}}_{\text{coarse_pad}}$. The cross-correlation estimate is given by

$$\hat{C}_{\mathbf{p}\mathbf{q}}[\tau] = \mathbf{F}^{-1}\{\mathbf{D}\{\mathbf{p}_{\text{pre_pad}}\}\mathbf{q}_{\text{coarse_pad}}\}. \quad (9.32)$$

The first $\hat{\tau}_n$ is evaluated given (9.32)

$$\hat{\tau}_n = 6W + 4N_p + 1 - \max_{\tau} \hat{C}_{\mathbf{p},\mathbf{q}}[\tau] \quad (9.33)$$

These processes are calculated using the FFT of the padded inputs, which has a running time of

$$O((W + N_p)\ln(W + N_p)) = O(W\ln(W)), \quad (9.34)$$

since $W = 4N_p$. Once the estimate is found, the instantaneous power of $\mathbf{q}_{\text{coarse}}$ is used to adjust $\hat{\tau}_n$. The adjustments attempts to take advantage of the signal power difference between noise and the detected signal to find the point where the preamble and signal power change, assuming that $\hat{\tau}_n$ is near this transition point. In the worst case, this process is $O(W)$, as every sample in $\mathbf{q}_{\text{coarse}}$ checked, but in practice this process is very fast, as $\hat{\tau}_n$ is close to τ_n (single digit ε_τ). If this assumption is made, then the ability to distinguish the signal from noise depends on the SNR, but also how flat the signal itself is in the preamble. The selection of a pilot with a low PAPR is important for the usage of the signal power to distinguish a pilot. If a pilot has a large PAPR, there could be portions of the signal that have similar power to noise, which can make it difficult to distinguish the difference. The algorithm for this process can be seen in algorithm 4.

Algorithm 4 Instantaneous Power Detection

```

1: while  $|q_{\text{coarse}}[\hat{\tau}_n - 1]|^2 > \beta \hat{N}_0$  AND  $\dots$  AND  $|q_{\text{coarse}}[\hat{\tau}_n - K]|^2 > \beta \hat{N}_0$  do
2:    $\hat{\tau}_n \leftarrow \hat{\tau}_n - 1$ 
3: end while
4:
5: while  $|q_{\text{coarse}}[\hat{\tau}_n + 1]|^2 < \beta \hat{N}_0$  AND  $\dots$  AND  $|q_{\text{coarse}}[\hat{\tau}_n + K]|^2 < \beta \hat{N}_0$  do
6:    $\hat{\tau}_n \leftarrow \hat{\tau}_n + 1$ 
7: end while
8:
9: return  $\hat{\tau}_n$ 

```

Individual noise samples can have large powers, so the noise power of multiple samples is observed, relative to the average power. Large in this case, is on the order of the βN_0 ,

which is similar to the signal power. The probability of this is given from the CDF of a K independent Gaussian given $\hat{N}_0 \sim \frac{N_0}{LQ} \chi^2(LQ)$ (9.23). Assuming $\varepsilon_\tau = 0$ (i.e., $\hat{\tau}_n = \tau_n$), the probability of a false-negative resulting in a backward adjustment is given by

$$P(\text{FN}) = P(|q_{\text{coarse}}[\hat{\tau}_n - 1]|^2 < \beta \hat{N}_0, |q_{\text{coarse}}[\hat{\tau}_n - 2]|^2 < \beta \hat{N}_0, \dots, |q_{\text{coarse}}[\hat{\tau}_n - K]|^2 < \beta \hat{N}_0) \quad (9.35)$$

$$= (P(|q_{\text{coarse}}[\hat{\tau}_n - 1]|^2 < \beta \hat{N}_0))^K \quad (9.36)$$

$$= \left(\int_{-\infty}^{\infty} f_{\hat{N}_0}(\hat{N}_0) d\hat{N}_0 \int_{-\infty}^{\beta \hat{N}_0} 2\Phi\left(\frac{x}{\sqrt{\hat{N}_0}}\right) dx \right)^K \quad (9.37)$$

$$= \left(\int_0^{\infty} f_{\hat{N}_0}(\hat{N}_0) d\hat{N}_0 \int_{-\infty}^{\beta \hat{N}_0} 2\Phi\left(\frac{x}{\sqrt{\hat{N}_0}}\right) dx \right)^K, \quad (9.38)$$

where the previous next samples have large instantaneous power. When $\varepsilon_\tau = 0$, forward adjustments will not erroneously happen, given a pilot with a low PAPR, as the preamble signal power should be larger than $\beta \hat{N}_0$ for each sample checked.

Once the fine timing synchronization is complete and $\hat{\tau}_n$ has been found, the preamble and OFDM symbol can be extracted as

$$\mathbf{q}_{\text{extract}} = (q[\hat{\tau}_n], q[\hat{\tau}_n + 1], \dots, q[\hat{\tau}_n + 4N_p + L_d N_p - 1]), \quad (9.39)$$

after which the rest of the signal processing can occur to continue with the communication protocol.

9.6 Carrier Frequency Offset Elimination

When two devices have independent oscillators that are used to independently generate the carrier frequency f_c , it is unlikely that each device will get the same frequency. The lower quality the oscillators are, the larger this difference becomes. Usually this error is known as the “frequency stability” and is given in parts per million (ppm) or parts per billion (ppb). The default oscillator in the Pluto SDR (Rakon RX03225M) has a frequency stability of ± 25 ppm, which is relatively bad [138]. For example, at 915 MHz the frequency difference between two Pluto SDRs could be as large as $2 \times 25 \times 10^{-6} \times 915 \times 10^6 = 45.750$ kHz. In contrast, the Libre SDR is advertised as having a frequency stability of ± 0.5 ppm, which would result in a maximum error of $2 \times 0.5 \times 10^{-6} \times 915 \times 10^6 = 915$ Hz. Against

a Pluto SDR, this could result in an error of $(0.5 + 25) \times 10^{-6} \times 915 \times 10^6 = 23.332$ kHz. This frequency difference is defined as Δf . From observing the transmission and reception chain for OFDM in Figure 2.8, the consequence of Δf on an OFDM transmission can be derived. For simplicity ignore the effects of the channel, noise, the cyclic-prefix, and other non-idealities (e.g., timing mismatch, etc.). For a transmission of an OFDM symbol $s[n] = \sum_{m=0}^{N-1} \tilde{s}[m] e^{j2\pi \frac{n}{N} m}$, the received symbol can be seen as,

$$\tilde{r}[m] = \frac{1}{N} \sum_{n=0}^{N-1} \left(\sum_{m'=0}^{N-1} \tilde{s}[m'] e^{j2\pi \frac{n}{N} m'} \right) e^{j2\pi f_{c1} n T_s} e^{-j2\pi f_{c2} n T_s} e^{-j2\pi \frac{m}{N} n} \quad (9.40)$$

$$\begin{aligned} &= \frac{1}{N} \sum_{n=0}^{N-1} \left(\sum_{m'=0}^{N-1} \tilde{s}[m'] e^{j2\pi \frac{n}{N} m'} \right) e^{j2\pi (f_{c1} - f_{c2}) \frac{n}{f_s}} e^{-j2\pi \frac{m}{N} n} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} \left(\sum_{m'=0}^{N-1} \tilde{s}[m'] e^{j2\pi \frac{n}{N} m'} \right) e^{j2\pi \frac{-\Delta f_c}{f_s} n} e^{-j2\pi \frac{m}{N} n} \\ &= \frac{1}{N} \sum_{m=0}^{N-1} \tilde{s}[m'] \sum_{n=0}^{N-1} e^{j2\pi \frac{(m' - m)}{N} n} e^{j2\pi \frac{-\Delta f_c}{f_s} n} \end{aligned} \quad (9.41)$$

$$= \frac{1}{N} \tilde{s}[m] \sum_{n=0}^{N-1} e^{j2\pi \frac{-\Delta f_c}{f_s} n}, \quad (9.42)$$

where the carriers in (9.40) are the 2 different carriers at modulation a demodulation of the transmitter and receiver after the receiver's ADC with sampling rate $f_s = 1/T_s$. The result of (9.42) shows that for a very small Δf_c , relative to the sampling frequency f_s , there can be considered a constant phase-shift on $\tilde{s}[m]$ over the N subcarriers. For subsequent symbols, phase-shift grows linearly over time. This effect is commonly known as the carrier frequency offset (CFO). An example of this can be seen between 2 Pluto SDRs in Figure 9.22 with 5 QPSK-OFDM symbols, through the observation of the first subcarrier only. As can be seen in the figure, 4 QPSK symbols in the first OFDM symbol (blue) rotates over the subsequent OFDM symbols.

Remove the effect of the CFO requires the modulation of received signals by $e^{j2\pi \Delta f_c}$. This requires the estimation of $\hat{f}_c$ to remove this. To perform this estimation, this thesis proposes a modified Moose's method [145]. With the preamble structure having $4\times$, as discussed in Section 9.3. Given that the pilots are 160 samples, any sampling rate over 1 MSPS will ensure that the channel is coherent over each of the samples. Ignoring the other possible effects (e.g., STO, etc.) and only considering the static channel, the only only

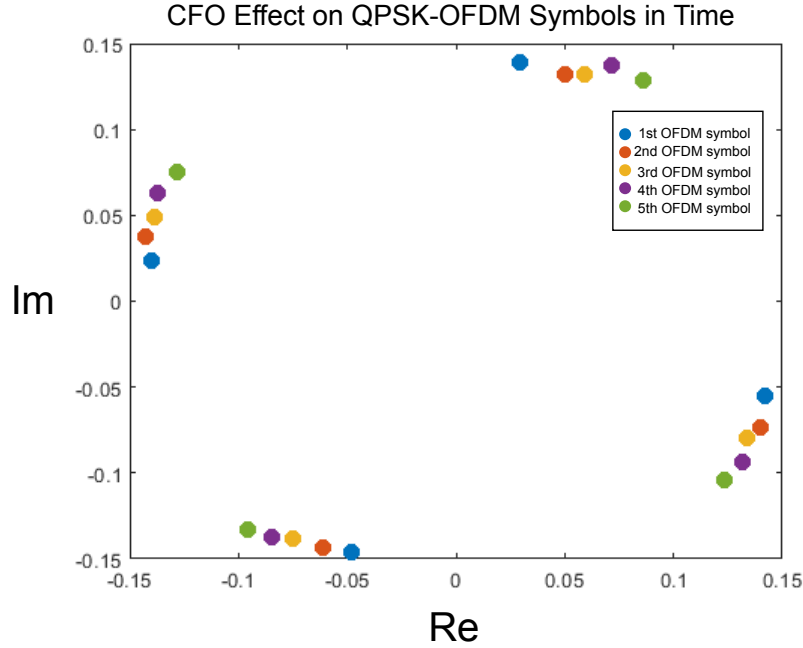


Figure 9.22: CFO comparing 4 QPSK symbols received on 4 subcarriers over 5 different OFDM symbols between 2 Pluto SDRs in LOS

difference between the pilots is the linear phase change caused by the CFO (9.42). Given a received $\mathbf{p}_{\text{pre}}$ (9.31), represented as $\hat{\mathbf{p}}_{\text{pre}}$. Using each frequency sample in each position of the 4 pilots, with the corresponding position within the other pilots, the estimate is given by

$$\frac{\Delta f_c}{f_s} = \frac{1}{N(\ell-1)} \left(\sum_{\ell=0}^{L-1} \sum_{n=0}^{N-1} \angle \hat{p}[n + \ell N] - \angle \hat{p}[n + (\ell+1)N] \right) \quad (9.43)$$

This process performs Moose's estimation on each neighbouring pilot subcarrier, instead of just 2 pilots. The more pilots chained together, the better the process is. The CFO estimation occurs after the timing synchronization step, where the preamble and OFDM symbol is extracted. A visualization of the process can be seen in Figure 9.23.

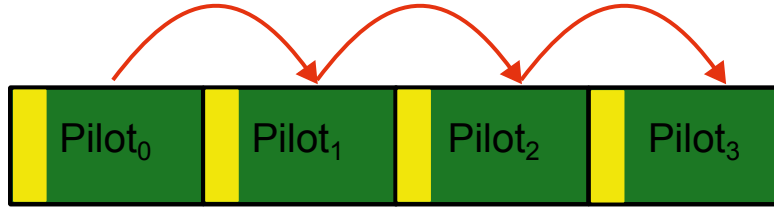


Figure 9.23: Neighbouring Moose's estimate on neighbouring pilots

An example of the result of the algorithm on OFDM symbols being shifted can be seen in Figure 9.24. There is still an overall phaseshift as the channel effects have not been removed.

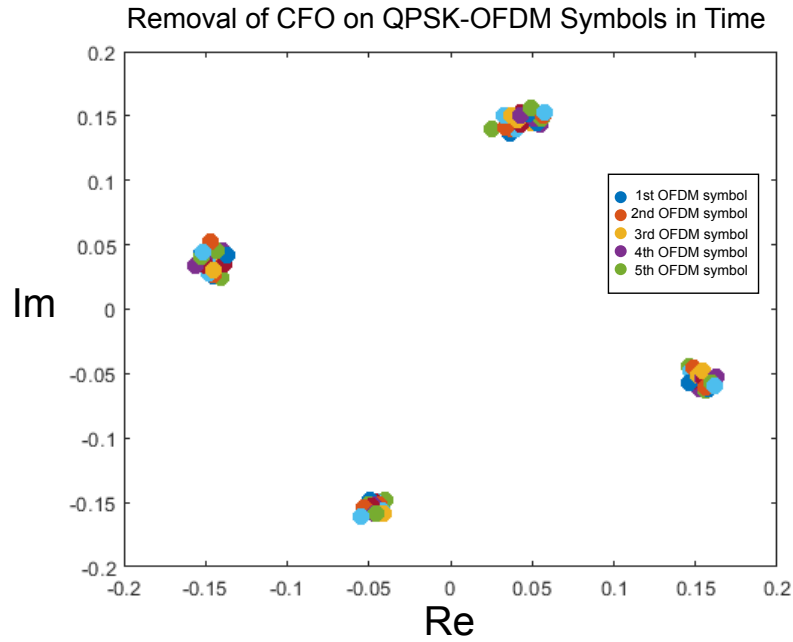


Figure 9.24: QPSK symbols in time after removal of CFO in first 5 symbols

9.7 Secure Waterfilling and Adaptive Modulation

In this section, a novel protocol for establishing secure channel feedback in a frequency-selective channel is proposed. Since channel information should be obscured against an

active relay attacker, it is important to ensure that any frequency power allocation does not reveal information about the physical channels to the relay attacker, which could then be used by used in active relay attacks. As a small symbol duration is a requirement for high-throughput data communications, it is common that the resulting transmissions is wideband. As a result, the transmissions can be affected by a frequency-selective channel, which can cause fading over subcarriers. This can result in symbol errors at the receiver side. Though OFDM helps reduce this effect by spreading the data over multiple subcarriers, frequency-selective fading can still be a problem, as some data symbols will be lost to AWGN at those subcarriers (i.e., some amount of bit error). The problem is further exacerbated by the random FIR filters introduced in the relay detection protocols. These filters essentially introduce frequency-selectivity, on top of the physical channel. Any bit error that occurs can have a significant impact on the protocols presented in this thesis. The particular issues from bit error include:

1. Challenge-response authentication protocols failing, because message authentication codes or digital signatures are used to perform verification. A single bit error can cause the entity authentication to fail, as these cryptographic primitives are random mappings. This will cause the false-positivity rate to increase,
2. Nonces exchanged in the protocols must not have any bit errors, as they are used to generate keyed filters to protect against active relay attacks. Bit errors can cause the corresponding bitstreams used to generate the keyed filters to not match between the prover and verifier. This will cause the false-positivity rate to increase,
3. Feedback utilized in the protocols, like in the decode-and-forward active relay attack detection protocol, will see more errors and an increased MSE, which can cause the false-positivity rate to increase

In general, bit errors in any step of the proposed protocols can cause the false-positive rate to increase. While some level of bit error is okay for media consumption, cryptographic systems essentially require no bit errors. Given a fixed transmission power, bit error can be reduced through the usage of:

1. Reduction in symbol space (i.e., data rates),
2. Forward error correction (FEC),
3. Diversity,

4. Power optimization.

Given that the symbol space is already small (QPSK) and the requirement that FEC would need additional symbol transmissions (as discussed in Section 9.3), the only options available are the increasing of diversity or power allocation amongst the different OFDM subcarriers. Diversity can be applied through time, frequency, or space. Utilizing time diversity would require additional symbol transmission (i.e., redundancy), which introduces the same problem with the use of FEC. Frequency diversity is already utilized through the usage of OFDM. Additional carrier frequencies are not applicable, due to the limitations of the ISM band, which is also discussed in Section 9.3. The Libre and Pluto SDR can support 2x2 transmission, which would allow for spatial diversity, but the processing capabilities would introduce too large of an overhead. For transmitting, the timing and clock synchronization would add a processing overhead, making transmit diversity prohibitively expensive on these embedded devices. Receiver diversity can utilize timing synchronization of both paths, but it would still require synchronization of both receive paths, which would either double the processing time or would require multi-threading to parallelize the reception of the receiver chains. The Pluto SDR was found to be too slow and to have too little memory to parallelize reception of more than a single receive path. With some optimization, it may be possible, but the feasibility of this was not fully explored in this thesis. The Libre SDR, with roughly twice the compute and memory capabilities of the Pluto SDR, should be able to parallelize the receive paths, but this is left to future work.

The last method, and the method explored in this section of the thesis is the usage of optimization to allocate power to subcarriers. For OFDM, optimal power allocation is a solved problem, where waterfilling can be utilized. With waterfilling, power is allocated to the subcarrier with the highest channel-to-noise ratio (CNR), up to a certain water level λ . The CNR for the m^{th} subcarrier is given by

$$\gamma_m = \frac{|h_{xy}[m]|^2}{N_0}, \quad (9.44)$$

where N_0 is the noise power, and h_{xy} is the channel gain in the m^{th} subcarrier. The CNR can be estimated through the CFR estimate $\hat{\mathbf{h}}_{xy}$ and the noise power estimate $\hat{N}_0$

$$\hat{\gamma}_m = \frac{|\hat{h}_{xy}[m]|^2}{\hat{N}_0}. \quad (9.45)$$

Power is then allocated to the subcarrier with the second highest CNR, and so on. The

optimal water level is given by

$$\lambda^* = \min_{\lambda} \sum_{m=1}^n \log(1 + (\frac{1}{\lambda} - \frac{1}{\hat{\gamma}_m})\hat{\gamma}_m)^+ - \lambda(\sum_{m=1}^n (\frac{1}{\lambda} - \frac{1}{\hat{\gamma}_m})^+ - P_T), \quad (9.46)$$

where P_T is the total transmit power and $(\cdot)^+ : \mathbb{R} \rightarrow \mathbb{R}$ is the maximum of $(\cdot)$ and 0. The power allocated to the m^{th} subcarrier is given by [157]

$$\alpha[m] = \left(\frac{1}{\lambda^*} - \frac{1}{|h_{xy}[m]|^2} \right)^+.$$

Using this gain, the transmitted QPSK symbol at the m^{th} subcarrier is

$$\tilde{s}^*[m] = \sqrt{\alpha[m]} \tilde{s}[m].$$

In general, (9.46) is a difficult water level to solve for, as it requires the minimization of a non-linear function (an NP hard problem). Usually a water level that is faster to calculate is chosen at the expense of a reduced capacity (suboptimal-waterfilling). To minimize the false-positive rate, for the duration of the protocols preventing bit errors should be prioritized, followed by the throughput. To achieve this, subcarriers that have a low CNR should be nulled (adaptive modulation). Adaptive modulation nulls weak subcarriers (i.e., $\alpha_m = 0$), while suboptimal-waterfilling optimizes the subcarriers that are left. To perform these processes, pilots and corresponding channel estimates are used to evaluate the quality of the OFDM subcarriers.

To complicate this process there are three issues that arise:

1. When no relay attack is occurring, the channel between the prover and verifier may not be reciprocal, which can cause them to have different null subcarriers,
2. A decode-and-forward relay may have 2 non-reciprocal channels between it and the prover and verifier, respectively,
3. Any feedback and pilot exchange may be vulnerable to or provide information to an attacker to perform an active relay attack,
4. Feedback cannot be sent over the unreliable channel due to the fading, which can influence what channels are considered to be of good quality .

The first issue requires channel feedback to prevent orthogonal power allocation (i.e., a receiver expects to receive at some subcarrier, but the transmitter is not utilizing that subcarrier). The second problem considers how the relay interacts with the power allocated information. If a decode-and-forward relay does not know the channel states from the prover and verifier (due to non-reciprocity), it cannot perform proper power allocation in any of the directions it is transmitting in. On the other hand, amplify-and-forward relays do not have this problem, as they maintain the power allocation of the original received symbols.

In the third problem security should be considered in the power allocation, otherwise the channel can be revealed to an attacker. An active relay attacker that could observe how power is allocated to channels to try to determine any information about the prover and verifier channels. This could lead to the attacker having some information that could be used to determine a filter. To accommodate these issues, a novel OFDM symmetric-key power allocation protocol and decode-and-forward relay model is proposed.

The last problem requires a backhaul channel with a low error rate, such that communications can occur. The assumption in this problem is that this channel does not exist prior to communications and must be established at the start to share the information between the users.

The steps in the protocol can be seen in Table 9.11 and are detailed in subsequent subsections. The prover and the verifier share a symmetric-key between them prior to the protocol operation.

Table 9.11: Steps of the Waterfilling Protocol

step	Details
1	Prover initiates protocol by transmitting request, $D\{\mathbf{h}_{\mathbf{k}_1}\}[\mathbf{p}, \mathbf{x}_{\text{req}}]$, where a nonce $\mathbf{x}_{\mathbf{h}_{\mathbf{k}_1}}$ is used to generate keyed FIR filter $\mathbf{h}_{\mathbf{k}_1}$
2	Verifier confirms request, then generates a response $D\{\mathbf{h}_{\mathbf{k}_2}\}[\mathbf{p}, \mathbf{x}_{\text{resp}}]$, where nonce $\mathbf{x}_{\mathbf{h}_{\mathbf{k}_2}}$ is used to generate keyed FIR filter $\mathbf{h}_{\mathbf{k}_2}$. The response is transmitted to the prover. Verifier also estimates $\hat{\mathbf{h}}_{vp,0}$ to find power levels.
3	Prover estimates $\hat{\mathbf{h}}_{pv,1}$, assign power levels, then identifies the strongest subcarrier $m_p^* = \max_m h_{pv,1}[m] ^2$. Prover transmits a $\mathbf{x}_{\text{resp}_1}$ at subcarrier single pulse m_p^* to the verifier.
4	Verifier checks the strongest subcarrier $m_v^* = \max_m h_{pv,0}[m] ^2$, and transmits a response $\mathbf{x}_{\text{resp}_2}$ at subcarrier m_v^* to the prover.
5	Prover estimates $\hat{m}_v^*$ from the maximum power of the subcarriers and uses only this subcarrier to transmit an 2^L -QAM symbol. The symbol specifies a power level (bit) for the 1 st L subcarrier.
6	Verifier estimates $\hat{m}_p^*$ from step 4 maximum power of the subcarriers and uses only this subcarrier to transmit an 2^L -QAM symbol. The symbol specifies a power level (bit) for the 1 st L subcarrier.
7	Prover and verifier continue this process $\lceil M/L \rceil$ times to share 1 bit for each subcarrier.
8	Prover transmits to the prover an authentication code on the symbols $\mathbf{x}_{p,\text{MAC}}$ on its transmitted power levels after communications using the newly allocated OFDM system.
9	Verifier transmits to the prover an authentication code on the symbols $\mathbf{x}_{v,\text{MAC}}$ on its transmitted power levels after communications using the newly allocated OFDM system.
10	Verifier checks the MAC. If it passes, then it transmits to the prover an authentication code on the symbols $\mathbf{x}_{v,\text{MAC}}$ on its transmitted power levels after communications using the newly allocated OFDM system. If it fails, then a new waterfilling process must be established by the prover (i.e., go back to step 1).
11	Prover checks the MAC. If it pass, then the prover can begin with which ever protocol or communications it wants to do. If it fails, the prover can decide to repeat step 1.

The result of the first three steps of the protocol are individual channel estimations for the prover and verifier, which depend on the real channels, $\mathbf{h}_{\mathbf{x}\mathbf{y}}$ and the keyed FIR filters $\mathbf{h}_{\mathbf{k}\mathbf{x}}$. The prover and verifier each assign power levels to each subcarrier from their estimated channels, where the 2^L levels are the number of possible power levels (including null power).

The prover and verifier may have different power levels allocated to different subcarriers, either due to channel non-reciprocity or the relay. In either case, the weaker of the two levels are selected, after they share their information in step 7. The only subcarrier revealed is each party's best subcarrier, which is used by the opposing party to transmit their power allocation. Encryption is used for the subsequent subcarriers to prevent any further information leakage. Further information leakage depends on the coherence bandwidth, and the random filters.

An important point about the protocol, is that the keyed filters and challenge filter would be the same ones used in relay detection protocols. As mentioned, these filters can introduce frequency-selectivity to the channels, which can contribute to fades and bit errors. Since these filters are applied to multiple transmissions, including the nonce transmissions used in multiple protocol steps, the channels must be incorporated in the evaluation of channel power levels. If these filters would cause a deep fade for a given subcarrier, the subcarrier should not be used to transmit. The alternative is to perform multiple protocols, where anytime the filter changes a new power allocation is calculated. This is common for systems where communications last longer than the coherence time (i.e., appropriate power allocation changes over time). For the simplicity of the protocols in this thesis, this would be a massive overhead, so it not performed, and only a single waterfilling protocol occurs, after the noise power estimation in the coarse timing synchronization, but before any data transmissions occur from the protocols. The consequence of this is some subcarriers are nulled, when they could otherwise be utilized. For example,

Step 1: Initiation

The prover initiates the protocol by transmitting to the verifier a request $\mathbf{x}_{\text{req}}$, which is a pre-established and publicly known symbol. The prover generates a random nonce $\mathbf{x}_{\mathbf{h}_{\mathbf{k}_1}}$, which is used with the symmetric-key as an IV to generate the keyed FIR filter $\mathbf{h}_{\mathbf{k}_1}$. The keyed FIR filter is used to filter the preamble and OFDM symbol to obscure the channel from any potential attacker. The nonce and original channel are used in subsequent protocol, so they should not be revealed to the verifier in any step in the secure waterfilling protocol. The keyed filter will be used to manage the confidentiality part of the waterfilling process.

Step 2: Response

The verifier receives the request $\hat{\mathbf{x}}_{\text{req}}$ and determines if this is a request to start the waterfilling protocol. Since there may be bit errors, the verifier can look at the Hamming distance between the bits from $\hat{\mathbf{x}}_{\text{req}}$ and the expected $\mathbf{x}_{\text{req}}$. If the error is low, then the verifier can assume a prover is looking to initiate the waterfilling protocol.

The verifier then estimates the channel $\hat{\mathbf{h}}_{vp,0}$. The verifier looks at the CNR each of the $0 \leq m < M$ subcarriers.

$$\hat{\gamma}_h[m] = 10 \log_{10} \left(\frac{P_T |\hat{h}_{vp,0}[m]|^2}{M \hat{N}_0} \right),$$

where $\hat{N}_0$ is established prior to communications (9.5.1). To establish the power allocation, the first minimum threshold $\text{CNR}_{\min}$ is selected. In the hardware experiments the initial threshold was usually selected as $\text{CNR}_{\min} > 10$ dB, to ensure a low chance of bit error for a QPSK symbol (around 10^{-3} BER, with $2Q(\hat{\gamma}_{m_{\text{dB}}}^{-1})$). The subcarriers are then sorted, based off their CNR. The weakest subcarrier is nulled, if $\hat{\gamma}_{m_{\text{dB}}} < \text{CNR}_{\min}$, and no power is allocated to it. After the first minimum subcarrier is identified, the threshold is updated, to take into account the reduced channels. The remaining sorted subcarriers are allocated power using this adaptive threshold. Overall, the subcarriers that are allocated power, get an equal amount of the power. The specific process for subcarrier power allocation can be seen in algorithm 5. An example can be seen in Figure 9.25. The example shows that if the $\text{CNR}_{\min}$ was not adaptive, none of the subcarriers would be allowed, but as the weakest ones are dropped, there is more power to accommodate the subsequent subcarriers. In general, literature has proven that this “constant-power waterfilling” results in a loss of at most 0.266 b/s/Hz in a Rayleigh fading channel [139]. This process of identifying and sharing channels with the constant power allocation requires only a binary feedback per subcarrier (i.e., on or off), which is a equivalent process to adaptive modulation. Given that the capacity is dictated by Shannon’s capacity $\log_{10}(1 + \text{CNR})$ and the logarithmic function grows slowly as the CNR gets large, it is the low CNR values that have a significant impact, while large CNRs do not need to be fine tuned. As the weakest subcarriers are nulled, the CNR increases for the remaining subcarriers, so the threshold should adapt, as shown in the algorithm.

Algorithm 5 Subcarrier Power Allocation Process

```

1:  $\gamma_h \leftarrow \text{Sort}\{D\{\mathbf{h}_{vp,0}\}^* \mathbf{h}_{vp,0}\}$ 
2:  $\gamma_{h_{\text{index}}} \leftarrow \text{Sort}\{D\{\mathbf{h}_{vp,0}\}^* \mathbf{h}_{vp,0}, \text{Indices}\}$ 
3:  $M_{\text{nulls}} \leftarrow 0$ 
4:  $\alpha_v \leftarrow (1, 1, \dots, 1, 1)$ 
5: for  $m$  in  $M$  do
6:   if  $10 \log_{10}(P_T |\gamma_h[m]|^2 / (M \hat{N}_0)) < \text{CNR}_{\min}$  then
7:      $\alpha[\gamma_{h_{\text{index}}}[m]] \leftarrow 0$ 
8:      $M_{\text{nulls}} \leftarrow M_{\text{nulls}} + 1$ 
9:      $\text{CNR}_{\min} \leftarrow \text{CNR}_{\min} + 10 \log_{10}(\frac{M-1}{M})$ 
10:  end if
11: end for
12:
13: return  $\alpha_v$ 

```

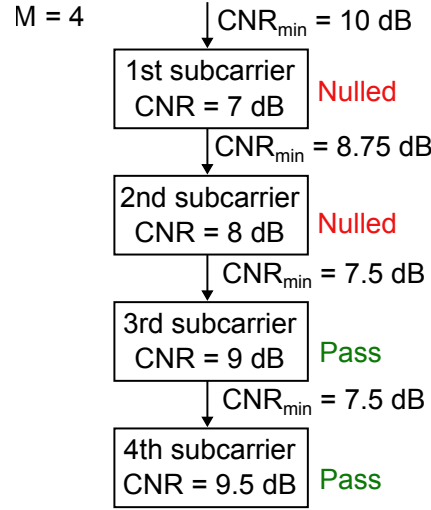


Figure 9.25: Example of how power levels are allocated for $M = 4$ subcarriers and an initial $\text{CNR}_{\min} = 10$ dB

The result of the algorithm is a vector α_v , which contains which subcarriers are nulled or not. After establishing their power allocation, the verifier transmits to the prover a response $\mathbf{x}_{\text{resp}}$, which the prover uses to repeat the same process. The response is filtered by the keyed filter $(\mathbf{h}_{k_2})$, to ensure the verifier accommodates all the possible fades. The response is not filtered by $\mathbf{h}_{\text{ch}}$, as no data is sent when the challenge is sent, and it may

reveal information, if it was incorporated. Like in step 1, the keyed filter is needed to obscure the channel from any potential relay.

Step 3: Prover Best Subcarrier

Similar to step 2 for the verifier, in step 3 the prover estimates the channel $\hat{\mathbf{h}}_{pv,1}$ and uses this to perform power allocation using algorithm 5, building a power allocation vector $\boldsymbol{\alpha}_p$. Once the prover has established its initial power allocation, it will begin sharing this information with the verifier m_p^* and transmits only this pulse and the symbol $\mathbf{x}_{\text{resp}_1}$ at the maximum transmit power P_T to the verifier. The transmission includes the pilots in the preamble and the keyed filter $\mathbf{h}_{\mathbf{k}_1}$, so the verifier can detect the symbol and to obscure the channel from any potential relay, respectively. While fading may exist, this maximum subcarrier is used as the backhaul to communicate from the verifier to the prover, as it is the best subcarrier the prover has to work with at the established bandwidth.

Step 4: Verifier Best Subcarrier

The verifier estimates the best subcarrier by identifying the subcarrier with the largest power (the only power) and the response symbol, as $\hat{m}_p^*$. The verifier repeats this process, by identifying m_v^* and transmitting a response $\mathbf{x}_{\text{resp}_2}$ to the prover with the pilots and $\mathbf{h}_{\mathbf{k}_2}$ on this subcarrier. Similarly, this maximum subcarrier is used as the backhaul to communicate from the prover to the verifier, as it is the best subcarrier the verifier has to work with at the established bandwidth.

Step 5: Prover Initiates Backhaul Channel Feedback

The prover notes the best subcarrier ($\hat{m}_v^*$) in the same way as the verifier, by identifying the subcarrier with the largest power and that has the response symbol from the verifier's transmission. Using this subcarrier, the prover generates an 2^L -QAM symbol, and transmits the symbol at the full transmit power to the verifier. The subcarrier $\hat{m}_v^*$ is the backhaul used from the prover to the verifier, which solves the requirement of needing a low error channel feedback. The 2^L -QAM symbol contains L bits, which designate whether the first L subcarriers in $\boldsymbol{\alpha}_p$ are nulled or not. Again, there is a pilot preamble and keyed filter applied.

Step 6: Verifier Responds on Backhaul Channel Feedback

The verifier demodulates the QAM symbol and responds to the prover using subcarrier $\hat{m}_p^*$, with a corresponding 2^L -QAM symbol, with the binary power allocation for its first L subcarriers in α_v , in the same manner as the prover. The prover's channel feedback is stored in a vector $\hat{\alpha}_p$.

Step 7: Continuing Backhaul Channel Feedback

Both the prover and the verifier continue the backhaul feedback on $\hat{m}_v^*$ and $\hat{m}_p^*$ to build out $\hat{\alpha}_v$ and $\hat{\alpha}_p$, respectively. There is total of $\lceil M/L \rceil$ exchanges, including Steps 4 and 5. After the exchange is complete, both the prover and verifier establish a shared power allocation between them, where only the subcarriers that are sufficient for both are considered. The reason for this is to accommodate a relay, if one existed. This is explored in Section 9.8.

Any future OFDM transmission will place symbols on the shared power allocation subcarriers, and null the weak subcarriers. For the prover, this is equivalent to

$$\hat{\alpha} = \alpha_p \oplus \hat{\alpha}_v.$$

Similarly, the verifier gets

$$\hat{\alpha} = \hat{\alpha}_p \oplus \alpha_v.$$

The power allocation for each subcarrier can be received by multiply $\hat{\alpha}$ with the total power and the number of allowed $P_T/\text{Sum}\{\hat{\alpha}\}$. The same subcarrier is used to simplify the demodulation process. All the prover or verifier needs to do is check the subcarriers identified in $\hat{\alpha}$ to know which symbols to extract.

Step 8: Prover Authentication and Integrity Check

To prevent a relay attack from interfering in this process by changing the bits of α , the prover utilizes the symmetric-key to generate a secure MAC on $bm\alpha_p$. The symbol $\mathbf{x}_{p,\text{MAC}}$ is transmitted utilizing the new waterfilling. This process also acts as check message. If the power allocation was established appropriately (i.e., $\hat{\alpha}$, then the MAC should have no bit errors.

Encryption does not help in providing any confidentiality, as any potential attacker would be able to see the power allocation when the transmissions occur or through the observation of the pilots. Not having encryption does not reveal any more information than

what is revealed through the FIR filters used in the other protocols. In fact, information is lost as the power allocation process is essentially a quantization process, where real numbers ($\mathbf{h}_{xy}$) are converted to single bits.

Step 9: Verifier Authentication and Integrity Check

The verifier checks the received MAC $\hat{\mathbf{x}}_{p,MAC}$ against $\hat{\alpha}_p$ and its symmetric-key. If it passes, then it responds with $\hat{\mathbf{x}}_{v,MAC}$ on $bm\alpha_v$, which the prover will also verify in a similar manner. If there is an error, then a new waterfilling protocol must be established.

Step 10: Prover Verification

The prover checks the received MAC $\hat{\mathbf{x}}_{v,MAC}$ against $\hat{\alpha}_v$ and its symmetric-key. If it passes, then it continues with the challenge-response authentication and relay protocol. If there is an error, then a new waterfilling protocol must be established.

9.8 Relay Attacks

The focus of this chapter, so far, has been on the implementation of an OFDM communication embedded on an SDR, with considerations of the specificity of performing OFDM transmissions in frequency-selective channel, for the purposes of doing the relay detection protocols. While many of the implementation ideas and various algorithms and protocols proposed in this chapter are novel, as they have a specific performance limitation, in general OFDM systems are ubiquitous in our day-to-day lives and an expansive set of literature already exists. Every single 4G, LTE, 5G, and WiFi device utilizes OFDM (billions of devices) and in the IEEE Xplore database there are over 69,000 published results containing the keyword “OFDM.” What is not common in literature and in existing hardware are practical relay attack implementations, whether for OFDM systems or for general communication systems. While the basics of relay attacks are not complicated (just repeat the transmission for a given communication system and protocol), this section looks at what implementation looks like in practice, particularly on an embedded SDR. In this section, consideration is also given to the practicality of implementing a relay in an experimental environment, where devices may not be far enough to assume that they do not detect the corresponding legitimate devices transmission. While this defeats the purpose of a relay attack, it is likely researchers must work in a small and isolated environment (e.g., office, lab,

anechoic chamber etc.), where it is difficult to prevent the interference between legitimate devices, without significantly reducing the SNR. An implementation must consider these conditions, which is also a novel consideration in this thesis. An example of this problem can be seen in Figure 9.26. In this example in a relatively large room with the prover and

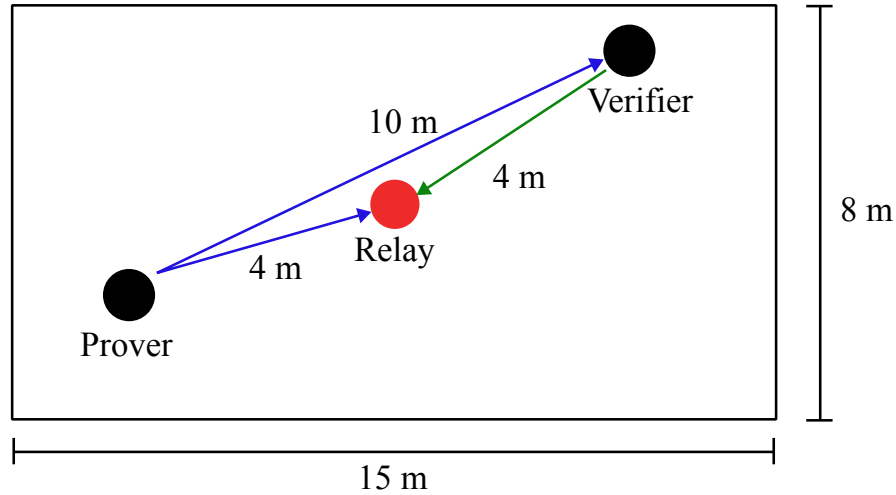


Figure 9.26: Example of relay interference between legitimate communication parties in a room

verifier 10 m away from each other and a relay in between, the prover and verifier will certainly detect each others transmissions. Trying to increase the distance between the prover and verifier will reduce the SNR in one of or both of the path ways will decrease due to this increased distance.

To allow the use of relaying in a controllable environment, the only option is to force the relaying. Assuming that the prover, verifier, and relay can detect every transmission from the other parties, this can happen by having the verifier reject any transmission from the prover and the prover reject any transmission from the verifier. By only accepting transmission from the relay, the devices can be at arbitrary distances from each and only the relay channel needs to be evaluated or controlled. There are 2 ways identified to allow this to happen. The first is to utilize an ID for the devices. After symbol mapping, a receiver can check the ID and reject IDs that are not from the relay. The second option is to map out whatever protocol is occurring, and to have the prover and verifier respectively ignore certain steps in the protocol associated with the other party.

As detailed in Section 2.2, there are two types of relay attacks, the decode-and-forward and amplify-and-forward relay attacks. Both these attacks require different considerations

when performing the relay. In particular, the decode-and-forward relay attack requires the demodulation of the system, while the amplify-and-forward system does not. To implement all of the protocols presented in this thesis, a round-trip must be completed within a coherence time interval, which establishes a hard upper limit on the time that the relay can introduce into the communication system. The upper bound on the RTT is given by the coherence time, which for a terrestrial system on a microwave carrier frequency is limited to

$$\begin{aligned} T_{\text{RTT}} &< \frac{v}{f_c} \cos(\theta_{\text{AoA}}) \\ &\leq \frac{100 \times 1000 / 60^2}{1 \times 10^6} \\ &= 27 \text{ ms}, \end{aligned}$$

where T_{RTT} is the total RTT, and a terrestrial speed of less than 100 km/hr is considered, at a frequency of 1 GHz. In general, the coherence time will vary within 10-100 ms as a target for relaying.

Considering the benchmarking shown in Figure 9.17, for a receiver the coarse timing synchronization process requires around 1 ms for a buffersize $Q > 8192$. When considering all the other processes shown in Figure 9.10, adding an ID system to determine which transmission to ignore becomes too cumbersome and risks causing the RTT to be larger than the coherence time. Furthermore, at the minimum, the ID will require $\lceil \log_2(3) \rceil = 2$ bits in every transmission. If there are K relays the ID system becomes even more significant, requiring $\lceil \log_2(K + 2) \rceil$ bits. On the other hand, a state machine for each protocol, where the prover and verifier ignore transmissions K transmission and a relay ignores $K - 1$ transmissions, before they process a received transmission. This method is how all relay processing is “hardcoded” in the hardware implementation and the recommended method for experimental relaying.

9.8.1 Amplify-and-Forward Relay Attack

The implementation of an amplify-and-forward relay can be simplified compared to a decode-and-forward relay. The relay is only relaying the analog signal. For the SDR this can be implemented by quickly by immediately transferring the content of the receiver buffers to transmitter buffer in the Pluto or Libre SDR. The issue with this process is all the noise gets transferred in the surrounding samples as well, which can allow it to be easier for a verifier to determine if a relay attack is occurring. The alternative is for

the relay to utilize the timing synchronization processes to narrow the specific signal to relay. Specifically, relaying can happen after the coarse timing synchronization or after the fine timing synchronization. The further in the synchronization process, the less noise is relayed, but more time is required to perform the relaying. Furthermore, extra processing time is required to convert from the 16-bit sample receiver buffers to the 12-bit I/Q samples and back to the 16-bit sample buffers for transmission. For the implementation, the fine synchronization was used for quicker operability for converting the decode-and-forward relay into an amplify-and-forward relay.

9.8.2 Relay Waterfilling

One of the considerations is how a decode-and-forward relay can embed itself within the waterfilling protocol in Section 9.7, and how that might influence how it performs. It is assumed that the relay attacker wants the communications to occur without error, otherwise the challenge-response will deny access if any error occurs. Given this assumption, when waterfilling occurs so the prover and verifier can establish power allocation, the relay must incorporate its channels into the feedback in Steps 4-7 in Table 9.11. The relay must also act as intermediary for the back-haul to happen without error.

In general, if a relay is acting as a decode-and-forward relay and they do not participate in the waterfilling protocol, then it is possible that errors will appear on the channel from the prover to the relay and from the verifier to the relay, if the channels are not reciprocal. Furthermore, the relay may not re-transmit with the appropriate power allocations. For the subcarriers that passed as a result of the reduction in the threshold, as shown in algorithm 5, this might result in bit errors, as power is allocated to bad subcarriers transmitting useless information.

Chapter 10

Conclusion and Future Work

10.1 Conclusion

The relay attack is a simple, but powerful tool for an attacker to circumvent challenge-response authentication for wireless cyber-physical systems. The relay attack cannot be solved by typical cryptographic methods, which motivates the search for alternative physical-layer based solutions. While several solutions are currently proposed in academia and industry, those solutions come with trade-offs in terms of cost and performance, which has prevented their mass adoption. This thesis proposed the use of the wireless channel as a base for detecting the different types of relay attacks. Specifically, four novel protocols that utilize different aspects of the communication channel to detect decode-and-forward and amplify-and-forward relay attacks, separately. The first proposed protocol leverages channel reciprocity to detect a decode-and-forward relay attack. The second proposed protocol generalizes detection to non-reciprocal channels to detect decode-and-forward relay attacks under stronger adversarial conditions. The third proposed protocol utilizes the change in distribution of a relayed channel as a feature to detect amplify-and-forward relay attacks. The fourth proposed protocol utilizes a deep neural network to generalize feature selection to detect an amplify-and-forward relay attack. Through the exploration of the motivation, theory, simulation results, and hardware results for the four channel based solutions to the relay attack, this thesis found that they are all sufficient at detecting relay attacks, in most scenarios.

Chapters 1, 2, and 3 focused on the introduction of the relay attack problem, the background on the details of the relay attack, and an overview of the current literature on relay attacks. These chapters acted as the motivation of how the wireless channel could be

used to detect relay attacks. The specific definitions of the relay attack presented in these chapters has not been done in other pieces of literature.

Chapters 4, 5, 6, 7, and 8 presented the novel protocols that utilize the channel to detect the relay. The background, theoretical analysis, and simulation results were presented each of the protocols, with hardware results presented in chapter 6 and 8 on software-defined radios. Near field and far field results were also distinguished.

Chapter 9 presented the details on the novel hardware implementation on software-defined radios. This includes a reliable OFDM communication system and the relay implementation details.

The specific contributions of the thesis are the following:

1. An overview of the different forms of relay attacks and the general approaches to deal with them,
2. Proposal of four novel channel based relay detections protocols for detecting when relay attacks occur considering under decode-and-forward, amplify-and-forward, passive, and active relay attacks,
3. Experimentation that shows the viability of channel based relay detection for near field relay detection,
4. Detailing of the first open-source hardware implementation of amplify-and-forward relay attacks and decode-and-forward relay attacks on software-defined radio,
5. The first hardware implementation of relay attack detection on software-defined protocols using the wireless channel,
6. Software simulations of the proposed protocols in various channel conditions,
7. Experimental evaluations of the proposed relay attacks, showing how they perform in software and hardware,
8. Theoretical evaluations of the proposed protocols, showing how they provide security, and how the security may change under different channel conditions.

10.2 Future Work

While many of the solutions have sound theoretical motivation (which was corroborated by the hardware experiments with software-defined radios and simulation experiments in software), the decode-and-forward detection protocols and the non-neural net amplify-and-forward detection protocol likely have closed expressions for the different performance criterion that were evaluated for in these experiments. For example, the false-positive and false-negative rate for the decode-and-forward protocols likely have a statistical relation between the mean square error, channel's distribution, and the Gaussian noise, which should allow for these expressions to be derived. Finding these expressions and comparing them with the simulation and hardware results could further validate the results gathered in this thesis. Having a theoretical expression, simulation results, and hardware results, and having them all corroborate each other would be ideal for the design and proposal of a physical-layer protocol.

In this thesis a solution for protecting against an active near field relay attack was not presented. While a decode-and-forward relay attack detection protocol was presented in Chapter 8, the protocol was only designed to handle passive attacks and could easily be circumvented by an attacker that could accommodate the fixed channel. Future work should explore if it is possible to design a protocol that can utilize channel state information to detect or prevent an active near field relay attack. At the moment, the near field and active relay attack appears to be the major limitation in channel based relay attack detection.

Another consideration alluded to in this thesis, but not explored in detail, is the relationship between the different channels, and how an attacker could potentially leverage the environment to perform active relay attacks. Specifically, given a distribution of two different channels (e.g., from the prover to relay and from the relay to verifier), is there a relationship between how the channels correlate and can an expression be found relating them? Based on how these channels may correlate, from an information theoretic perspective what mutual information can an attacker extract and potentially use in their active relay attack? Similarly, experiments found the reciprocity was not guaranteed, particularly with the full-duplex system found in most software-defined radios. Given these differences (particularly the distance), future work could look at how the spatial correlation be used to model how the channels relate in simulations and in the derivations.

In general, there is a lot of research potential in evaluating the joint distributions, autocorrelation, and cross-correlation between different channels in frequency and time for decode-and-forward and amplify-and-forward channels. This could possibly yield better

protocols for evaluating if a relay attack is occurring. While the usage of a deep neural network was proposed to abstract specific feature selection, understanding the channel models, particularly for cascaded amplify-and-forward channel, could lead to the utilization of other transforms, as opposed to the discrete Fourier transform. For example, the generalized Zak transform, for observation in the delay-Doppler domain, is hypothesized as an alternative transform for evaluating relay attacks.

Future work should also focus on expanding the hardware experiments. The experiments were performed in limited environments (e.g., indoor, short distance, etc.). One of the important distinctions of the proposed protocols compared to alternative methods, like ultra-wideband distance-bounding, is their theoretical ability to detect long distance relay attacks. Performing more experiments at larger distances and in situations where the speed is faster should be done to corroborate the theory and simulations. Furthermore, hardware experiments were not completed for the neural network amplify-and-forward detection protocol. It is likely that more work will be required to validate the results on hardware. This will include performing software optimization so the model can run directly on an SDR, which has limited memory and compute capabilities.

Another area of future work is in the tangential area of the application of relay attacks. Particularly, how they interact with other forms of physical-layer security and in cellular communication systems. Physical-layer key generation and identification have been proposed as the main applications of physical-layer security. It is hypothesized that amplify-and-forward and active decode-and-forward relay attacks could totally circumvent these systems. Terrorist fraud relay attacks may also have a major impact in cellular communication systems. For example, they could be used to manipulate the location of device by altering via a decode-and-forward or quantize-and-forward relaying, similar to a virtual private network.

References

- [1] J.H. Conway, “On Numbers and Games”, CRC Press, Dec. 1976
- [2] Y. Desmedt, C. Goutier and S. Bengio, “Special Uses and Abuses of the Fiat-Shamir Passport Protocol,” in *Advances in Cryptology — CRYPTO ’87*, pp. 21–39, Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-48184-2_3
- [3] U. Feige, A. Fiat, and A. Shamir, “Zero-knowledge proofs of identity,” *J. Cryptology* 1, pp. 77–94 (1988), <https://doi.org/10.1007/BF02351717>
- [4] S. Bengio, G. Brassard, Y. Desmedt, C. Goutier and J. Quisquater, “Secure implementation of identification systems,” in *Journal of Cryptology*, 4(3), pp. 175–183. <https://doi.org/10.1007/BF00196726>
- [5] S. Brands and D. Chaum, “Distance-bounding Protocols”, Workshop on the Theory and Application of Cryptographic Techniques on *Advances in Cryptology, EURO-CRYPT ’93*, Lofthus, Norway, 1993
- [6] A. Levi, E. Çetintaş, M. Aydos and C. Koç, M. Çağlayan, “Relay Attacks on Bluetooth Authentication and Solutions,” in Aykanat, C., Dayar, T., Körpeoğlu, İ. (eds) *Computer and Information Sciences - ISCIS 2004. ISCIS 2004. Lecture Notes in Computer Science*, vol. 3280. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-30182-0_29
- [7] G. P. Hancke and M. G. Kuhn, “An RFID Distance Bounding Protocol,” *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM’05)*, 2005, pp. 67-73, doi: 10.1109/SECURECOMM.2005.56.
- [8] G. P. Hancke, “Distance-bounding for RFID: Effectiveness of ‘terrorist fraud’ in the presence of bit errors,” *2012 IEEE International Conference on RFID-Technologies and Applications (RFID-TA)*, 2012, pp. 91-96, doi: 10.1109/RFID-TA.2012.6404574.

- [9] Bussard and W. Bagga, W. (n.d.). “Distance-Bounding Proof of Knowledge to Avoid Real-Time Attacks,” *Security and Privacy in the Age of Ubiquitous Computing*, pp. 223–238. https://doi.org/10.1007/0-387-25660-1_15
- [10] A. Francillon, B. Danev and S. Capkun, “Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars,” *Cryptology ePrint Archive*, Report 2010/332, 2010.
- [11] L. Francis, G. P. Hancke, K. Mayes and K. Markantonakis, “Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones,” *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6370, pp. 35–49. https://doi.org/10.1007/978-3-642-16822-2_4
- [12] G. P. Hancke, “Practical attacks on proximity identification systems,” 2006 IEEE Symposium on Security and Privacy (S& P’06), 2006, pp. 6 pp.-333, doi: 10.1109/SP.2006.30.
- [13] J. Cox, “Meet the Guy Selling Wireless Tech to Steal Luxury Cars in Seconds,” *Vice News, Motherboard Tech By Vice*, Feb. 11. 2020, Accessed: Nov. 17 2022, <https://www.vice.com/en/article/7kz48x/guy-selling-relay-attack-keyless-repeaters-to-steal-cars>
- [14] C. Boyd, “Car owners warned of another theft-enabling relay attack,” *Malwarebytes Labs*, May 17 2022, Accessed: Nov. 17 2022, <https://www.malwarebytes.com/blog/news/2022/05/car-owners-warned-of-another-theft-enabling-relay-attack>
- [15] S. Gallichan-Lowe, “CCTV video shows suspects using electronic method to steal cars in northeast Toronto,” *CityNews*, July 14 2020, Accessed: Nov. 17 2022, <https://toronto.citynews.ca/2020/07/14/toronto-northeast-vehicle-electronic-thefts/>
- [16] D. Cavdar and E. Tomur, “A practical NFC relay attack on mobile devices using card emulation mode,” 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015, pp. 1308-1312, doi: 10.1109/MIPRO.2015.7160477.
- [17] S. Oh, T. Doo, T. Ko, J. Kwak and M. Hong, “Countermeasure of NFC relay attack with jamming,” 2015 12th International Conference & Expo on Emerging Technologies for a Smarter World (CEWIT), 2015, pp. 1-4, doi: 10.1109/CEWIT.2015.7338165.

- [18] C. Thorpe, J. Tobin and L. Murphy, “An ISO/IEC 7816-4 Application Layer Approach to Mitigate Relay Attacks on near-field Communication,” in *IEEE Access*, vol. 8, pp. 190108-190117, 2020, doi: 10.1109/ACCESS.2020.3031979.
- [19] E. Lee, “NFC hacking: The easy way,” in *Proc. DEF CON*, Las Vegas, NV, USA, 2012, pp. 1–29.
- [20] Munilla and A. Peinado, “Distance bounding protocols for RFID enhanced by using void-challenges and analysis in noisy channels,” *Wireless Communications and Mobile Computing*, 8(9), pp. 1227–1232. <https://doi.org/10.1002/wcm.590>
- [21] I. Boureanu and S. Vaudenay, “Challenges in Distance Bounding,” in *IEEE Security & Privacy*, vol. 13, no. 1, pp. 41-48, Jan.-Feb. 2015, doi: 10.1109/MSP.2015.2.
- [22] Avoine, M. Bingöl, I. Boureanu, S. Ćapkun, G.P Hancke, S. Kardaş, C. Kim, C. Lauradoux, B. Martin, J. Munilla, A. Peinado, K. Rasmussen, D. Singelée, A. Tchamkerten and R. Trujillo-Rasua and S. Vaudenay, “Security of Distance-Bounding: A Survey, ” *ACM Computing Surveys*, 2018, vol.51(5), pp. 1–33. <https://doi.org/10.1145/3264628>
- [23] J. Clulow, G.P. Hancke, M.G. Kuhn, T. Moore, “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.
- [24] Wei, H. Zhang and Y. Wang, “A New Relay Attack on Distance Bounding Protocols and Its Solution with Time-Stamped Authentication for RFID,” *Wuhan University Journal of Natural Sciences*, 2016, vol. 21, no.1, pp. 37–46. <https://doi.org/10.1007/s11859-016-1136-y>
- [25] G. Kim, K. Lee, S. Kim and J. Kim, “Vehicle Relay Attack Avoidance Methods Using RF Signal Strength”, *Communications and Network*, 2013, vol.5, pp. 573-577.
- [26] C. Wonsuk, S. Minhye and D. Hoon Lee, “Sound-Proximity: 2-Factor Authentication against Relay Attack on Passive Keyless Entry and Start System”, in *Journal of Advanced Transportation*, 2018. <https://doi.org/10.1155/2018/1935974>
- [27] M. Lenhart, M. Spanghero and P. Papadimitratos, “Relay/replay attacks on GNSS signals,” in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2021, pp. 380-382

- [28] U. Ahmad, H. Song, A. Bila, M. Alazab and A. Jolfaei, "Securing smart vehicles from relay attacks using machine learning," *J. Supercomput.*, vol. 76, pp. 2665–2682, 2020, doi: 10.1007/s11227-019-03049-4
- [29] Francis, K. Mayes, G.P. Hancke and K. Markantonakis, "A location based security framework for authenticating mobile phones," 2010 Proceedings of the 2nd International Workshop on Middleware for Pervasive Mobile and Embedded Computing, pp. 1–8. <https://doi.org/10.1145/1890791.1890796>
- [30] T. Halevi, H. Li, D. Ma, N. Saxena, J. Voris and T. Xiang, "Context-Aware Defenses to RFID Unauthorized Reading and Relay Attacks," in *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 2, pp. 307-318, Dec. 2013, doi: 10.1109/TETC.2013.2290537.
- [31] D. Ma, N. Saxena, T. Xiang and Y. Zhu, "Location-Aware and Safer Cards: Enhancing RFID Security and Privacy via Location Sensing," in *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 2, pp. 57-69, March-April 2013, doi: 10.1109/TDSC.2012.89.
- [32] Y. Tu and S. Piramuthu, "On addressing RFID/NFC-based relay attacks: An overview," *Decision Support Systems*, vol. 129, 113194–. <https://doi.org/10.1016/j.dss.2019.11319>
- [33] J. Li, Y. Dong, S. Fang, H. Zhang and D. Xu, "User Context Detection for Relay Attack Resistance in Passive Keyless Entry and Start System," *Sensors (Basel)*, 2020, vol. 20, no. 16:4446. doi: 10.3390/s20164446.
- [34] B. Shrestha, N. Saxena, H. T. T. Truong and N. Asokan, "Sensor-Based Proximity Detection in the Face of Active Adversaries," in *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 444-457, 1 Feb. 2019, doi: 10.1109/TMC.2018.2839604.
- [35] Ji, Xia, L., Lin, J., Wang, Q., Lei, L., and Song, L. (2019). "Chord: Thwarting Relay Attacks Among near field Communications," In *Information Security and Cryptology* (pp. 329–348). Springer International Publishing. https://doi.org/10.1007/978-3-030-14234-6_18
- [36] H. T. Thu Truong, J. Toivonen, T. D. Nguyen, C. Soriente, S. Tarkoma and N. Asokan, "DoubleEcho: Mitigating Context-Manipulation Attacks in Copresence Verification," 2019 IEEE International Conference on Pervasive Computing and Communications (PerCom, 2019, pp. 1-9, doi: 10.1109/PERCOM.2019.8767404.

- [37] I. Gurulian, G. P. Hancke, K. Markantonakis, and R. N. Akram, "May the Force Be with You: Force-Based Relay Attack Detection," in Smart Card Research and Advanced Applications. CARDIS 2017. Lecture Notes in Computer Science, vol. 10728, T. Eisenbarth and Y. Tegliala, Eds. Cham: Springer, 2018, pp. 9–26. doi: 10.1007/978-3-319-75208-2_9.
- [38] T. Halevi, H. Li, D. Ma, N. Saxena, J. Voris and T. Xiang, "Context-Aware Defenses to RFID Unauthorized Reading and Relay Attacks," in IEEE Transactions on Emerging Topics in Computing, vol. 1, no. 2, pp. 307-318, Dec. 2013, doi: 10.1109/TETC.2013.22905
- [39] M. Mohamed, B. Shrestha and N. Saxena, "SMASheD: Sniffing and Manipulating Android Sensor Data for Offensive Purposes," in IEEE Transactions on Information Forensics and Security, vol. 12, no. 4, pp. 901-913, April 2017, doi: 10.1109/TIFS.2016.2620278.
- [40] J. Vila and R. J. Rodríguez, "Practical Experiences on NFC Relay Attacks with Android: Virtual Pickpocketing Revisited," RFIDSec 2015, pp. 87–103. https://doi.org/10.1007/978-3-319-24837-0_6
- [41] F. Stajano, FL. Wong and B. Christianson, "Multichannel Protocols to Prevent Relay Attacks," *Financial Cryptography and Data Security 2010*, 2010, vol. 6052, pp.4-19, Springer, Berlin, Heidelberg, February 2010.
- [42] J. Wallace, "Secure physical-layer Key Generation Schemes: Performance and Information Theoretic Limits," 2009 IEEE International Conference on Communications, 2009, pp. 1-5, doi: 10.1109/ICC.2009.5199440.
- [43] I. Csiszar and J. Korner, "Broadcast channels with confidential messages," in IEEE Transactions on Information Theory, vol. 24, no. 3, pp. 339-348, May 1978, doi: 10.1109/TIT.1978.1055892.
- [44] I. Csiszar and J. Korner, "Broadcast channels with confidential messages," in IEEE Transactions on Information Theory, vol. 24, no. 3, pp. 339-348, May 1978, doi: 10.1109/TIT.1978.1055892.
- [45] U. M. Maurer, "Secret key agreement by public discussion from common information," in IEEE Transactions on Information Theory, vol. 39, no. 3, pp. 733-742, May 1993, doi: 10.1109/18.256484.

- [46] A. D. Wyner, "The wire-tap channel," in *The Bell System Technical Journal*, vol. 54, no. 8, pp. 1355-1387, Oct. 1975, doi: 10.1002/j.1538-7305.1975.tb02040.x
- [47] A. Goldsmith, "Path Loss, Shadowing, and Multipath" in *Wireless Communications*, Cambridge University Press, Aug. 2005, ch. 2. Accessed on: Oct. 19, 2024. [Online]. Available: http://web.stanford.edu/class/ee359/doc/WirelessComm_Ch1-16_March32020.pdf
- [48] A. Goldsmith, "Statistical Multipath Channel Models" in *Wireless Communications*, Cambridge University Press, Aug. 2005, ch. 3. Accessed on: Oct. 19, 2024. [Online]. Available: http://web.stanford.edu/class/ee359/doc/WirelessComm_Ch1-16_March32020.pdf
- [49] M. Fliess, C. Join, and H. Sira-Ramirez, "Non-linear estimation is easy," *Int. J. Model., Ident. Control*, vol. 4, no. 1, pp. 12-27, 2008.
- [50] J. Zhang, M. Ding, G. Li and A. Marshall, "Key Generation Based on Large Scale Fading," in *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8222-8226, Aug. 2019, doi: 10.1109/TVT.2019.2922443.
- [51] J. Zhang et al., "Experimental Study on Key Generation for physical-layer Security in Wireless Communications," in *IEEE Access*, vol. 4, pp. 4464-4477, 2016, doi: 10.1109/ACCESS.2016.2604618.
- [52] S. N. Premnath et al., "Secret Key Extraction from Wireless Signal Strength in Real Environments," in *IEEE Transactions on Mobile Computing*, vol. 12, no. 5, pp. 917-930, May 2013, doi: 10.1109/TMC.2012.63.
- [53] S. Jana, S. N. Premnath, M. Clark, S. K. Kasera, N. Patwari and S. V. Krishnamurthy, "On the effectiveness of secret key extraction from wireless signal strength in Real Environments," *Proceedings of the 15th annual international conference on Mobile computing and networking - MobiCom '09*, 2009.
- [54] S. Mathur, W. Trappe, N. Mandayam, C. Ye and A. Reznik "Radio-Telepathy: Extracting a secret key from an unauthenticated wireless channel," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw. (MobiCom '08)*, 2008, pp. 128-139, doi: 10.1145/1409944.1409960.
- [55] T. Aono, K. Higuchi, T. Ohira, B. Komiyama and H. Sasaoka, "Wireless secret key generation exploiting reactance-domain scalar response of multipath fading channels," in *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 11, pp. 3776-3784, Nov. 2005, doi: 10.1109/TAP.2005.858853

- [56] Y. Liu, S. C. Draper and A. M. Sayeed, "Exploiting Channel Diversity in Secret Key Generation From Multipath Fading Randomness," in *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 5, pp. 1484-1497, Oct. 2012, doi: 10.1109/TIFS.2012.2206385.
- [57] H. Liu, Y. Wang, J. Yang and Y. Chen, "Fast and practical secret key extraction by exploiting channel response," 2013 Proceedings IEEE INFOCOM, 2013, pp. 3048-3056, doi: 10.1109/INFOCOM.2013.6567117.
- [58] K. Zeng, D. Wu, A. Chan and P. Mohapatra, "Exploiting Multiple-Antenna Diversity for Shared Secret Key Generation in Wireless Networks," 2010 Proceedings IEEE INFOCOM, 2010, pp. 1-9, doi: 10.1109/INFOCOM.2010.5462004.
- [59] S. Gollakota and D. Katabi, "physical-layer wireless security made fast and channel independent," 2011 Proceedings IEEE INFOCOM, 2011, pp. 1125-1133, doi: 10.1109/INFOCOM.2011.5934889.
- [60] N. Aldaghri and H. MahdaviFar, "physical-layer Secret Key Generation in Static Environments," in *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2692-2705, 2020, doi: 10.1109/TIFS.2020.2974621.
- [61] G. Li, H. Yang, J. Zhang, H. Liu and A. Hu, "Fast and Secure Key Generation with Channel Obfuscation in Slowly Varying Environments," IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, 2022, pp. 1-10, doi: 10.1109/INFOCOM48880.2022.9796694.
- [62] P. Staat, H. Elders-Boll, M. Heinrichs, R. Kronberger, C. Zenger and C. Paar, "Intelligent Reflecting Surface-Assisted Wireless Key Generation for Low-Entropy Environments," 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2021, pp. 745-751, doi: 10.1109/PIMRC50174.2021.9569556.
- [63] G. Li, A. Hu, J. Zhang, L. Peng, C. Sun and D. Cao, "High-Agreement Uncorrelated Secret Key Generation Based on Principal Component Analysis Preprocessing," in *IEEE Transactions on Communications*, vol. 66, no. 7, pp. 3022-3034, July 2018, doi: 10.1109/TCOMM.2018.2814607.
- [64] N. Patwari, J. Croft, S. Jana and S. K. Kasera, "High-Rate Uncorrelated Bit Extraction for Shared Secret Key Generation from Channel Measurements," in *IEEE Transactions on Mobile Computing*, vol. 9, no. 1, pp. 17-30, Jan. 2010, doi: 10.1109/TMC.2009.88.

- [65] G. Brassard and L. Salvail, “Secret-Key Reconciliation by Public Discussion,” in *Advances in Cryptology — EUROCRYPT ’93. EUROCRYPT 1993. Lecture Notes in Computer Science*, vol. 765. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48285-7_35
- [66] C. H. Bennett, G. Brassard, C. Crepeau and U. M. Maurer, “Generalized privacy amplification,” in *IEEE Transactions on Information Theory*, vol. 41, no. 6, pp. 1915-1923, Nov. 1995, doi: 10.1109/18.476316.
- [67] H. M. Furqan, J. M. Hamamreh and H. Arslan, “New physical-layer Key Generation Dimensions: Subcarrier Indices/Positions-Based Key Generation,” in *IEEE Communications Letters*, vol. 25, no. 1, pp. 59-63, Jan. 2021, doi: 10.1109/LCOMM.2020.3025262.
- [68] 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>
- [69] R. Abubaker, “Channel based relay attack detection protocol,” University of Waterloo, 2019
- [70] C. T. Zenger, M. Pietersz and C. Paar, “Preventing relay attacks and providing perfect forward secrecy using PHYSEC on 8-bit μ C,” 2016 IEEE International Conference on Communications Workshops (ICC), 2016, pp. 110-115, doi: 10.1109/ICCW.2016.7503773.
- [71] S. Jain and J. S. Baras, “Preventing wormhole attacks using physical-layer authentication,” 2012 IEEE Wireless Communications and Networking Conference (WCNC), 2012, pp. 2712-2717, doi: 10.1109/WCNC.2012.6214260.
- [72] P. L. Yu, J. S. Baras and B. M. Sadler, “Physical-Layer Authentication,” in *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 38-51, March 2008, doi: 10.1109/TIFS.2007.916273.
- [73] P. Staat, K. Jansen, C. Zenger and C. Paar, “Securing Phone as a Key Against Relay Attacks,” 18th escar Europe, 2020, pp.48-58, doi: 10.13154/294-7546
- [74] P. Staat, K. Jansen, C. Zenger, H. Elders-Boll and C. Paar. “Analog Physical-Layer Relay Attacks with Application to Bluetooth and Phase-Based Ranging,” 2022 Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile

Networks (WiSec '22). Association for Computing Machinery, New York, NY, USA, 60–72. <https://doi.org/10.1145/3507657.3528536>

- [75] Bluetooth SIG. Bluetooth Core Specification Revision v5.3. 2021. pp. 2874, Accessed: Nov. 11, 2022. [Online]. Available: https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=521059
- [76] B. Danev, T. S. Heydt-Benjamin and S. Čapkun. 2009. “Physical-layer identification of RFID devices,” In Proceedings of the 18th conference on USENIX security symposium (SSYM’09). USENIX Association, USA, 199–214.
- [77] S. Chinnappa Gounder Periaswamy, D. R. Thompson and J. Di, “Fingerprinting RFID Tags,” in IEEE Transactions on Dependable and Secure Computing, vol. 8, no. 6, pp. 938–943, Nov.-Dec. 2011, doi: 10.1109/TDSC.2010.56.
- [78] J. Li, A. Li, D. Han, Y. Zhang, T. Li and Y. Zhang, “RCID: Fingerprinting Passive RFID Tags via Wideband Backscatter,” IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, 2022, pp. 700–709, doi: 10.1109/INFOCOM48880.2022.9796663.
- [79] D. Zanetti, P. Sachs and S. Capkun, “On the practicality of uhf rfid fingerprinting: How real is the rfid tracking problem?” in Privacy Enhancing Technologies, Waterloo, Canada, July 2011.
- [80] K. Bonne Rasmussen and S. Capkun, “Implications of radio fingerprinting on the security of sensor networks,” 2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm 2007, 2007, pp. 331–340, doi: 10.1109/SECCOM.2007.4550352.
- [81] J. Hall, M. Barbeau and E. Kranakis, “Detecting rogue devices in bluetooth networks using radio frequency fingerprinting,” Proceedings of the Third IASTED International Conference on Communications and Computer Networks, 2006. pp 108–113.
- [82] É. Helluy-Lafont, A. Boé, G. Grimaud and M. Hauspie, “Bluetooth devices fingerprinting using low cost SDR,” 2020 Fifth International Conference on Fog and Mobile Edge
- [83] H. Givehchian et al., “Evaluating Physical-Layer BLE Location Tracking Attacks on Mobile Devices,” 2022 IEEE Symposium on Security and Privacy (SP), 2022, pp. 1690–1704, doi: 10.1109/SP46214.2022.9833758.

- [84] Rappaport, T.T., Blankenship, K.T. and Xu, H. (1997), "Propagation and radio system design issues in mobile radio systems for the glomo project," 1997.
- [85] R. Gaetz, "ADALM-PLUTO Overview," Oct. 22 2021, Accessed: Nov. 15 2022 <https://wiki.analog.com/university/tools/pluto>
- [86] Analog Devices, "ADALM-PLUTO Software-Defined Radio Active Learning Module," Accessed: Nov. 15 2022 <https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html#eb-overview>
- [87] Analog Devices, "AD9363 RF Agile Transceiver," Accessed: Nov. 16 2022 <https://www.analog.com/en/products/AD9363.html#product-overview>
- [88] D. Huang, A. Al-Hourani, K. Sithamparanathan, W. S. T. Rowe, L. Bulot and A. Thompson, "Deep Learning Methods for Device Authentication Using RF Fingerprinting," 2021 15th International Conference on Signal Processing and Communication Systems (ICSPCS), 2021, pp. 1-7, doi: 10.1109/ICSPCS53099.2021.9660226.
- [89] B. Razavi, "RF Microelectronics" in *Wireless Communications*, Prentice Hall, Aug. 2005, ch. 2.3 -Noise
- [90] J. Woodward, "New tricks leading to alarming rate of car thefts in Toronto," CTV News Toronto, Jan. 20, 2023. Accessed: Nov. 4, 2024. [Online]. Available: <https://toronto.ctvnews.ca/new-tricks-leading-to-alarming-rate-of-car-thefts-in-toronto-1.6234708>
- [91] D. Common, "Canada grapples with surge in auto thefts as stolen vehicles flow overseas," CBC News, Aug. 26, 2023. Accessed: Nov. 4, 2024. [Online]. Available: <https://www.cbc.ca/news/world/auto-theft-canada-1.6953242>
- [92] S. Gallichan-Lowe, "Toronto police warn of spike in electronic vehicle thefts in city's northeast," CityNews Toronto, Jul. 14, 2020. Accessed: Nov. 4, 2024. [Online]. Available: <https://toronto.citynews.ca/2020/07/14/toronto-northeast-vehicle-electronic-thefts>
- [93] CAA, "How to Prevent Keyless Car Theft," Canada Automobile Association North & East Ontario, Mar. 6, 2023. [Online]. Available: <https://caaneo.ca/blog/automotive/how-to-prevent-keyless-car-theft>

- [94] B. Chappell, “Chess player Hans Niemann settles lawsuit over cheating allegations,” NPR, Sep. 26, 2023. Accessed: Nov. 4, 2024. [Online]. Available: <https://www.npr.org/2023/09/26/1201734274/chess-hans-niemann-anal-beads-cheating>
- [95] L. Chen and G. Gong, *Communication System Security*, CRC Press, 2012, ch. 6.2 -Authentication Server
- [96] T. Bocek, C. Killer, C. Tsiaras and B. Stiller , “An NFC Relay Attack with Off-the-shelf Hardware and Software,” 2016 IFIP Autonomous Infrastructure, Management and Security (AIMS), 2016, pp. 71-83. doi:10.1007/978-3-319-39814-3_ 8
- [97] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, Wiley, 2020, ch. 4.3.3 - The MIG-in-the-Middle Attack
- [98] N. Tippenhauer, H. Luecken, M. Kuhn and S. Capkun, “UWB rapid-bit-exchange system for distance bounding,” in *2015 Proceedings of the 8th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, New York, NY, USA, 2015, pp. 1-12, DOI:10.1145/2766498.2766504.
- [99] P. Leu, G. Camurati, A. Heinrich, M. Roeschlin, C. Anliker, M. Hollick, S. Capkun, and J. Classen, “Ghost Peak: Practical Distance Reduction Attacks Against HRP UWB Ranging,” in *Proceedings of the 31st USENIX Security Symposium (USENIX)*, Boston, MA, USA, 2022, pp. 1343-1359.
- [100] K. Joo, D. H. Lee, Y. Jeong, and W. Choi, “Protecting HRP UWB Ranging System Against Distance Reduction Attacks,” in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS 23)*, New York, NY, USA, 2023, pp. 622–635, DOI: 10.1145/3576915.3623145.
- [101] Characteristics of Ultra-Wideband Technology, document ITU-R SM.1755-0, ITU, 2006. Accessed: Sept 2024. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/sm/R-REC-SM.1755-0-200605-I!!PDF-E.pdf
- [102] Electronic Code of Federal Regulation (e-CFR), FCC-15.503-Defintions, FCC, 2024. Accessed: Sept 2024. [Online]. Available: <https://www.ecfr.gov/current/title-47/chapter-I/subchapter-A/part-15/subpart-F/section-15.503>
- [103] “IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and physical-layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs): Amendment 1: Add Alternate PHYs,” in IEEE Std 802.15.4a-2007

- (Amendment to IEEE Std 802.15.4-2006) , vol., no., pp.1-210, 31 Aug. 2007, doi: 10.1109/IEEESTD.2007.4299496.
- [104] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)," in IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003) , vol., no., pp.1-320, 7 Sept. 2006, doi: 10.1109/IEEESTD.2006.232110.
 - [105] E. Udayakumar and V. Krishnaveni, "Analysis of various Interference in Millimeter-Wave Communication Systems: A Survey," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 2019, pp. 1-5, doi: 10.1109/ICCCNT45670.2019.8944417
 - [106] W. C. Chen, "5G mmWAVE Technology Design Challenges and Development Trends," 2020 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), Hsinchu, Taiwan, 2020, pp. 1-4, doi: 10.1109/VLSI-DAT49148.2020.9196316
 - [107] E. Karapistoli, F. -N. Pavlidou, I. Gragopoulos and I. Tsetsinas, "An overview of the IEEE 802.15.4a Standard," in IEEE Communications Magazine, vol. 48, no. 1, pp. 47-53, January 2010, doi: 10.1109/MCOM.2010.5394030
 - [108] AirTag, "Buy AirTag," Apple Inc, 2021. Accessed: Sept 2024. [Online]. Available: <https://www.apple.com/ca/shop/buy-airtag/airtag/1-pack>
 - [109] P. Sedlacek, M. Slanina and P. Masek, "An Overview of the IEEE 802.15.4z Standard its Comparison and to the Existing UWB Standards," 2019 29th International Conference Radioelektronika (RADIOELEKTRONIKA), Pardubice, Czech Republic, 2019, pp. 1-6, doi: 10.1109/RADIOELEK.2019.8733537
 - [110] R. Abubaker and G. Gong, "High-speed OFDM Physical-Layer Key Exchange," 2023 IEEE Conference on Communications and Network Security (CNS), Orlando, FL, USA, 2023, pp. 1-9, doi: 10.1109/CNS59707.2023.10288672
 - [111] P. Yin, L. Peng, J. Zhang, M. Liu, H. Fu and A. Hu, "LTE Device Identification Based on RF Fingerprint with Multi-Channel Convolutional Neural Network," 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 2021, pp. 1-6, doi: 10.1109/GLOBECOM46510.2021.9685067

- [112] W. Lee, S. Y. Baek and S. H. Kim, “Deep-Learning-Aided RF Fingerprinting for NFC Security,” in *IEEE Communications Magazine*, vol. 59, no. 5, pp. 96-101, May 2021, doi: 10.1109/MCOM.001.2000912
- [113] Y. Yuan and L. Peng, “Wireless Device Identification Based on Improved Convolutional Neural Network Model,” 2018 IEEE 18th International Conference on Communication Technology (ICCT), Chongqing, China, 2018, pp. 683-687, doi: 10.1109/ICCT.2018.8600086
- [114] A. Elmaghoub and B. Hamdaoui, “A Needle in a Haystack: Distinguishable Deep Neural Network Features for Domain-Agnostic Device Fingerprinting,” 2023 IEEE Conference on Communications and Network Security (CNS), Orlando, FL, USA, 2023, pp. 1-9, doi: 10.1109/CNS59707.2023.10288752
- [115] M. Fomichev, L. F. Abanto-leon, M. Stiegler, A. Molina, J. Link, and M. Hollick, “Next2You: Robust Copresence Detection Based on Channel State Information,” in *ACM Transaction on Internet of Things*, vol. 3, no. 2, pp. 1-31, May 2022, doi: 10.1145/3491244
- [116] H. Ólafsdóttir, A. Ranganathan, and S. Capkun, “On the Security of Carrier Phase-based Ranging,” *Arxiv*, Oct 2016. Accessed: Sept 2024. [Online]. Available: <https://arxiv.org/abs/1610.06077>
- [117] A. Riaz, D. Nash, J. Ngo, C. Juvekar, P. Nadeau, T. Yu, and R. Tugce Yazicigil, “Security Assessment of Phase-Based Ranging Systems in a Multipath Environment”, *ACM Journal on Emerging Technologies in Computing Systems*, vol. 18, no. 4, Oct 2022, pp. 1-19, doi.org/10.1145/3517809
- [118] B. Nisarga, “High Accuracy, Low Cost, Secure Ranging with Bluetooth Channel Sounding,” *Texas Instruments: Application Note*, 2024. Accessed: Sept 2024. [Online]. Available: <https://www.ti.com/lit/an/swra791/swra791.pdf?ts=1727144853294>
- [119] M. Woolley, “Bluetooth Channel Sounding: Technical Overview,” *Bluetooth Special Interest Group*, 2024. Accessed: Sept 2024. [Online]. Available: <https://www.bluetooth.com/channel-sounding-tech-overview/>
- [120] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, “Tool Release: Gathering 802.11n Traces with Channel State Information,” 2011 ACM Signal Communication (SIGCOMM), Toronto, Canada, 2011, vol. 41, no. 1, Jan 2011, doi: 10.1145/1925861.1925870

- [121] “Buildroot: Making Embedded Linux Easy,” Buildroot, Accessed: Sept. 2024 [Online]. Available: <https://www.buildroot.org/>
- [122] Analog Devices, Inc., ”AD9363: 2400 MHz to 2500 MHz/5150 MHz to 6050 MHz RF Agile Transceiver,” data sheet, Rev. B, Norwood, MA, USA, Aug. 2016. [Online]. Available: <https://www.analog.com/media/en/technical-documentation/data-sheets/AD9363.pdf>
- [123] R. Abubaker and G. Gong, “Frequency distance sequences for packet detection in physical-layer security,” *Designs, Codes and Cryptography*, 2024, doi:10.1007/s10623-024-01475-x
- [124] M. Hennerich and T. Collins, “PlutoSDR Workshop,” presented at GNU Radio Conference (GRCon) 2019, Huntsville, AL, USA, Sep. 16-20, 2019. [Online]. Available: https://wiki.analog.com/_media/adiplutoworkshop_grcon2019.pdf
- [125] M. Hennerich, “PlutoSDR: Capabilities, Use Cases and Getting Started,” presented at GNU Radio Conference (GRCon) 2018, Henderson, NV, USA, Sep. 17-21, 2018. [Online]. Available: https://www.gnuradio.org/grcon/grcon18/presentations/PlutoSDR/8-Michael_Hennerich.pdf
- [126] Analog Devices, “Libiio release v0.25,” GitHub, 2024. Accessed: Nov. 2024. [Online]. Available: <https://github.com/analogdevicesinc/libiio/releases/tag/v0.25>
- [127] hz12opensource, “LibreSDR with Zynq SoC overclock and Optimized Sample Rates,” GitHub, 2024. Accessed: Nov. 2024. [Online]. Available: <https://github.com/hz12opensource/libresdr>
- [128] R. Abubaker and G. Gong, “Relay Attack Detection using OFDM Channel-Fingerprinting,” *TechRxiv*, 2024. Accessed: Oct 2024. [Online]. Available: <https://www.techrxiv.org/users/719682/articles/705419-relay-attack-detection-using-ofdm-channel-fingerprinting>
- [129] A. Milewski, “Periodic Sequences with Optimal Properties for Channel Estimation and Fast Start-Up Equalization,” in *IBM Journal of Research and Development*, vol. 27, no. 5, pp. 426-431, Sept. 1983, doi: 10.1147/rd.275.0426.
- [130] C. Phillips and J. Parr, *Signals, Systems, and Transforms*, Prentice-Hall, 2008, ch. 12.6 - Applications of the Discrete Fourier Transform

- [131] W. H. Mow, "A new unified construction of perfect root-of-unity sequences," Proceedings of ISSSTA'95 International Symposium on Spread Spectrum Techniques and Applications, Mainz, Germany, 1996, pp. 955-959 vol.3, doi: 10.1109/ISSSTA.1996.563445.
- [132] R. Abubaker, "Pilot Sequence Simulation," GitHub, 2024. [Online]. Available: <https://github.com/rabubake/pilot-sequence-sim>
- [133] M. M. U. Gul, X. Ma and S. Lee, "Timing and Frequency Synchronization for OFDM Downlink Transmissions Using Zadoff-Chu Sequences," in IEEE Transactions on Wireless Communications, vol. 14, no. 3, pp. 1716-1729, March 2015, doi: 10.1109/TWC.2014.2372757
- [134] O. Soffke, P. Zhao, T. Hollstein, and M. Glesner, "Modelling HF and UHF RFID Technology for System and Circuit Level Simulations," 3rd European Workshop on RFID Systems and Technologies, 2007. Accessed: Oct 16 2024. [Online] http://rfid-systech.eu/20070612_1A_1505_Soffke_ModellingHFAndUHFSystems.pdf
- [135] BAE Systems, "What are IFF Technologies?" Accessed: Oct. 2024. [Online]. Available: <https://www.baesystems.com/en-us/definition/what-are-iff-technologies>
- [136] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice-Hall, 1998, ch, 5 - General MVU Estimation
- [137] Pennsylvania State University, "Sampling Distribution of Sample Variance," STAT 414: Introduction to Probability Theory, Eberly College of Science, The Pennsylvania State University. Accessed: Oct. 2024. [Online]. Available: <https://online.stat.psu.edu/stat414/lesson/26/26.3>
- [138] Rakon Ltd., "RXO3225M 40 MHz Oscillator Specifications," Rakon Ltd., Datasheet, Document No. 513371, 1.8V, Accessed: Oct. 2024. [Online]. Available: https://wiki.analog.com/_media/university/tools/pluto/hacking/rakon_rxo3225m_40mhz_513371_1.8v.pdf
- [139] Wei Yu and J. M. Cioffi, "Constant-power waterfilling: performance bound and low-complexity implementation," in IEEE Transactions on Communications, vol. 54, no. 1, pp. 23-28, Jan. 2006, doi: 10.1109/TCOMM.2005.861678.

- [140] Raspberry Pi Ltd., “Buy a Raspberry Pi 5,” Raspberry Pi, Accessed: Oct.2024.[Online]. Oct. 2024. Available: <https://www.raspberrypi.com/products/raspberry-pi-5/>
- [141] M. Simon, “Product of Random Variables,” in *Probability distribution involving Gaussian random variables: a handbook for engineers and scientists*. New York, NY, USA: Springer, 2006, ch. 6, pp.49-60.
- [142] M. Simon, “ProdeOct.2Aduct of Random Variables,” in *Probability distribution involving Gaussian random variables: a handbook for engineers and scientists*. New York, NY, USA: Springer, 2006, ch. 6, pp.49-60.
- [143] P. Royston, “Approximating the Shapiro-Wilk W-test for non-normality,” *Statistics and Computing*, vol. 2, pp. 117–119, Sept. 1992, DOI:10.1007/BF01891203
- [144] T. S. Rappaport, K. Blankenship, and H. Xu, “Propagation and radio system design issues in mobile radio systems for the GloMo project,” Mobile and Portable Radio Research Group, Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, 1997.
- [145] P. Moose, “A Technique for Orthogonal Frequency Division Multiplexing Frequency Offset Correction,” *IEEE Transactions on Communications*, Vol. 42, No. 10, October 1994
- [146] DIGITPOL, “Relay Attack with Massive Antenna - Criminals Steal BMW M2 Competition in 56s - Tesla Not Looked At,” YouTube, Accessed: Nov. 4, 2024. [Online]. Available: https://www.youtube.com/watch?v=uxzm_6SYBFo
- [147] A. Mejias, “Relay Attack - Useless Faraday Key Fob Box,” YouTube, Accessed: Nov. 4, 2024. [Online]. Available: <https://www.youtube.com/watch?v=Yq0fNp9nQSA>
- [148] Car Theft CCTV, “Mercedes E-Class Coupe Stolen — Keyless Entry,” YouTube, Accessed: Nov. 4, 2024. [Online]. Available: <https://www.youtube.com/watch?v=FgKTbLIM3Vg>
- [149] KEYSHIELD, “Keyless Car Relay Attack,” YouTube, Nov. 4, 2024. [Online]. Available: <https://www.youtube.com/watch?v=nBezSk95L0w>
- [150] CityNews, “Thieves use electronic method to steal cars,” YouTube, Accessed: Nov. 4, 2024. [Online]. Available: <https://www.youtube.com/watch?v=MK7k0vktIo4>

- [151] harriermk2, "Car thieves, essex, relay theft," YouTube, Nov. 4, 2024. [Online]. Available: <https://www.youtube.com/shorts/Ri1uK7nbpK4>
- [152] Newsflare, "Thieves caught using keyless hack to steal £90,000 Tesla in 30 seconds," YouTube, Accessed: Nov. 4, 2024. [Online]. Available: <https://www.youtube.com/watch?v=hj3ZRv9cMBw>
- [153] R. Frank, S. Zadooff, and R. Heimiller, "Phase shift pulse codes with good periodic correlation properties," in IRE Transactions on Information Theory, vol. 7, no. 4, pp. 254-257, October 1961, doi: 10.1109/TIT.1961.1057655roperties (Corresp.)," IRE Transaction on Information Theory, pp. 381-382, October 1962
- [154] D. Chu, "Polyphase codes with good periodic correlation properties (Corresp.)," in IEEE Transactions on Information Theory, vol. 18, no. 4, pp. 531-532, July 1972, doi: 10.1109/TIT.1972.1054840
- [155] J. A. Davis and J. Jedwab, "Peak-to-mean power control in OFDM, Golay complementary sequences, and Reed-Muller codes," in IEEE Transactions on Information Theory, vol. 45, no. 7, pp. 2397-2417, Nov. 1999, doi: 10.1109/18.796380
- [156] R. Abubaker and G. Gong, "Evaluating Channel Based Relay Attack Detection in the Near-Field," IEEE International Conference on RFID Technology and Applications 2024, to be published
- [157] A. Bazzi, "The Water Filling Algorithm in Wireless Communications," YouTube, Nov. 20, 2020. Accessed: Nov. 4 2024. [Online]. Available: <https://www.youtube.com/watch?v=P4z9RqxTaRc>
- [158] R. Abubaker and G. Gong, "Evaluating Channel Based Relay Attack Detection in the Near-Field," IEEE International Conference on RFID Technology and Applications 2024, to be published
- [159] P. Yu, J. Baras, and B. Sadler, "Physical-layer authentication," IEEE Transactions on Information Forensics and Security, vol. 3, no. 1, pp. 38-51, Mar. 2008
- [160] F. S. Nahm, "Receiver operating characteristic curve: overview and practical use for clinicians," Korean J. Anesthesiol., vol. 75, no. 1, pp. 25-36, Feb. 2022. DOI: 10.4097/kja.21209