

Security and Privacy in the Internet of Things

Submitted by
Zechariah Gerard Tan Jia Le

In partial fulfilment of the
requirements for the Degree of
Bachelor of Engineering (Computer Engineering)
National University of Singapore

B.Eng. Dissertation

Security and Privacy in the Internet of Things

By

Zechariah Gerard Tan Jia Le

National University of Singapore

2019/2020

Project ID: H63819

Project Supervisor: Assoc Prof Biplab Sikdar

Deliverables:

Report: 1 Volume

ABSTRACT

The Internet of Things (IoT) has certainly made our lives much more convenient, tightly integrating our everyday lives with smart technology. Powered by wireless communications, these smart appliances stay connected to the internet to provide us with seamless functionality. As IoT continues evolving and bringing in more everyday appliances, the newest technology powering wireless communication has been recently introduced, known as the 802.11ax standard, or simply Wi-Fi 6.

Wi-Fi 6 comes with new features, one of which is Orthogonal Frequency Division Multiple Access (OFDMA). A brief introduction to this feature is given along with its close cousin Orthogonal Frequency Division Multiplexing (OFDM). As future IoT devices adopts this new Wi-Fi technology and expands to include even more devices, it becomes increasingly lucrative for adversaries to target these devices. In this paper, the usage of jamming attacks to compromise the availability to use such IoT devices will be discussed. Past attacks against its cousin OFDM will be looked at, as well as attacks against OFDMA but on other technologies. An experiment is then designed based on mostly commercial equipment that is readily available and conducted to observe the impact of some of these attacks against OFDMA. It is observed that while barrage jamming may be the most destructive attack against OFDMA, other jamming attacks, particularly the asynchronous off-tone jamming attack, are more power-efficient but able to deal respectable damage.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor Biplab Sikdar, for his guidance and support throughout the entire period of my final year.

I would also like to thank Mr Siow Hong Lin, Eric, for supporting the purchase of equipment required for this research as well as directing me to the correct resources.

Special thanks to my schoolmates, my family, and friends who always encourage, support and care for me throughout this project.

Lastly, I greatly appreciate all the supports and helps from the staff in National University of Singapore to completion of this thesis.

CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENTS	ii
LIST OF FIGURES	v
LIST OF TABLES	vii
LIST OF SYMBOLS AND ABBREVIATIONS	viii
CHAPTER 1 - Introduction	1
1.1 Introduction to Field of Study	1
1.2 Specific Field of Study	2
1.3 Link to Thesis Topic	4
CHAPTER 2 - Background	6
2.1 Wireless Signals	6
2.2 Workings of Wi-Fi	8
2.2.1 OFDM	8
2.2.2 OFDMA	10
CHAPTER 3 – Literature Review	13
3.1 Past Research on OFDM	13
3.1.1 Wideband / Barrage Jamming (BNJ)	14
3.1.2 Partial Band Jamming (PBJ)	15
3.1.3 Pilot Jamming (PJ)	16
3.1.4 Pilot Nulling (PN)	18
3.1.5 Multi-Tone Jamming (MTJ)	19
3.1.6 Asynchronous Off-Tone Jamming	20
3.1.7 False Preamble Timing Attack	22
3.1.8 Preamble Phase Warping	23
3.1.9 Differential Scrambling Attack	24
3.1.10 Cyclic Prefix (CP) Jamming	25
3.2 Past Research on OFDMA	26
3.2.1 HARQ ACK Attack	26
3.2.2 Resource Allocation Attack	27
3.2.3 Random Access Channel Attack	27

CHAPTER 4 – Design and Implementation	29
4.1 Apparatus used in Experiment	29
4.1.1 Transmitter Node	29
4.1.2 Receiver Node	30
4.1.3 Jamming Device	31
4.2 Experiment Procedure & Set-Up	32
4.2.1 Procedure	32
4.2.2 Quantification of Results	32
4.2.3 Wireless Connection Environment Set-Up	34
4.2.4 Operating Parameters of Transmitter and Receiver	34
4.2.5 Operating Parameters of Jammer	37
4.3 Actual Experiment Procedure	39
4.3.1 Experimental Settings used for the Jammer	39
4.3.2 Deriving the independent variable	42
4.3.3 Conducting the transmission from transmitter to receiver	44
4.3.3 Deriving the dependent variable	47
CHAPTER 5 – Results	49
CHAPTER 6 – Conclusion	56
6.1 Summary	56
6.2 Future work that can be considered	57
REFERENCES	58
APPENDIX A – C Source Code used for UDP sending and receiving	61
APPENDIX B – C Source Code used for calculating Hamming Distance	67

LIST OF FIGURES

Fig. 1 - Internet of Things application ranking [5]	4
Fig. 2 - Frequency Spectrum of a Modulated Waveform [7]	7
Fig. 3 - FDM Process [8]	7
Fig. 4 - OFDM: Transmitting Data through multiple subcarriers [9]	8
Fig. 5 - Subcarrier sideband overlap [11]	9
Fig. 6 - Frequency Band Allocation for OFDMA [16]	11
Fig. 7 - Frequency Band Allocation for OFDM [16]	11
Fig. 8 - Visualization of the different frequency spectrum for different jamming attacks [19]	14
Fig. 9 - Performance of Barrage Jamming (BNJ), quantified by BER [20]	15
Fig. 10 - Performance of Partial Band Jamming (PBJ) quantified in BER [21]	16
Fig. 11 - Performance of Pilot Jamming (PJ) and Pilot Nulling (PN) quantified in BER [22]	17
Fig. 12 - Performance of Multi-Tone Jamming (MTJ) quantified in BER [19]	19
Fig. 13 - ICI caused by asynchronous jamming tones [19]	20
Fig. 14 - Performance of asynchronous off-tone jamming quantified in BER [19]	21
Fig. 15 - Performance of False Preamble Timing Attack quantified in BER [15]	22
Fig. 16 - Destructive Effects of Preamble Phase Warping, with the frequency offset after the jammer is active highlighted (the normal offset without the presence of a jammer is indicated in the freq. offset box) and the BER of the attack [24]	23
Fig. 17 - Performance of Differential Scrambling Attack quantified in BER [25]	24
Fig. 18 - Performance of Random Access Channel Attack quantified in Rate of Correct Detection	28
Fig. 19 - The Archer AX10 Wi-Fi 6 AX1500 Gigabit Router with 1.5GHz Triple-Core CPU used as the transmitter node [28]	29

Fig. 20 - Dual Band Wireless AX200NGW 2.4 Gbps 802.11ax Wireless Intel AX200 Wi-Fi card with Bluetooth 5.0 for Windows 10, used as the receiver node [35]	30
Fig. 21 – Rohde & Schwarz SMW200A Vector Signal Generator [36]	31
Fig. 22 - Frequency Spectrum of the Wi-Fi 6 router used as transmitter node	35
Fig. 23 - Additional Details about the Wi-Fi 6 router used as transmitter node	36
Fig. 24 - Additional Details about the Wi-Fi 6 router used as transmitter node	36
Fig. 25 - Installed Wi-Fi Card Details	36
Fig. 26 – General Set-up of the Signal Generator for generating AWGN	37
Fig. 27 - Set-up of the Signal Generator for generating 2 AWGN Signals	38
Fig. 28 - Illustration of the sub-carriers in a 20 MHz channel from the Rohde & Schwarz IEEE 802.11ax Technology Introduction white paper [34]	40
Fig. 29 - Frequency Spectrum of Partial Band Jamming (PBJ) Signal	41
Fig. 30 - Compiling and Running the program in preparation to receive the file from transmitter	44
Fig. 31 - Compiling and Running the program to send the file to receiver	45
Fig. 32 - File successfully received from transmitter	46
Fig. 33 - The transmitted / original file (left) vs the received file (right)	47
Fig. 34 - Performance of BNJ quantified in BER	50
Fig. 35 - Performance of PBJ quantified in BER	51
Fig. 36 - Performance of MTJ quantified in BER	52
Fig. 37 - Performance of Asynchronous Off-Tone Jamming quantified in BER	53
Fig. 38 - Comparison of different jamming attacks	54

LIST OF TABLES

Table 1 – 802.11ax (Wi-Fi 6) vs Legacy Wi-Fi versions	2
Table 2 - Technical Specifications for Dual Band Wireless AX200NGW 2.4 Gbps 802.11ax Wireless Intel AX200 Wi-Fi card with Bluetooth 5.0 for Windows 10, used as the receiver node [29]	30
Table 3 - Results of BER against SJR under Barrage Jamming (BNJ)	50
Table 4 - Results of BER against SJR under Partial Band Jamming (PBJ)	51
Table 5 - Results of BER against SJR under Multi-Tone Jamming (MTJ)	52
Table 6 - Results of BER against SJR under Asynchronous Off-Tone Jamming	53

LIST OF SYMBOLS AND ABBREVIATIONS

m	Milli, denotes a magnitude factor of 10^{-3}
k	Kilo, denotes a magnitude factor of 10^3
M	Mega, denotes a magnitude factor of 10^6
G	Giga, denotes a magnitude factor of 10^9
Hz	Hertz, the unit for frequency, which is the reciprocal of the Système Internationale (SI) unit of seconds
α	Path loss component
μ	Micro, denotes a magnitude factor of 10^{-6}
π	Pi, used in radians to indicate phase differences between waves. π radians = 180° . Also used as its absolute value in calculation as 3.14159...
4G	4 th Generation
ACK	Acknowledgment
ARQ	Automatic Repeat reQuest
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BNJ	Barrage Noise Jamming
bps	bits per second
BSS	Basic Service Set
CIA	Confidentiality, Integrity, Availability
CP	Cyclic Prefix
CRC	Cyclic Redundancy Check
CSI	Channel State Information
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance

dB	Decibels
dBm	Power ratio expressed in decibels (dB) with reference to one milliwatt (mW)
DFS	Dynamic Frequency Selection
DL	Downlink
FDM	Frequency Division Multiplexing
FO	Frequency Offset
HARQ	Hybrid Automatic Repeat reQuest
ICI	Inter-Channel Interference
ICT	Infocomm and Communication Technology
IEEE	Institute of Electrical and Electronics Engineers
IMDA	Info-communications Media Development Authority
IoT	Internet of Things
ISI	Inter-Symbol Interference
LAN	Local Area Network
LTE	Long Term Evolution
MAC	Media Access Control
MB	Megabytes
MIMO	Multiple Input, Multiple Output
MTJ	Multi-Tone Jamming
MU	Multi-User
OFDM	Orthogonal Frequency-Division Multiplexing
OFDMA	Orthogonal Frequency-Division Multiple Access
PBJ	Partial Band Jamming
PJ	Pilot Jamming
PN	Pilot Nulling

PSD	Power Spectral Density
QAM	Quadrature Amplitude Modulation
RU	Resource Unit
SINR	Signal-to-Interference-plus-Noise Ratio
SJR	Signal-to-Jamming Ratio
SSID	Service Set Identifier
TPC	Transmit Power Control
TSAC	Telecommunications Standards Advisory Committee
TWT	Target Wake Time
UDP	User Datagram Protocol
UL	Uplink
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Inter-operability for Microwave Access

CHAPTER 1 - Introduction

1.1 Introduction to Field of Study

With the increased use of technology in our everyday lives, technology has become more deeply embedded in the everyday household items we deal with, leading to the rise of IoT. The convenience brought about from the integration of technology is undeniable, however, the need for these technologies to be integrated online has led to security concerns such as the leakage of confidential information or private data we do not wish to share.

Technology these days is integrated into the internet with Wi-Fi. It is one of the most widely used technologies which is easy to use and reliable. Wi-Fi is a wireless broadband network service which has transmitters (Access Point) to transmit signals and wireless stations such as laptops and handphones to respond to the signal and communicate with one another. The wireless feature of Wi-Fi eliminates the need for wirings, which can incur additional overhead costs. It also allows anyone in range to connect to the Wi-Fi network easily. The demand for Wi-Fi has been increasing, with the total public Wi-Fi hotspots growing sevenfold from 2015 to 2020, from 64.2 million in 2015 to 432.5 million by 2020 [1].

This thesis will examine the concept of Wi-Fi. Specifically, a new standard of Wi-Fi has rolled out last year at the time of this writing (2019). The newest generation of iPhone and Samsung phones (at the time of writing), the iPhone 11 and the Samsung Galaxy S10 and Note 10, now supports this standard, 802.11ax, or simply known as Wi-Fi 6 [2].

There were many revisions of Wi-Fi. The first Wi-Fi, IEEE standard 802.11n (now simply known as Wi-Fi 4), was released in 2009. This is followed by IEEE standard 802.11ac (now simply known as Wi-Fi 5) in 2014. This 802.11ac standard is now about to be superseded by IEEE 802.11ax, or Wi-Fi 6.

1.2 Specific Field of Study

The new Wi-Fi 6 introduces various new revolutionary techniques to improve our Wi-Fi experiences. These new techniques include OFDMA, MU-MIMO, TWT, usage of 1024-QAM and the long OFDM symbol as well as BSS Colouring. Some of these features compared to the legacy Wi-Fi revisions are shown in Table 1 below. [3] These aim to optimize spectral efficiency, increase throughput and reduce power consumption. This thesis mainly focuses on the revolutionary upgrade of Wi-Fi of the new OFDMA. Previous revisions of Wi-Fi used to use OFDM.

Table 1 – 802.11ax (Wi-Fi 6) vs Legacy Wi-Fi versions

	Legacy Wi-Fi features	802.11ax features
OFDM Constellation Order	256-QAM (in Wi-Fi 5)	1024-QAM
OFDM Symbol Duration	3.2 μ s	12.8 μ s
MIMO Order	4 (in Wi-Fi 4), 8 (in Wi-Fi 5)	8
Basic Channel Access	CSMA/CA	OFDMA on top of CSMA/CA
MU Technology	MU-MIMO (in Wi-Fi 5)	MU-MIMO, OFDMA
MU Transmission Direction	DL (in Wi-Fi 5)	DL and UL
Spatial Reuse	Sectorization (in 802.11ah)	Adaptive Power and Sensitivity Thresholds, Colour
Power Management	Many	Enhanced TWT, Enhanced Microsleep

The usage of OFDMA is not new but is new to Wi-Fi. This technique has been already been use in cellular networks, such as the popular LTE network, which is what allows us to use our 4G cellular networks while on the go on our mobile devices such as smartphones and tablets. [3] However, there are still vulnerabilities in OFDMA that is prevalent. Particularly, past researches have focused their efforts on documenting the weaknesses of LTE networks against jamming attacks. Wi-Fi 6 is still a very novel technology and is thus not well researched on yet.

In this thesis, we shall explore how effective several jamming attacks are on Wi-Fi 6, based on jamming techniques on OFDMA. This research is also a first to use mostly easily commercially available equipment rather than specialized lab equipment, except for a jammer device since that is not easily available in the consumer market.

We have found that certain types of jamming such as asynchronous off-tone jamming are more damaging yet power efficient to wireless signals than others. Obviously, the lower the SJR, the higher the damage is to the wireless signal.

This report will first cover the background of wireless signals, going in-depth into how OFDMA works by going through its predecessor techniques. It will then cover past researches on different types of jamming attacks against both OFDM and OFDMA techniques. The experiment parameters and how the experiment is conducted will be explained. Along the way, the quantification of results will be mentioned to serve as a metric of how damaging jamming attacks are on OFDMA signals. Finally, the results are presented with the quantified variables mentioned before the results and a conclusion is derived.

1.3 Link to Thesis Topic

This thesis is about the privacy and security of the Internet of Things (IoT). Over the next few months or years, the new Wi-Fi 6 will be adopted by almost every upcoming smart device, which includes laptops, smartphones, tablets, and even smart devices such as home assistants or smart appliances. It is expected that 1 trillion devices might be connected to the internet by 2025. [4] It is thus of concern to investigate Wi-Fi 6 which will affect the IoT soon. When looking at security, the CIA triad in security comes to mind. When talking about jamming attacks against OFDMA, we are specifically looking at the Availability aspect of security.

IoT is becoming a bigger and bigger part of our lives and has many types of applications. Fig. 1 below shows the various applications.

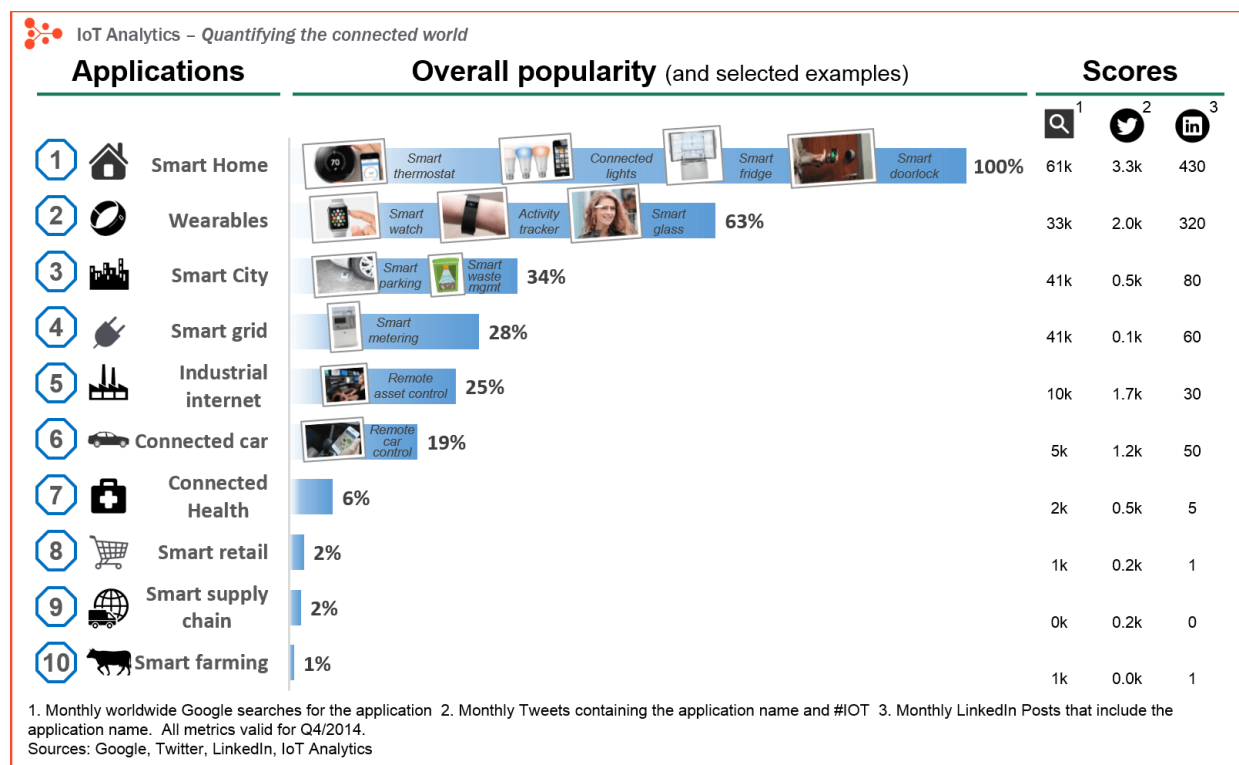


Fig. 1 - Internet of Things application ranking [5]

If jamming attacks were to occur, it could severely affect applications such as smart parking, smart waste management, smart monitoring, remote asset control, remote car control, healthcare systems, and perhaps much more. These attacks would affect our daily lives as well as disrupt city systems.

CHAPTER 2 - Background

2.1 Wireless Signals

In the basic essence, Wi-Fi involves the use of wireless signals. Today, we use Wi-Fi to surf the internet, make internet calls and remotely control another device. The very basic workings of Wi-Fi are to transfer data from one device/node to another. In the case of web surfing, the device you use is simply trying to talk to the server of the website that you wish to surf, and the website's server replies to you the data that you require to load your webpage, audio, video, etc.

While it sounds as simple as two nodes trying to communicate data to each other, the problem becomes much more complicated when there are many other users in the vicinity that also wish to communicate with each other. Like several people trying to talk to different parties in close vicinity, it becomes very difficult for the receiving parties to hear what the other person is trying to say when there are several messages being transmitted at the same time on the air.

Fortunately, Wi-Fi mostly solves that problem by using a technique known as FDM. FDM is a telecommunicating method which assigns non-overlapping frequency ranges to every device/node that uses the same medium. Each node thus gets its own frequency band and only listens for wireless signals that are within this frequency band. The original wireless signal, which is a waveform, is first transformed by combining it with a carrier waveform which is generated by an electronic oscillator. This carrier waveform has a constant frequency which is much higher than that of the original wireless signal. The signals are multiplied together in modulator circuits. The resulting waveform is a waveform with

frequencies only within the frequency band allocated to the node. [6] An example frequency spectrum, which measures the amplitude of various frequencies in the waveform is shown in Fig. 2 below:

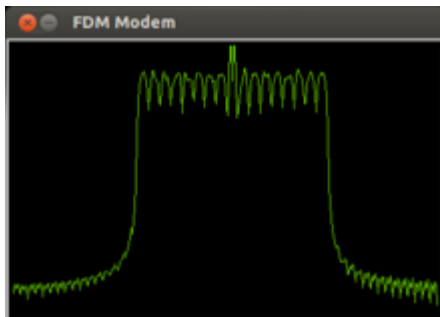


Fig. 2 - Frequency Spectrum of a Modulated Waveform [7]

The modulated waveform above is then sent over the air to the other party. At possibly the same time, other nodes which also wish to communicate with another party also send their modulated waveforms over the air. This results in the waveforms being added up. At the receiving end, the receiving parties then must reconstruct the original signal sent by the sender. It does this by first applying a bandpass filter to only receive the frequency band of the sender, and then demodulating the signal with the same carrier signal used by the sender, essentially reversing the process done at the sender. Fig. 3 below shows the process:

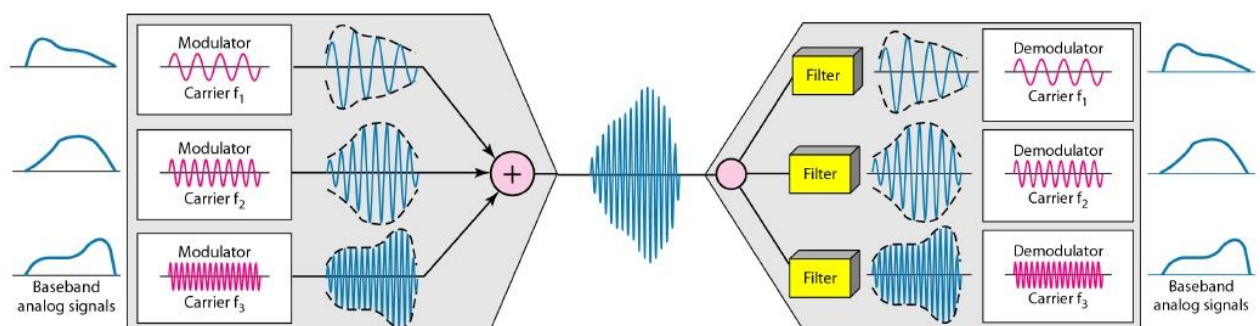


Fig. 3 - FDM Process [8]

2.2 Workings of Wi-Fi

With the understanding of FDM, we can explain how Wi-Fi manages communication between devices. FDM is the oldest multiplexing technique but is still employed in many fields of wireless communications. [6]

2.2.1 OFDM

Most modern Wi-Fi employs a derivative version of FDM known as OFDM. In FDM, the receiver is only interested in a narrow frequency band. However, in OFDM, the receiver now listens out for all frequencies. To make the data rate higher, the data is carried on multiple subcarriers. The data is equally distributed on all the subcarriers. Each of the subcarriers modulates the part of the data it was given by multiplying it by the carrier frequency. Then, all the subcarriers are sent out at the same time to the receiver. This is shown in Fig. 4 below. The receiver demodulates all the signals using a bandpass filter to recover each of the subcarrier waveforms, just like in FDM.

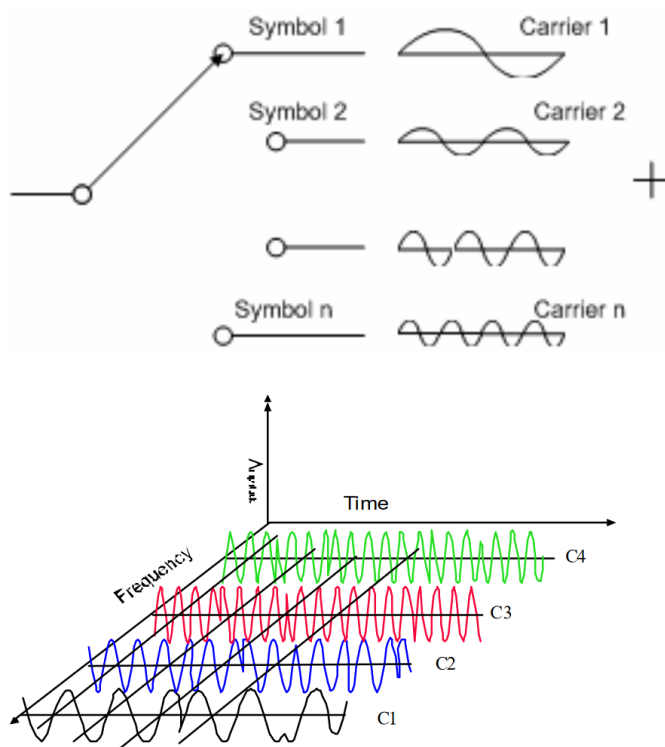
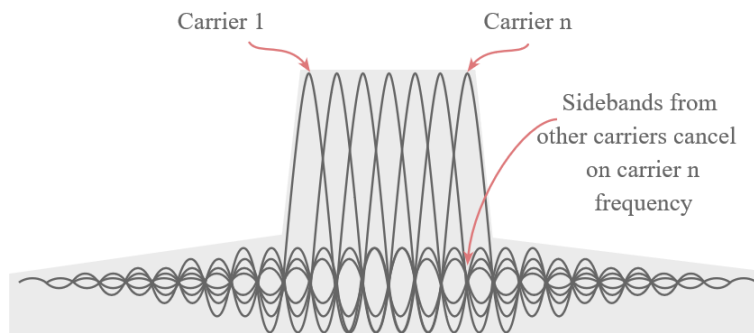


Fig. 4 - OFDM: Transmitting Data through multiple subcarriers [9]

To optimize the frequencies even better, the subcarrier frequency bands are placed as close together as possible. They are so close to each other, that their sidebands even overlap with each other as shown in Fig. 5 below. [10]



Basic concept of OFDM, Orthogonal Frequency Division Multiplexing

Fig. 5 - Subcarrier sideband overlap [11]

Despite the overlap between the sidebands, the receiver is still able to fully reconstruct the subcarrier signals, because the signals are what is known as orthogonal to each other, hence the name OFDM. The Orthogonal property in this case is when the peak amplitude of one subcarrier is the zero point of another subcarrier at a certain frequency. This is achieved by setting the first subcarrier to be at least twice information rate from the Nyquist Sampling Theorem, and the other subcarriers as multiples of that first subcarrier, which is the harmonic frequency. [9]

While most subcarrier waves are used to carry actual information, there are a few subcarriers which are used to determine the CSI. These subcarrier waves carry training OFDM symbols or pilot symbols. These symbols are sprinkled across the subcarrier waves, to provide CSI continuously. [12] The pilot subcarriers carry a known data sequence. This data sequence is used to determine CSI because the timing it is received and the bits received allows the

receiver to determine any frequency offsets and phase noise, as well as adjust for other factors such as multi-paths (the arrival of the same signal at different times as the wireless signal, which is transmitted omnidirectionally take different paths). [13] If the data in these pilot symbols stay constant, then the channel estimates are reliable enough for other symbol detections, which include the symbol detection for data subcarriers. [12]

2.2.2 OFDMA

However, the newest Wi-Fi 6 that is of research in this thesis uses an improved version of OFDM which is known as OFDMA. The main difference between OFDM and OFDMA is that the former does not allow multiple access, since OFDM makes a node use the entire frequency spectrum. Instead, OFDMA allows more than one node to share certain frequency bands.

These days, as the data rates get higher and higher, and the number of packets we need to send increases due to increasing number of devices using the network [1], there is significantly more competition compared to in the past. In OFDM, a transmitter needs to claim superiority over the air before it can transmit, since not doing so may cause other transmitters to transmit at the same time, thus causing interference and the transmitted data to be corrupted. The process of claiming superiority over the air may take a very short time in the order of microseconds, but it still incurs some overhead time due to the need for a contention period and a preamble. When there are many nodes that wish to transmit different packets, this short time would add up over time and soon become significant. [14]

OFDMA instead allows multiple nodes to be preamble at the same time, and their packets are then sent to them simultaneously. Instead of each node using the entire frequency spectrum to transmit the packets, different frequency bands are assigned to each transmitter. This is equivalent to splitting the different sub-carriers available to different transmitters, and the

band of frequencies assigned to a transmitter can differ based on its bandwidth or the data packet size. Devices with lower data rate can be allocated a smaller number of frequency bands. [14] Fig. 6 shows the frequency band allocation for OFDMA, compared to OFDM in Fig. 7. Each of the blocks below represents an RU. Most of the time, RUs are grouped together since allocating each of every RU in its own time and frequency has a very high overhead and would slow down transmissions, since more information about the allocation of RUs would be required. [15]



Fig. 6 - Frequency Band Allocation for OFDMA [16]

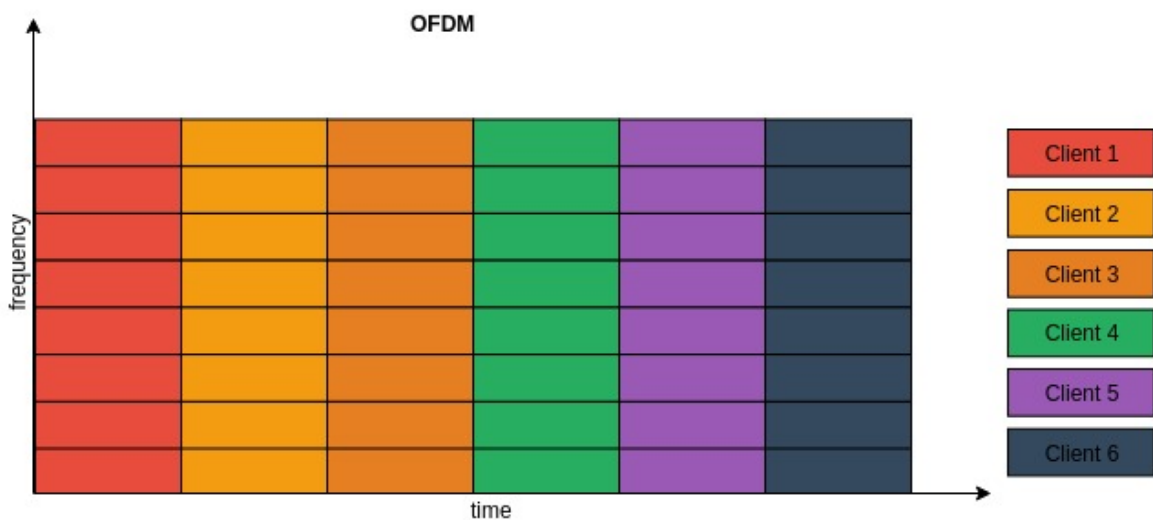


Fig. 7 - Frequency Band Allocation for OFDM [16]

The advantage of OFDMA over OFDM in this case is that there are times where there simply isn't enough data to be put on all the subcarriers, wasting some of the frequency bands. To sum it up, OFDMA allocates bandwidth to transmit data in both the time and frequency domain, while OFDM only makes use of the time domain to allocate bandwidth, since OFDM simply uses the entire frequency spectrum available. [14]

Another improvement of OFDMA over OFDM is the introduction of a longer OFDM symbol time of $12.8\mu\text{s}$ over $3.2\mu\text{s}$, which translates to the decrease of sub-carrier size and spacing from 312.5 kHz to 78.125 kHz and the increase of the number of subcarriers by 4 times the original number of sub-carriers available. [17]

CHAPTER 3 – Literature Review

Several researches have been done on its predecessor technique, which is OFDM, as well as OFDMA systems. These researches have been done on earlier devices, or emulated devices, but not on devices which adopt the new Wi-Fi 6 technology.

As both techniques rely on the usage of orthogonality, it is also worth looking at past research on OFDM, particularly those that investigate attacks on orthogonality.

OFDMA is not a new technology and has been implemented in other types of wireless networks, including LTE and WiMAX. [18] We will also look at attacks against OFDMA on these wireless networks.

3.1 Past Research on OFDM

There are several ways to jam OFDM signals, which include attacks as simple as BNJ.

Slightly more complicated attacks include the use of PBJ where certain bands of OFDM signals are being jammed intentionally. Other attacks on OFDM include PJ, PN, MTJ, false preamble timing, preamble phase warping, cyclic prefix attack and differential scrambling attack. A novel technique known as asynchronous off tone jamming attack has also been proposed [19], and shown to be more effective than the conventional BNJ.

Simpler jamming attacks often transmit noise in a large range of frequency bands, thus disrupting the OFDM signal. However, such an attack is not only very costly in terms of the power consumption but is easily detectable. Being able to disrupt the information sent on OFDM signals without being detected makes network engineers monitoring the network much less likely able to detect the presence of an attack, or unable to isolate the cause of the signal disruption.

Fig. 8 visualizes how some of the different jamming attacks work. The PSD indicates the signal strength at that frequency, the curves that do not have the area under them shaded is the frequency spectrum of an OFDM signal, while those curves with the area under them shaded indicates the frequency spectrum of the various jamming signals.

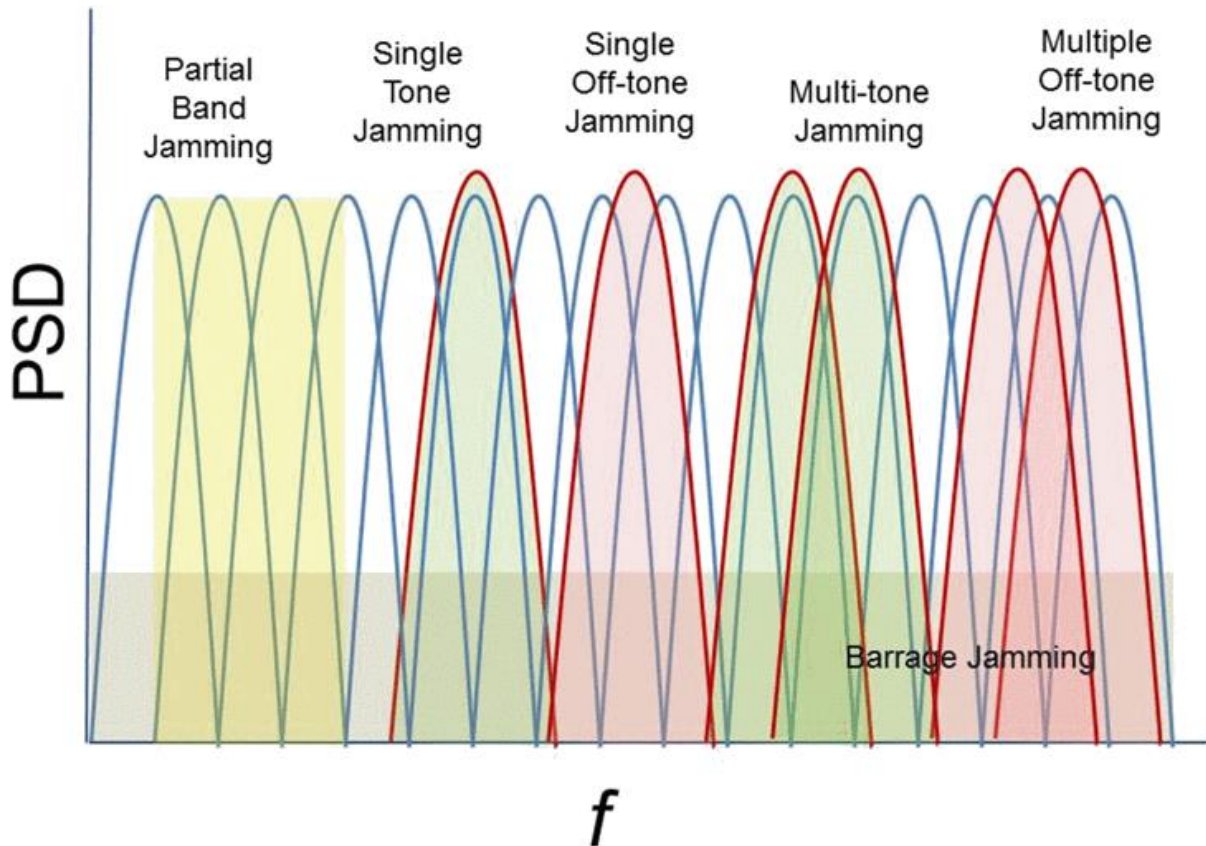


Fig. 8 - Visualization of the different frequency spectrum for different jamming attacks [19]

3.1.1 Wideband / Barrage Jamming (BNJ)

The simplest jamming attack, BNJ simply generates signal noise in the entire frequency spectrum of the OFDM signal that it is intended to jam. The noise generated is an AWGN, which is white noise with a uniform normal distribution. Since the noise generated by the BNJ covers the entire frequency spectrum of the OFDM signal, the variance of the signal increases since the signal's overall strength in dB has been raised. The overall noise increases (aside from background noise), which decreases the SINR. [19]

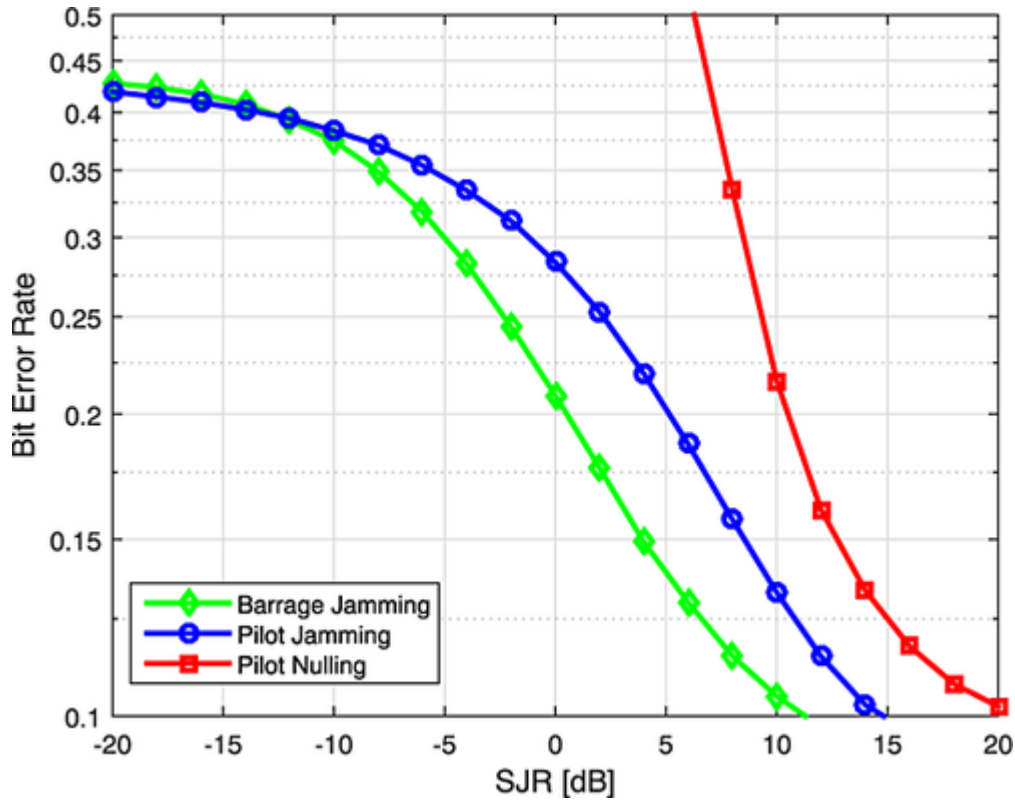


Fig. 9 - Performance of Barrage Jamming (BNJ), quantified by BER [20]

BNJ is shown to have a noticeable effect on OFDM signals, causing significant BER. With reference to Fig. 9, when the SJR is 0 dB when the target is operating at a SINR of 5 dB, the BER becomes about 0.21, meaning there would be about 21 bits that become unreadable by the receiver for every 100 bits. At 6 dB SJR, the BER becomes about 0.13. [20]

3.1.2 Partial Band Jamming (PBJ)

This jamming attack only generates additive gaussian noise in a much smaller frequency band, not affecting all the frequencies used by the OFDM signal but only a smaller subset of frequencies. However, as the range of frequencies emitted by this jamming signal is much lesser, the power of the noise generated at these frequencies can be larger without consuming more power than barrage jamming. PBJ signals also have two frequency bands in which they emit signals, one is the frequency band that overlaps a small part of the OFDM signal, the other is a frequency band that falls completely outside the frequency band of the OFDM signal. [19]

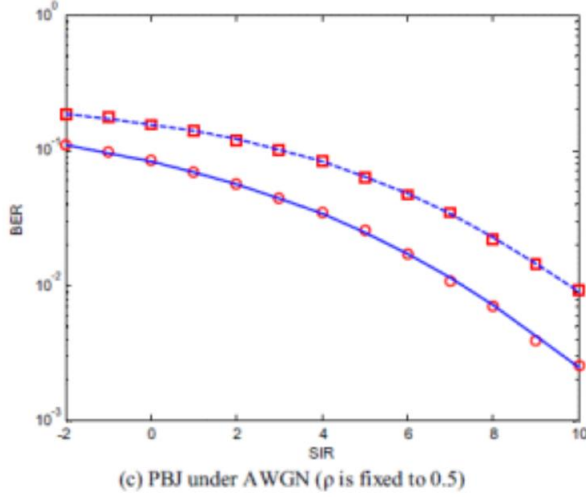


Fig. 10 - Performance of Partial Band Jamming (PBJ) quantified in BER [21]

PBJ is shown to have a noticeable effect on OFDM signals, causing significant BER. With reference to Fig. 10, when the SJR is 0 dB when at a the OFDM signal was SINR of 10 dB, the BER becomes 0.08. At 6 dB SJR, the BER becomes 0.018. [21]

3.1.3 Pilot Jamming (PJ)

This equalization attack specifically targets the pilot subcarriers in the OFDM signals, thus making the determination of CSI very difficult. The attacker would transmit AWGN at the frequency bands of the pilot subcarriers to add significant noise to the pilot subcarrier signals, hopefully making it unreadable by the receiver. The effect of this makes the receiver unable to determine CSI accurately, thus not knowing whether to apply any frequency offsets or accommodate for phase noise. Some of the bits transmitted over the air thus cannot be appropriately corrected, resulting in bit errors. [22]

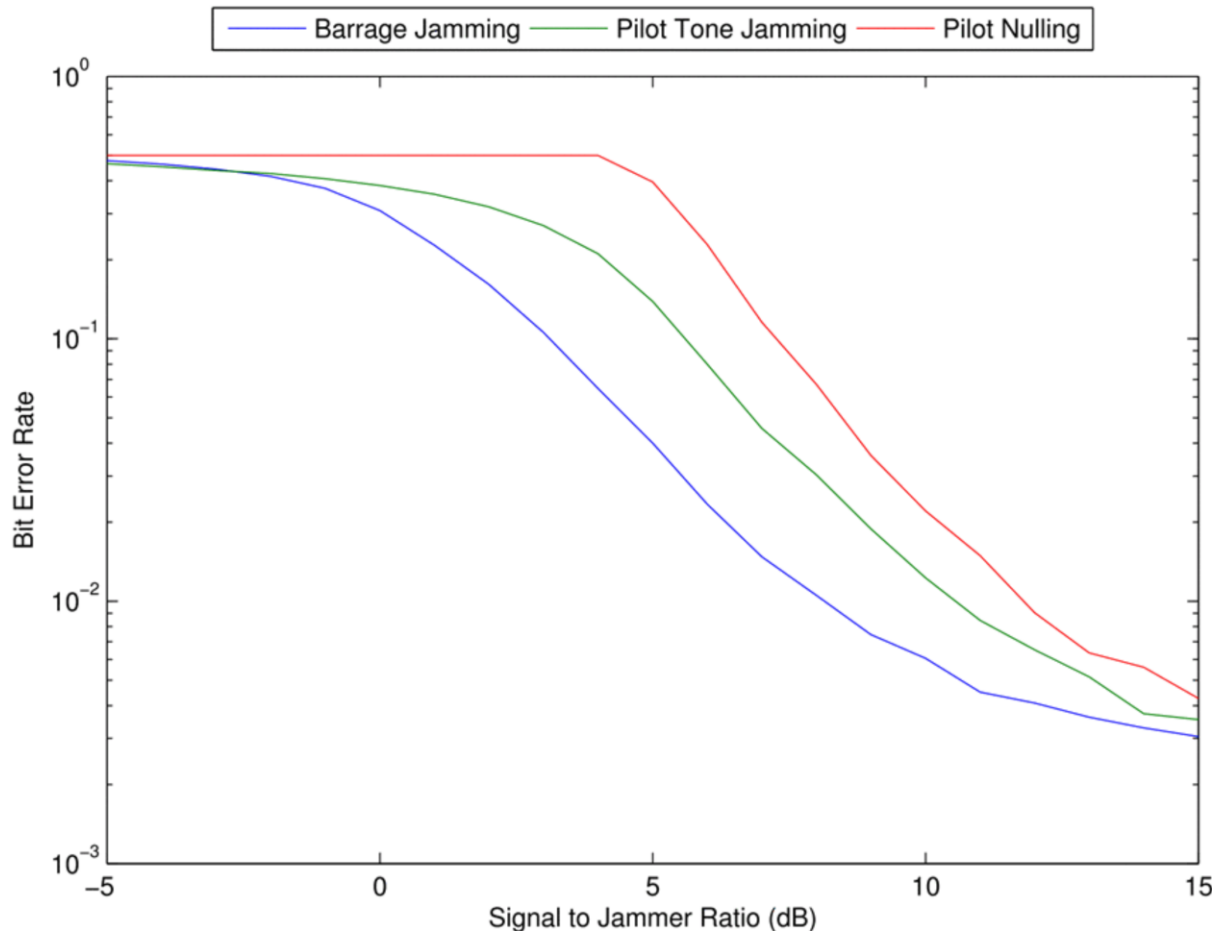


Fig. 11 - Performance of Pilot Jamming (PJ) and Pilot Nulling (PN) quantified in BER [22]

Because of the need to know what the frequency bands of the pilot subcarriers are, PJ has a higher complexity, and the attacker requires enough understanding of the CSI. However, it is shown that PJ has much better success in disrupting the wireless communications. With reference to Fig. 11, when the OFDM signal was operating at a 20 dB SINR, the BER for PJ is 0.4 compared to 0.3 for BNJ when the SJR is 0 dB. [22]

3.1.4 Pilot Nulling (PN)

Also an equalization attack, PN is a more destructive version of PJ, which completely nullifies the pilot subcarrier signals by sending a destructive waveform to neutralize the pilot subcarrier signals, attempting to make the signal as close to zero as possible by perfectly cancelling the entire waveform of the pilot subcarriers. It does this by sending the same pilot subcarrier waveforms to the receiver but offsetting them by π radians. [22] The sending of the π -phase shifted pilot tones is possible because the data used in these pilot tones is always constant, for the receiver to establish the CSI.

This makes the frequency response of the pilot subcarrier signals near zero, resulting in a large margin of error in calculating the equalized signal.

Because of the need to know what the frequency bands of the pilot subcarriers are, as well as needing to capture the pilot subcarriers to be retransmitted by the attacker, PN is much more complex. However, PN is shown to be even more destructive than PJ or BNJ. Referring to Fig. 11, PJ causes a BER of 0.2 compared to 0.08 for PJ and 0.02 for BNJ at 6 dB SJR when the OFDM signal was operating at a 20 dB SINR. [22]

3.1.5 Multi-Tone Jamming (MTJ)

MTJ is a form of jamming which is much more focused than PBJ, only sending a few high-powered tones to the receiver to jam the signal. The power of each of the tones is the same. The jamming tones generated are synchronous with the subcarrier frequencies they intend to jam, which means the fundamental frequency of each of the tones is ideally the same as that of the subcarrier they intend to jam, as shown in Fig. 8 previously. [19]

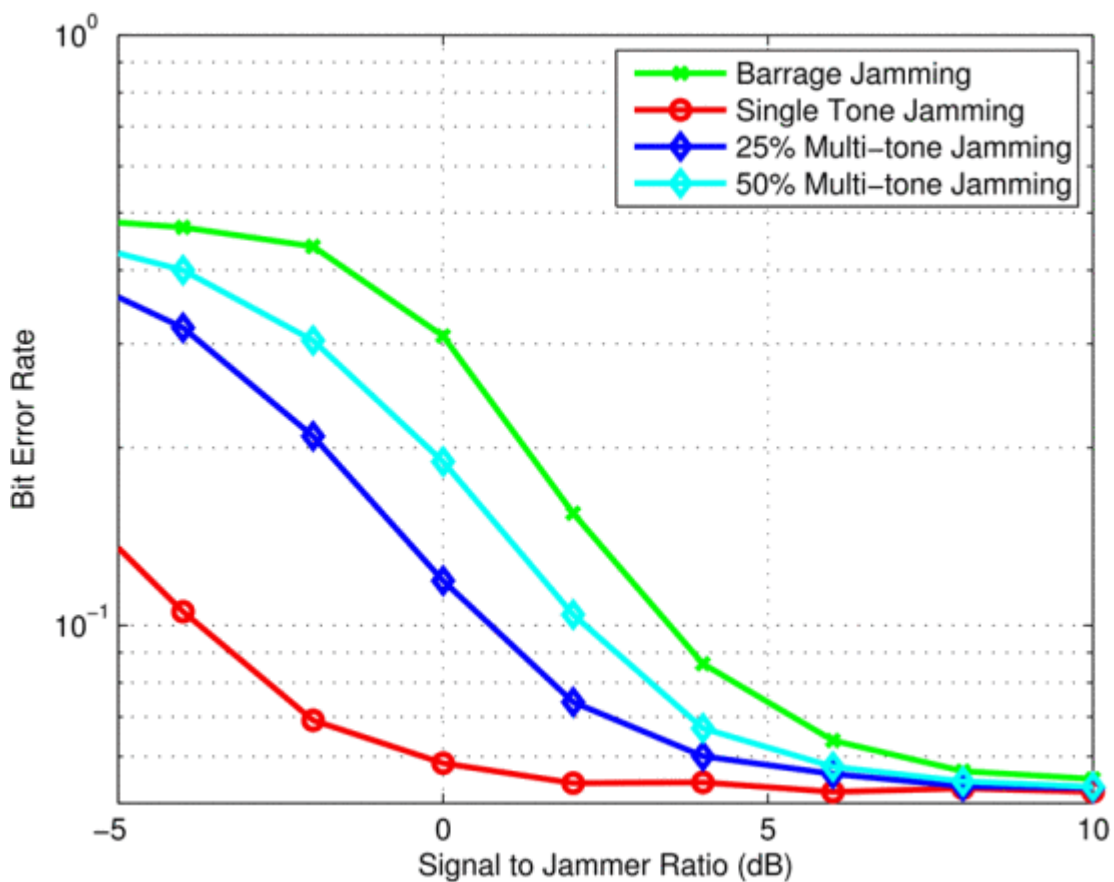


Fig. 12 - Performance of Multi-Tone Jamming (MTJ) quantified in BER [19]

MTJ is slightly less effective than BNJ, only causing a BER of 0.12 while BNJ causes a BER of 0.3, when the SJR is 0 dB and the OFDM signal is operating at an initial condition of 5 dB SINR as shown in Fig. 12. [19]

3.1.6 Asynchronous Off-Tone Jamming

Like MTJ, this jamming technique works by sending only a few high-powered tones to the receiver, with each of the tones having the same power. However, the jamming tones generated are instead not synchronous, meaning their fundamental frequency is offset from any 2 adjacent subcarriers, as shown in Fig. 8 previously. The asynchronous nature of these jamming tones is shown to be effective than MTJ, because these tones disrupt the orthogonality of OFDM signals by causing ICI. Fig. 13 below shows how ICI occurs with asynchronous jamming tones. The waveform with a higher peak indicates the jamming waveform while the other 2 waveforms are that of the subcarriers. [19]

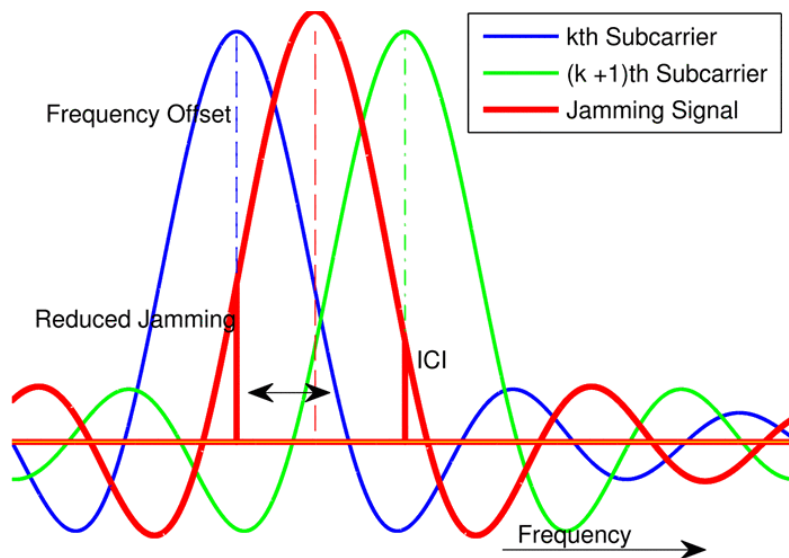


Fig. 13 - ICI caused by asynchronous jamming tones [19]

Because of the asynchronous nature of the tones being generated, this jamming technique is not very complex and does not require full knowledge of the CSI. This technique is shown to be more effective than BNJ. However, for more destructive results, it is more desirable for the fundamental frequency of the tones generated to be in the middle of its 2 adjacent subcarrier fundamental frequencies.

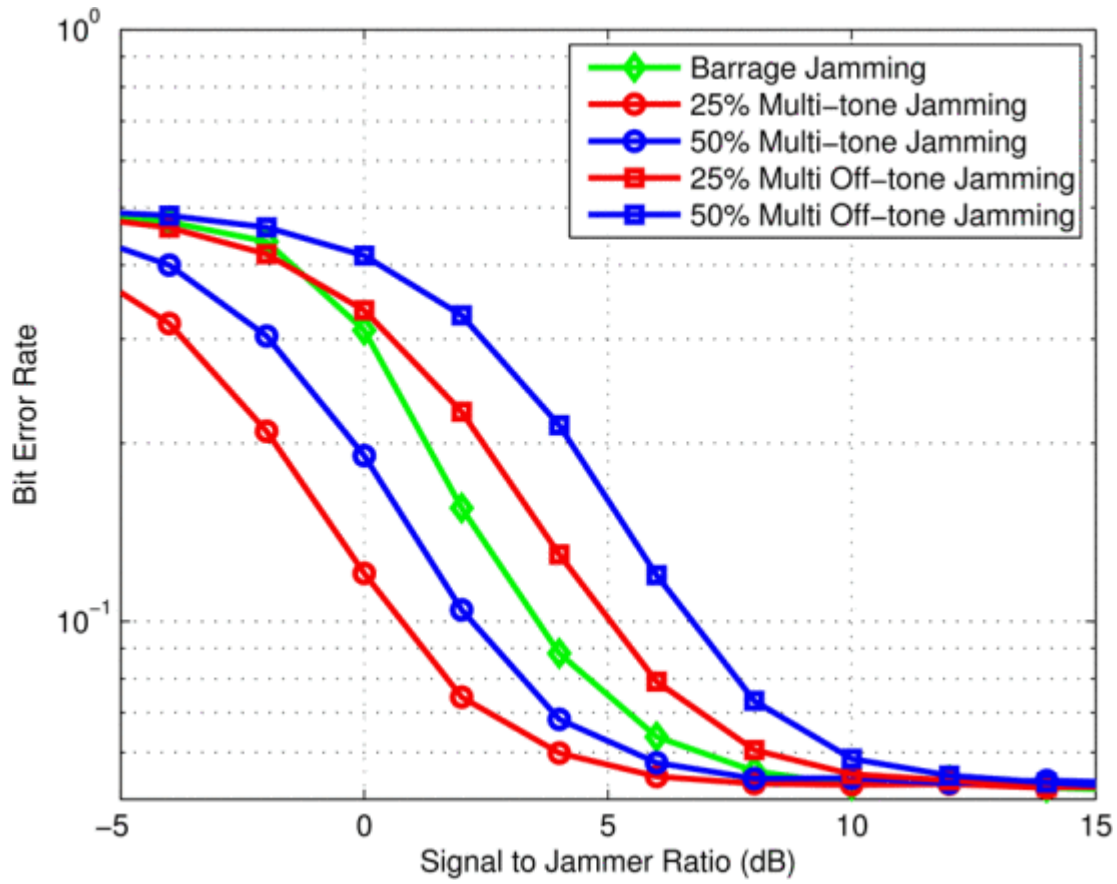


Fig. 14 - Performance of asynchronous off-tone jamming quantified in BER [19]

With a 25% subcarrier jamming (Or -12 dB subcarrier jamming tone signal to subcarrier signal ratio), which refers to the “Reduced Jamming” portion of the jamming tone signal as shown in Fig. 13, this technique causes a BER of 0.22 for an OFDM signal operating at 5 dB SINR under 3 dB SJR as shown in Fig. 14. In Fig. 14, with a 50% subcarrier jamming (Or -6 dB subcarrier jamming tone signal to subcarrier signal ratio), this technique causes a BER of 0.32 for an OFDM signal operating at 5 dB SINR under 3 dB SJR. This contrasts with BNJ which only causes a BER of 0.16. [19]

3.1.7 False Preamble Timing Attack

The preamble sent by the transmitting node allows for frequency lock, identification of the start OFDM symbol and channel estimation. [23] This attack focuses on creating a new timing metric peak. The attacker uses data on the preamble to either retransmit the same preamble or creating a different preamble symbol. The attacker's preamble is transmitted at an equal or higher power, such that the receiver will either take both preambles to estimate the wrong timing or take the attacker's preamble respectively, in both cases causing the wrong timing. If transmitting a completely new preamble, the preamble symbol is then constructed with any pseudo-random sequence. [15]

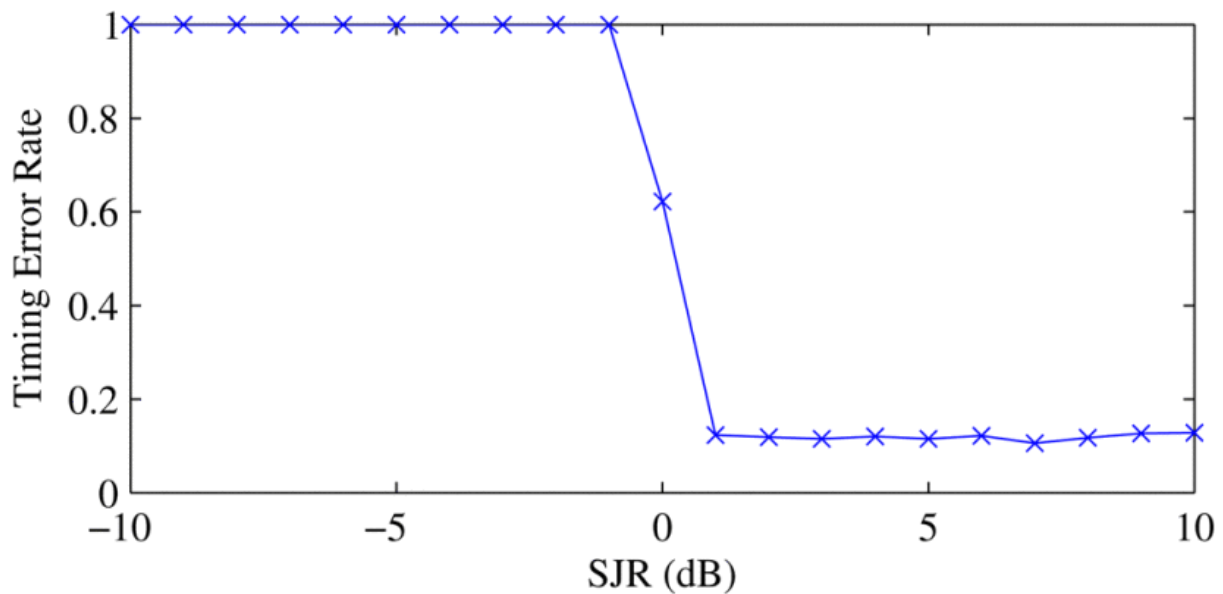


Fig. 15 - Performance of False Preamble Timing Attack quantified in BER [15]

As the receiver only takes the preamble of the attacker into account when the power of the jamming attack is either equal or higher than the original preamble signal's power, the BER for false preamble timing attack quickly drops and flatlines after a certain point. With reference to Fig. 15, the BER is 1.0 for an SJR of -1 dB and below. The BER is 0.6 for an SJR of 0 dB, and then quickly drops and flatlines to 0.1 for an SJR of 1 dB and above. [15]

3.1.8 Preamble Phase Warping

Between the transmitting node and the receiving node, due to intrinsic hardware differences, there will be differences in the operating frequencies of the two nodes. This requires a FO correction, which is addressed in a broadcasted preamble to every frame before transmission.

One of the possibilities this jamming technique could work is to jam a small part of the preamble, the part which is responsible for FO estimation. The attacker would need to roughly estimate the FO and quickly intercept the frames transmitted. The jamming signal would then change the value of the FO significantly such that the receiver would interpret the subcarriers received wrongly, thus scrambling the data received. This results in massive ICI and subsequent degradation of SINR. The jamming signal could be simply AWGN, or it could be a smart false preamble if the attacker can estimate the FO.

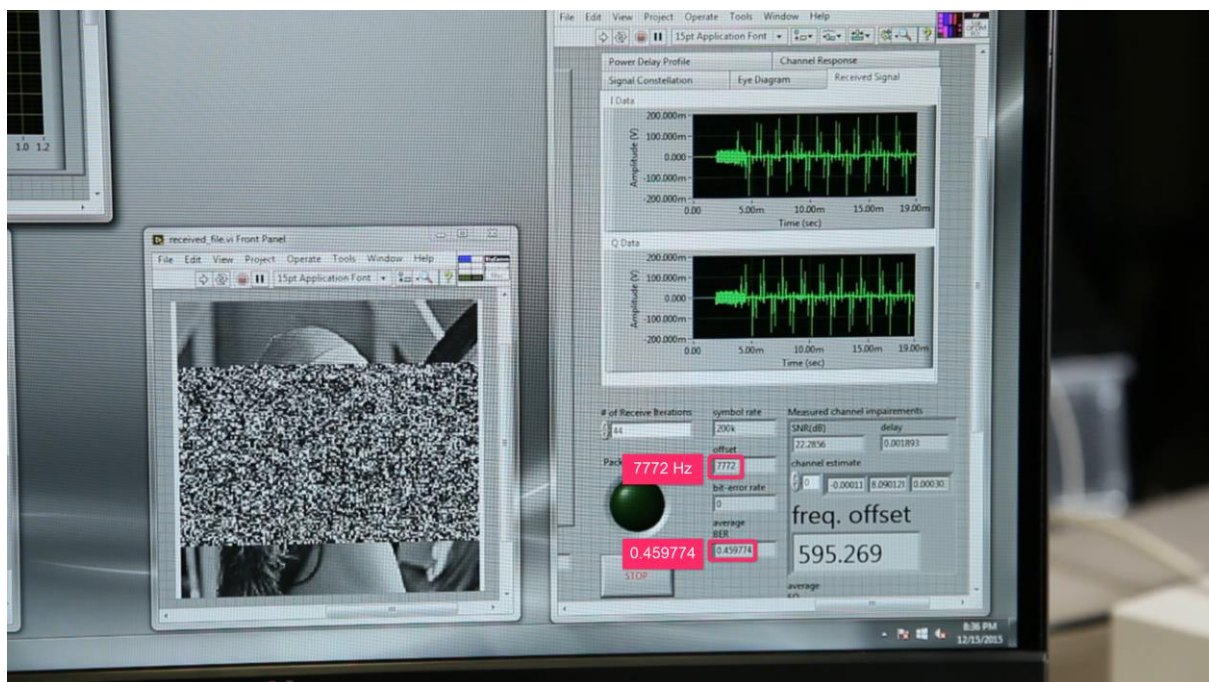


Fig. 16 - Destructive Effects of Preamble Phase Warping, with the frequency offset after the jammer is active highlighted (the normal offset without the presence of a jammer is indicated in the freq. offset box) and the BER of the attack [24]

This attack has a success of causing a BER of up to 0.5 as shown in Fig. 16. [24]

3.1.9 Differential Scrambling Attack

This attack disrupts the coarse frequency error estimation at the receiving node. This error is a subcarrier misalignment at the receiving node due to clock frequency discrepancies. A constant stream of symbols across the subcarriers used in the first preamble symbol would be transmitted. Despite being similar in structure to the false preamble timing attack, this attack corrupts the amplitude and phase of the received subcarriers in the first preamble symbol, changing the differential sequence at the receiver. The symbols transmitted on each subcarrier would be the same and derived from the fact that the pseudo-random sequence of the first preamble symbol is unknown. This corruption of the sequence using the symbols sent by the attacker will diminish the performance of the coarse frequency estimation and can result in subcarrier misalignment at the receiver. [25]

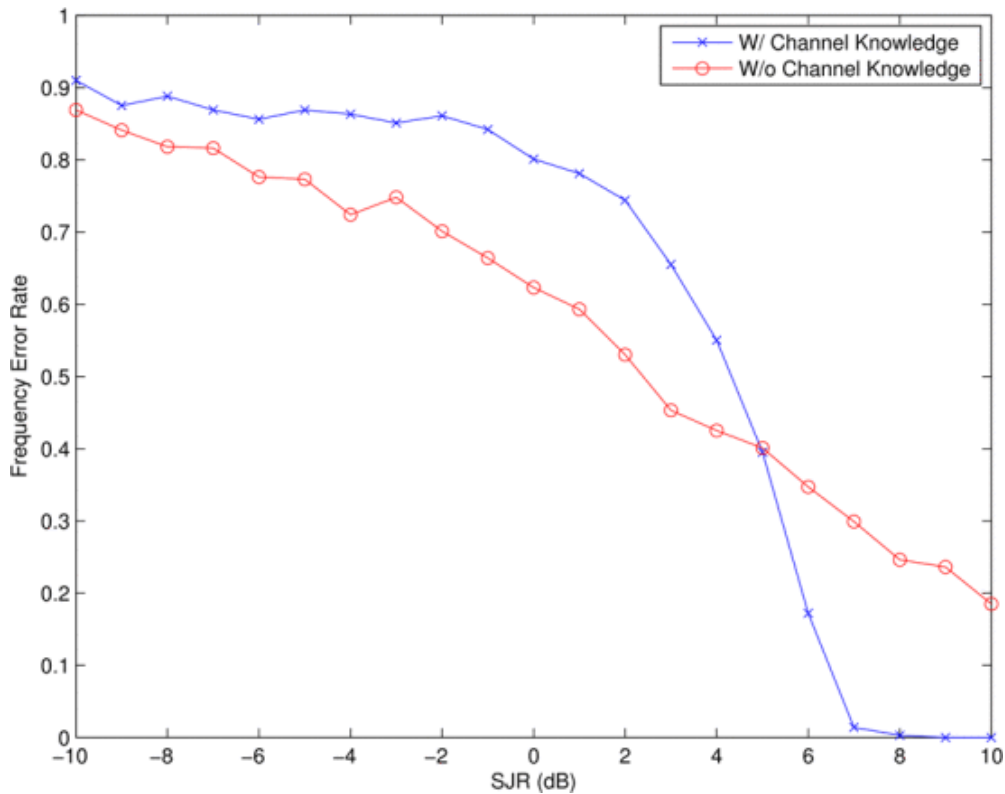


Fig. 17 - Performance of Differential Scrambling Attack quantified in BER [25]

The differential scrambling attack is shown to be effective, causing a BER of 0.4 at an SJR of 5 dB as shown in Fig. 17. [25]

3.1.10 Cyclic Prefix (CP) Jamming

In OFDM, a CP is used. The CP repeats a small portion of the trail end of the data in the subcarrier, prefixed to the front of the OFDM symbol of the subcarrier. This CP allows channel equalization to be carried out and is also used as a guard interval to eliminate any possible ISI. Channel equalization is used for the receiver to obtain CSI. The CP is also used for correcting the symbol-time delay and the carrier frequency offset of the signal by correlating the repeated parts of a symbol. [15]

This jamming technique works by only jamming the CP portion of the subcarriers. This saves considerable power rather than jamming the entire subcarrier. This technique is shown to be effective with a BER of 0.0168 compared to BNJ which was 0.0148, and a base case where there are no jamming attacks with a BER of 0.000524. [26]

3.2 Past Research on OFDMA

As OFDMA is derived from OFDM, the attacks in OFDM can also be used on OFDMA.

However, there are other attacks which are more applicable to OFDMA due to its Multiple Access property which allows for some additional forms of attacks.

3.2.1 HARQ ACK Attack

OFDMA systems use an error control mechanism which is known as ARQ. In the case of HARQ, it uses both ARQ and channel coding which accounts for forward error correction.

This mechanism involves the receiver transmitting either a positive or negative ACK indicators to the transmitting node. These ACKs trigger a retransmission when the forward error correction has insufficient redundancy to correct the channel errors. [15]

This attack works by disrupting the ARQ mechanism by targeting the ACK indicators. When ACKs are sent back from the receiver to the transmitter, there is no error checking. This means that corrupting the ACK signal may cause either unnecessary or delayed retransmissions. [15]

The authors of [15] estimated that a BER of 0.1 is enough for these ACKs to be corrupted to cause such unnecessary or delayed retransmissions. For such a BER, the authors estimate that an SJR of 1 dB is enough. The jamming attack used could be a simple AWGN.

This attack not only causes unnecessary or delayed transmissions, but it also has the capability to compromise the confidentiality of the communication, since the retransmitted packets can be eavesdropped on in order to have higher chances to decode the information. In fact, there is a probability of 0.33 that eavesdropping would be successful when the jammer power is at 6 dB. This probability increases to 0.35 when the jammer is at a power of 12 dB. [27]

3.2.2 Resource Allocation Attack

OFDMA's main advantage over OFDM is the capability to serve multiple users at the same time, allocating them varying widths of frequency bands. This means the need to allocate users based on the time and frequency domains. This allocation must be done extremely quickly, since data packets these days are getting smaller but a lot more frequent. The allocation of frequency bands is dependent on the channel conditions seen by each node, so a rapidly changing channel will see resources being re-allocated rather frequently. Each node would receive information on which frequency bands it is being allocated, which is transmitted in a short duration and can be as little as 1 millisecond. The resource allocation information is transmitted on the same physical channel as the downlink modulation indicators, thus making it possible to destroy both the resource allocation information and the downlink modulation indicators with the same technique of jamming. [15]

The degradation of resource allocation information causes denial of service as nodes are unable to properly interpret the data received. Thus, they cannot determine which data in which frequency bands are meant for them. [15]

3.2.3 Random Access Channel Attack

Random access requests are used by nodes for them to be able to retrieve information from the base station that they are connected to. This request exerts the presence of the node and is used to ask for a portion of the channel to be allocated to it. They are sent in a specially allocated bandwidth using a contention-based access scheme. This scheme involves having no coordination between the different nodes who transmit their requests to the base station, which causes collisions occasionally. To reduce collision, each node picks and transmits a randomly chosen sequence from a set of sequences that have low correlation with each other. This allows the base station to interpret two or more overlapping random access requests. It will then reply the resource allocation information to the nodes. [15]

This random access channel attack jams the specially allocated frequency bands used to transmit these requests. The jammer simply transmits AWGN in order to make the base station unable to interpret such requests appropriately, effectively denying nodes from being able to communicate. As the SJR increases, the probability of the base station detecting a sequence wrongly increases. When the base station misinterprets the sequence, the base station will still reply to it, but the reply is not useful for any of the nodes to initiate communications. [15]

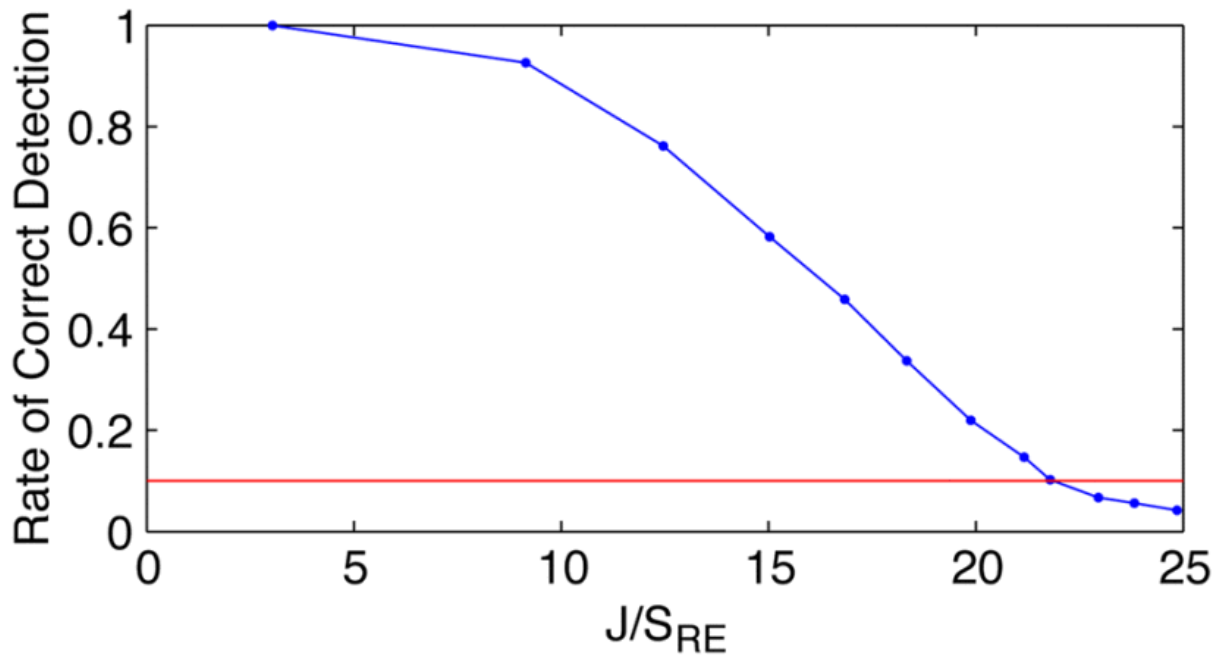


Fig. 18 - Performance of Random Access Channel Attack quantified in Rate of Correct Detection

Fig. 18 shows that random access channel attack causes a BER of 0.8 at an SJR of -20 dB, a BER of 0.45 at an SJR of -15 dB. [15] Note that $BER = 1 - (\text{Rate of Correct Detection})$ and $SJR = - (J/S_{RE})$ since J/S means jammer-to-signal ratio, which is simply the reciprocal of signal-to-jammer. When converted to dB, this means multiplying by -1.

CHAPTER 4 – Design and Implementation

4.1 Apparatus used in Experiment

4.1.1 Transmitter Node

The client transmitting the file is a laptop of model ASUS ZenBook UX303LN. This is of no matter because the computer is connected via a LAN Cable to a router. The wireless transmitting node is the router. The router is a TP-Link Archer AX10 Wi-Fi 6 AX1500 Gigabit Router with 1.5GHz Triple-Core



Fig. 19 - The Archer AX10 Wi-Fi 6 AX1500 Gigabit Router with 1.5GHz Triple-Core CPU used as the transmitter node [28]

CPU as shown in Fig. 19. Listed below are the technical specifications of the router [28]:

- 2×2 antenna arrangement
- Speeds of up to 1201 Mbps with Wi-Fi 6 in the 5 GHz operating range, compared to 1000 Mbps for Gigabit Ethernet, 867 Mbps for 2×2 antenna arrangement for Wi-Fi 5 solution, 433 Mbps for 1×1 antenna arrangement for Wi-Fi 5 solution
- 1024-QAM with higher symbol rate
- OFDMA support
- 1.5 GHz Triple-Core CPU compared to 0.9 GHz Single-Core CPU in traditional 2×2 AC routers
- Supports Beamforming

4.1.2 Receiver Node

The receiver node is a laptop equipped with a Wi-Fi 6 enabled card. The computer model used is a Dell XPS 9560, with its Wi-Fi card swapped out to the Wi-Fi 6 enabled card. With both transmitter and receiver node supporting Wi-Fi 6, the wireless protocol used becomes of standard 802.11ax, which is the latest Wi-Fi 6. The Wi-Fi card used



Fig. 20 - Dual Band Wireless AX200NGW 2.4 Gbps 802.11ax Wireless Intel AX200 Wi-Fi card with Bluetooth 5.0 for Windows 10, used as the receiver node [35]

is a Dual Band Wireless AX200NGW 2.4 Gbps 802.11ax Wireless Intel AX200 Wi-Fi card with Bluetooth 5.0 for Windows 10 as shown in Fig. 20. The laptop used is not important since the most important part is that the Wi-Fi Card can function well on the laptop. Listed in Table 2 are the technical specifications of the Wi-Fi card [29]:

Table 2 - Technical Specifications for Dual Band Wireless AX200NGW 2.4 Gbps 802.11ax Wireless Intel AX200 Wi-Fi card with Bluetooth 5.0 for Windows 10, used as the receiver node [29]

General	
Dimensions (Height × Width × Depth)	22mm × 30mm × 2.4mm [1.5mm Max (Top Side)/ 0.1mm Max (Bottom Side)]
Weight	2.83 ± 0.3 g
Connector Interface	M.2: PCIe, USB
Operating Temperature (Adapter Shield)	0°C to +80°C
Humidity Non-Operating	50% to 90% RH non-condensing (at temperatures of 25°C to 35°C)
Operating Systems	Microsoft Windows 10, Linux, Chrome OS
Wi-Fi Alliance	Wi-Fi CERTIFIED a, b, g, n, ac with wave 2 features, WMM, WMM-Power Save, WPA, WPA2, WPS, PMF, Wi-Fi Direct, Wi-Fi Miracast, Wi-Fi Agile Multiband, Wi-Fi Optimized Connectivity, Wi-Fi Location, Passpoint, Wi-Fi Aware, and Wi-Fi TimeSync
IEEE WLAN Standard	IEEE 802.11-2016 and select amendments (selected feature coverage) IEEE 802.11a, b, g, n, ac, ax, d, e, h, i, k, r, u, v, w, ai; Fine Timing Measurement based on 802.11-2016

Security Features	
Security Methods	WPA and WPA2 Personal and Enterprise; WPA3 (pending OS support)
Authentication Protocols	802.1X EAP-TLS, EAP-TTLS/MSCHAPv2, PEAPv0 - MSCHAPv2 (EAP-SIM, EAP-AKA, EAP-AKA')
Encryption	64-bit and 128-bit WEP, TKIP, 128-bit AES-CCMP, 256-bit AES-GCMP
Networking Specifications	
TX/RX Streams	2 × 2
Bands	2.4GHz, 5GHz (160MHz)
Max Speed	2.4Gbps
Wi-Fi CERTIFIED	Wi-Fi 6 (802.11ax)
Advanced Technologies	
MU-MIMO	Yes
Orthogonal Frequency-Division Multiple Access (OFDMA)	Yes

4.1.3 Jamming Device

The jammer node used is a signal generator.

The model of the signal generator used is a R&S®SMW200A vector signal generator as shown in Fig. 21.



Fig. 21 – Rohde & Schwarz SMW200A Vector Signal Generator [36]

It can generate AWGN signals of varying bandwidths and centre frequencies, which is

enough to conduct the jamming attacks being tested. It has the capability to create signals up to 6 GHz with the R&S®SMW-B1006 option installed on one of its RF transmitters, which is enough for the experiment.

The R&S®SMW200A vector signal generator is highly customizable with various options that can be installed based on the user's demands. The other customizable options will not be discussed since it is not relevant to the experiment, but can be found in its specification document: <https://d3fdwrtpsindh7j.cloudfront.net/Docs/datasheet/SMW200A.pdf>

4.2 Experiment Procedure & Set-Up

4.2.1 Procedure

UDP is used to transfer the file through the air. UDP is used because it has no CRC and it will not retransmit in the case of bit errors received at the receiver node. This is because the presence of jamming will cause repeated retransmissions, which can occur infinitely given a high enough BER. Over-the-air transmission occurs only between the Wi-Fi 6 enabled router and the receiver node, but not between the transmitter and router.

A very simple UDP program is used to get the test file transmitted. ACKs will not be used in transmissions. This is because the presence of jamming may also cause ACKs to be corrupted and cause unnecessary retransmissions. In the case of very high BER, it may even cause the transmission to fail completely and be unable to collect any results.

4.2.2 Quantification of Results

The file used for sending is a 50 KB file which is randomly generated by the Linux's kernel random number generator. The random number generator gathers environment noise from device drivers and other sources into an entropy pool. [30] The generated file contains nonsensical information which makes absolutely no sense, but it is of no matter since BER is determined by the number of bits that have changed in the file and not the actual information itself. The command to generate the random file is as follows:

```
head -c 51200 </dev/urandom > myfile.txt
```

50 KB is big enough to pick up errors under very small BERs. The 50 KB file is placed in both the transmitter and receiver nodes. The 50 KB file which is placed in the receiver node serves as the perfect file copy under absolute zero BER in transmission. When the transmitter sends the 50 KB file over the air, this received file is then compared, byte-by-byte to the 50

KB file which has been already placed in the receiver node. Since all data in real world applications are interpreted by bytes (since data is interpreted by characters, which is one byte long), rather than down to each individual bit, it would suffice to compare the different bytes between the 2 files by bytes and quantify BER using the number of bytes that differ from the original file. The number of bytes that differ from the original 50 KB file is determined, and from there, BER can be calculated.

Because no environment used is perfect and free from environmental noise, the experiment is repeated 10 times and averaged. The formula used to calculate BER is as shown below.

Equation 1 - Calculation of BER

$$\begin{aligned}
 BER &= \sum_{x=1}^{10} \frac{(\text{Number of different bytes from original file for trial } x)}{\text{Total number of bytes in the 50 KB file}} \\
 &= \sum_{x=1}^{10} \frac{(\text{Number of different bytes from original file for trial } x)}{51200} \\
 &= \frac{\sum_{x=1}^{10} (\text{Number of different bytes from original file for trial } x)}{512000}
 \end{aligned}$$

4.2.3 Wireless Connection Environment Set-Up

The full codes for the UDP transfer for both the transmitter and receiver are included in APPENDIX A – C Source Code used for UDP sending and receiving.

The IP Address of the receiver is first recorded such that the transmitter can send the 50 KB file over to the receiver. This can be done by running the following command on the receiver's Linux terminal console:

```
hostname -I
```

This IP Address will then be used as the input field used when the transmitter sends the file. The IP Address of the receiver is found to be 192.168.0.223.

4.2.4 Operating Parameters of Transmitter and Receiver

This experiment was conducted in Singapore. Different countries have slightly varying allowed frequency bands for Wi-Fi signals. The IMDA is advised by TSAC in setting the ICT standards as well as on the development and recommendation of specifications, standards, information notes, guidelines and other forms of documentation for adoption and advancement of the standardisation effort of the Singapore ICT industry.

According to IMDA, the allowed frequency bands for Wi-Fi (what they refer to as Wireless LAN in their document) in Singapore to operate are in the range of 2.4000 – 2.4835 GHz, 5.725 – 5.850 GHz, 5.150 – 5.350 GHz and 5.470 – 5.725 GHz. However, the frequency bands 5.250 – 5.350 and 5.470 – 5.725 GHz are required to employ DFS and TPC. The power usage of all wireless signals including Wi-Fi signals are also being regulated but will not be discussed as it is not relevant to this research. [31]

A network monitoring tool called Homedale (<http://www.the-sz.com/products/homedale/>) is used to find out the operating frequency band of the Wi-Fi 6 router. The SSID of the Wi-Fi 6 dual band router is named Venu5. The other network SSID, Venus, is the SSID of the same router, but transmits in the 2.4 GHz range and does not use the 802.11ax protocol, but the 802.11ac (Wi-Fi 5) protocol.

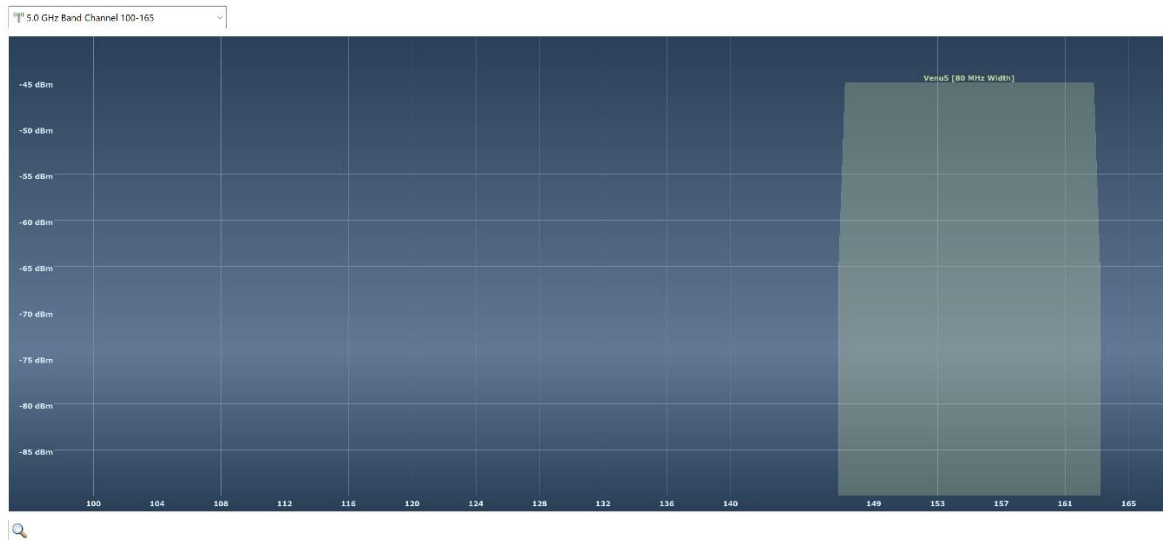


Fig. 22 - Frequency Spectrum of the Wi-Fi 6 router used as transmitter node

As seen in Fig. 22, this Wi-Fi 6 router occupies the channels 149, 153, 157 and 161. This corresponds to the frequency band 5.735 – 5.815 GHz. These channels are all 20 MHz wide.

```

SSID 2 : Venu5
Network type      : Infrastructure
Authentication    : WPA2-Personal
Encryption        : CCMP
BSSID 1           : 74:da:88:93:5a:2a
Signal            : 91%
Radio type        : 802.11ax
Channel           : 157
Basic rates (Mbps) : 6 12 24
Other rates (Mbps) : 9 18 36 48 54

```

Fig. 23 - Additional Details about the Wi-Fi 6 router used as transmitter node

Access Point	MAC Address	Vendor	Signal Strength	Signal Strength History	Frequency	Band	Bitrates
Venu5	74:DA:88:93:5A:2A	TP-Link Technologies Ltd	-40 dBm		Ch 157 [5.785 GHz] + Ch 161, 5.775 GHz [80 MHz Width]	5.0 GHz	866.5, 433, 300, 270, 240, 180, 150, 135, 120, 90, 60, 54, 48, 45, 36, 30, 24, 18, 15, 12, 9, 6 Mbps
Venus	74:DA:88:93:5A:2B	TP-Link Technologies Ltd	-32 dBm		Ch 8 [2.447 GHz]	2.4 GHz	300, 270, 240, 180, 150, 135, 120, 90, 60, 54, 48, 45, 36, 30, 24, 18, 15, 12, 11, 9, 6, 5.5, 2, 1 Mbps

Fig. 24 - Additional Details about the Wi-Fi 6 router used as transmitter node

More details on the Wi-Fi 6 router being used, including its MAC Address and signal strength, are as shown in Fig. 23. As shown under the channel field, channel 157 is currently the main channel used for the transmission of data in the absence of a jammer.

The transmitting power of the wireless router is 23 dBm, as is the case with most wireless routers. [32] This degrades due to many factors such as antenna gain of both the router and the receiver, the distance between the router and the receiver and the path loss, which explains why the received wireless signal power is -40 dBm.

```

There is 1 interface on the system:

Name           : Wi-Fi 2
Description    : Intel(R) Wi-Fi 6 AX200 160MHz
GUID           : 
Physical address : 24:
State          : connected
SSID           : Venu5
BSSID          : 74:da:88:93:5a:2a
Network type   : Infrastructure
Radio type     : 802.11ax
Authentication : WPA2-Personal
Cipher         : CCMP
Connection mode : Profile
Channel        : 157
Receive rate (Mbps) : 1201
Transmit rate (Mbps) : 1201
Signal         : 91%
Profile        : Venu5

```

Fig. 25 - Installed Wi-Fi Card Details

Understanding the frequency spectrum of the Wi-Fi router allows us to know which frequencies to jam. In this experiment, 4 types of jamming against OFDMA enabled Wi-Fi 6 will be investigated, which are BNJ, PBJ, MTJ and asynchronous off-tone jamming.

4.2.5 Operating Parameters of Jammer

Next, the signal generator is set up to emit AGWN in order to perform the jamming attacks.

Fig. 26 below shows the general set-up of the signal generator:

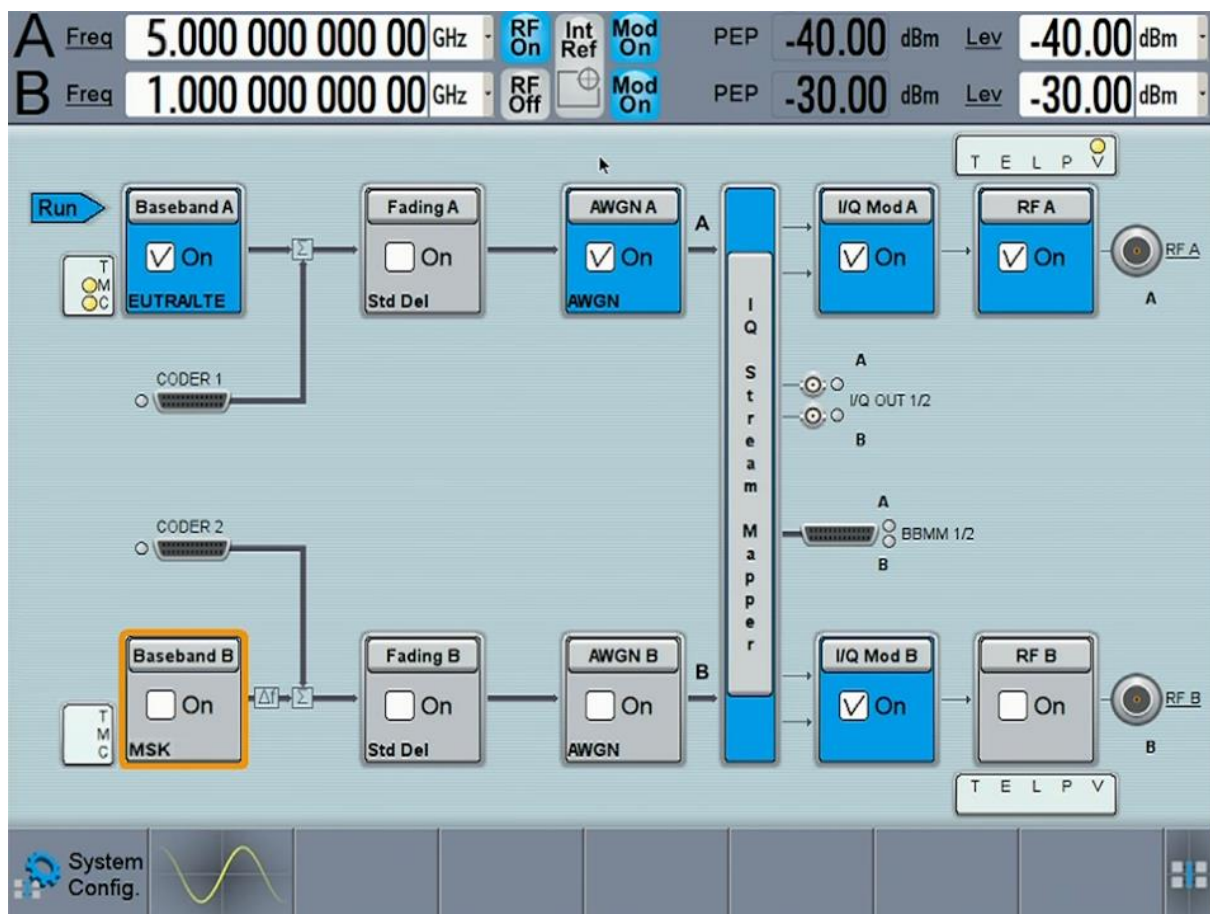


Fig. 26 – General Set-up of the Signal Generator for generating AWGN

The signal generator supports up to 2 transmitters, however, we only require one source of signal. Thus, the second RF option is switched off, which is RF B.

However, when generating 2 AWGN signals of different centre frequencies, the set-up of the signal generator would differ slightly. Generating 2 AWGN signals would be required for MTJ and asynchronous off-tone jamming. The frequency of the second AWGN signal is set through the frequency offset option which is accessed by tapping Δf to the right of the Baseband B block, and then setting the frequency offset for Baseband B.

The blocks are then reconnected as follows to route both AWGN signals into the transmitter at RF A as shown in Fig. 27.

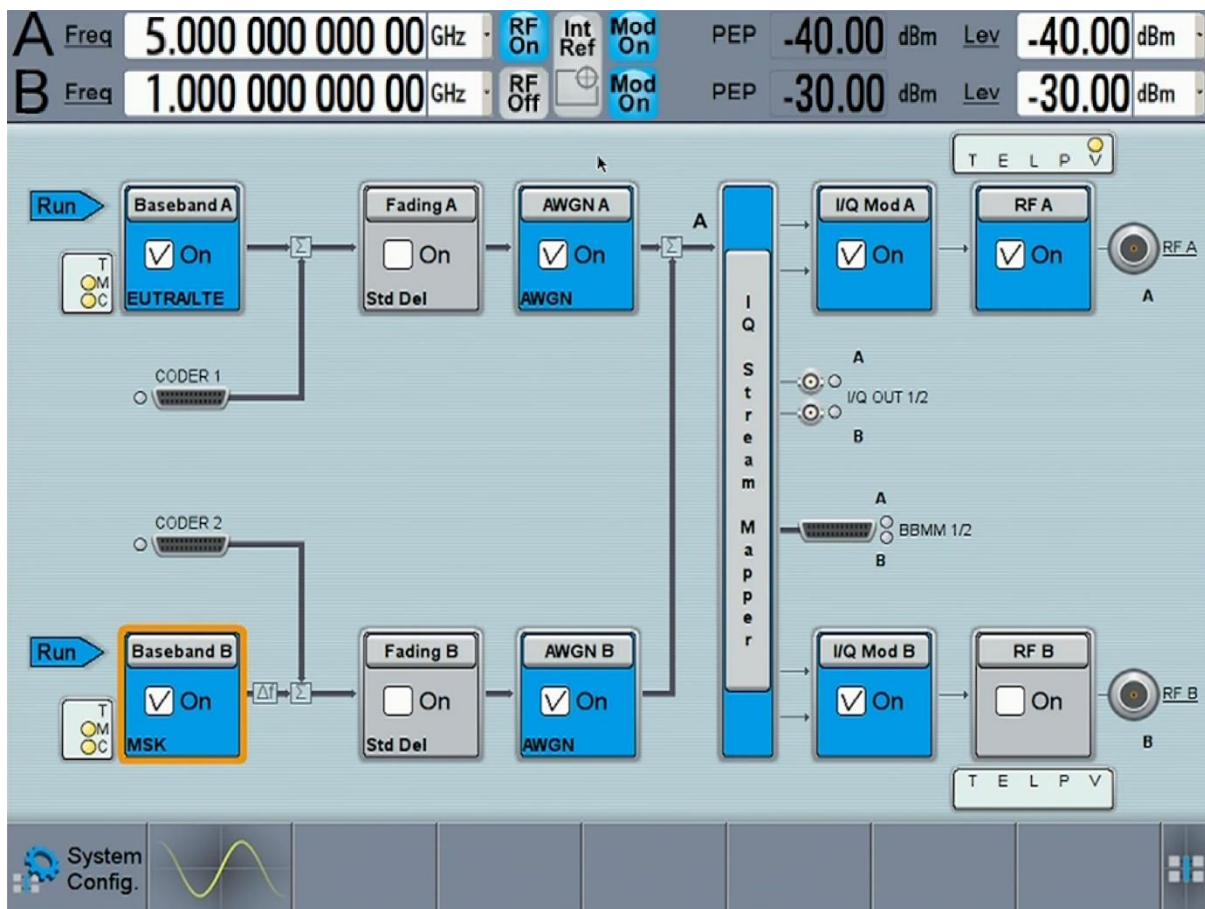


Fig. 27 - Set-up of the Signal Generator for generating 2 AWGN Signals

For both set-ups (in Fig. 26 and Fig. 27), fading is turned off since it is not required as the testing of wireless signals under Rayleigh fading channels is not within the scope of this experiment. Rayleigh fading channels introduce delays in the transmissions due to the movement of nodes and multipath reception, but in this experiment, all 3 nodes (transmitter,

receiver, jammer) are stationary. Multipath reception can occur when there are obstacles between the transmitter and receiver nodes. [33] However, in this experiment, all the nodes are in line of sight, which eliminates this problem.

The power of the jamming signal will be set in the Lev field, which is set to -40.00 dBm in Fig. 27. The centre frequency of the signal will be set in A Freq field, which is set to 5.000 000 000 GHz in Fig. 27.

The most important aspect in this block programming of the signal generator is the AWGN box. In this box, the AWGN generator block is set to operate in the “Noise Only” mode rather than the “Additive Noise” or “CW Interferer” mode. The “System Bandwidth” would be adjusted according to the frequencies that will be jammed. This “System Bandwidth” is half of the actual bandwidth, since the “System Bandwidth” extends in both left and right directions from the centre frequency defined by the A Freq field.

4.3 Actual Experiment Procedure

4.3.1 Experimental Settings used for the Jammer

The jammer is first set to different power levels of the AWGN signals being sent.

While the power levels of the AWGN signals being sent vary, there are also 4 different types of jamming signal being sent. The first type is BNJ. For this type, a centre frequency of 5.775 GHz is used, with a bandwidth of 80 MHz. The first set-up of the signal generator (which is the jammer) shown in Fig. 26 is used, the “A Freq” field is set to 5.775 000 000 00 GHz and the “System Bandwidth” is set to 40 MHz.

The second type of jamming being used is PBJ. As shown in Fig. 23 the primary channel used by the receiver node is channel 157. PBJ will thus jam only channel 157 while leaving

the other channels 151, 153 and 161 untouched. A centre frequency of 5.785 GHz is used, with a bandwidth of 20 MHz. The first set-up of the signal generator (which is the jammer) shown in Fig. 26 is used, the “A Freq” field is set to 5.785 000 000 00 GHz and the “System Bandwidth” is set to 10 MHz.

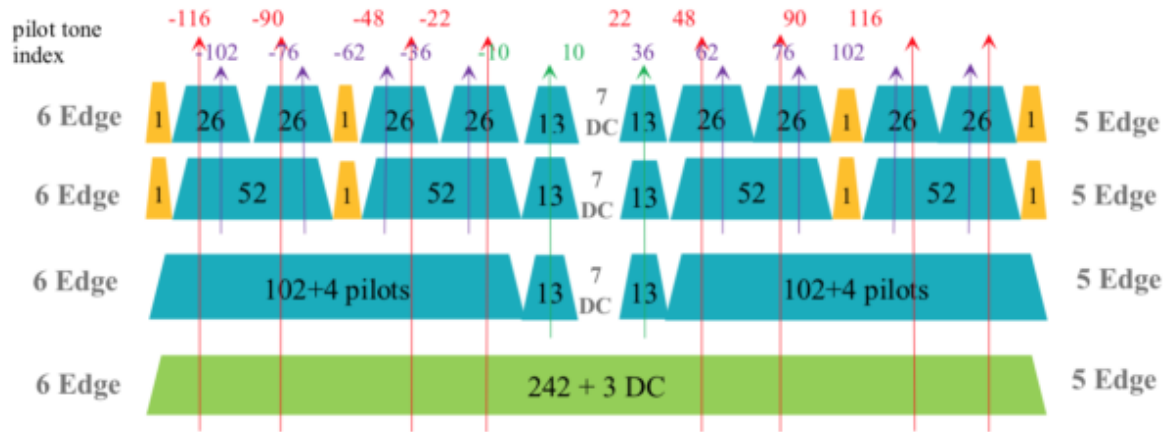


Fig. 28 - Illustration of the sub-carriers in a 20 MHz channel from the Rohde & Schwarz IEEE 802.11ax Technology Introduction white paper [34]

The third type of jamming being used is MTJ. 2 AWGN signals will be used, thus the set-up of the signal generator in Fig. 27 will be used. As detailed in Fig. 28, the first 6 sub-carriers in a 20 MHz channel are edge sub-carriers, or null sub-carriers. There is thus no point to jam these sub-carriers. The sub-carrier after the 6 sub-carriers may also be a null sub-carrier, depending on the resource allocation, which may not be a good idea to jam as well. Thus, the next 2 sub-carriers after the first 7 subcarriers will be jammed. These 2 subcarriers to jam shall be in channel 157, which is the main channel used by the receiver. This works out to jamming the sub-carrier that has a centre frequency of 5775.5859375 MHz ($5775 \text{ MHz} + 7.5 \times 78.125 \text{ kHz}$). The “A Freq” field is thus set to 5.775 585 937 50 GHz and the “System Bandwidth” is set to 78.125 kHz. The second AWGN signal is set to have a frequency offset of 78.125kHz as this value is the spacing between 2 OFDMA sub-carriers, which is the value of the frequency offset for Baseband B.

The final type of jamming being used is asynchronous off-tone jamming. 2 AWGN signals will be used, thus the set-up of the signal generator in Fig. 27 will be used. To achieve maximum BER, as proved by the authors of [19], 50% subcarrier jamming (Or -6 dB subcarrier jamming tone signal to subcarrier signal ratio) will be used. This means putting the centre frequency of the jamming tones exactly in the middle of the centre frequencies of the 2 adjacent sub-carriers. This works out to placing the centre frequency of one of the jamming tones to be at 5775.625 MHz ($5775 \text{ MHz} + 8 \times 78.125 \text{ kHz}$). The “A Freq” field is thus set to 5.775 625 000 00 GHz and the “System Bandwidth” is set to 78.125 kHz. The second AWGN signal is set to have a frequency offset of 78.125kHz as this value is the spacing between 2 OFDMA sub-carriers, which is the value of the frequency offset for Baseband B.

The waveform analysis of jamming signals being produced by the signal generator looks something like in Fig. 29 (using PBJ as an example), with the other 3 types of jamming signals having similar resemblance to Fig. 29, other than either having a wider width and possibly having two peaks instead.

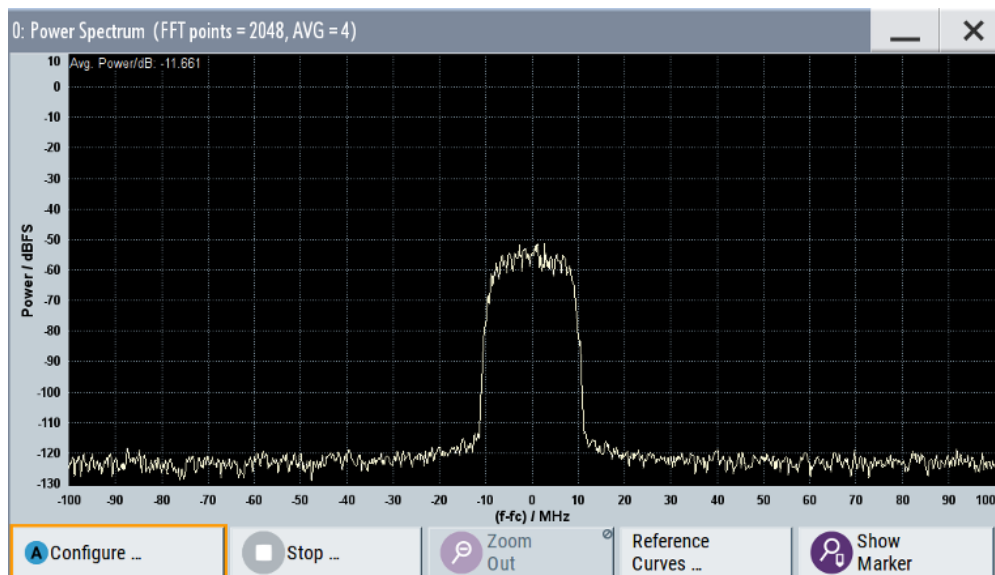


Fig. 29 - Frequency Spectrum of Partial Band Jamming (PBJ) Signal

4.3.2 Deriving the independent variable

Using the arbitrary chosen values for the power of the jamming signal, set in the Lev field in terms of dBm, the SJR can be calculated. To calculate the SJR, the equation for free space propagation model in line of sight can be used to derive a formula for the SJR:

Equation 2 - Propagation Model in Line-of-Sight

$$P_R = P_T G_T G_R \left(\frac{c}{4\pi R f} \right)^\alpha$$

Where P_R is the power received by the receiver in watts (W), P_T is the power transmitted in watts (W), G_T is the transmitter antenna gain in dB, G_R is the receiver antenna gain in dB, c is the speed of light, R is the distance between the transmitter and receiver in metres (m), f is the frequency of the signal being transmitted in Hz and α is the path loss component.

The SJR is expressed as the ratio of the power received of the useful wireless signal to that of the jammer signal. The subscript T will be replaced by subscript S to indicate the useful wireless signal and subscript J to indicate the jammer signal. The distance R will be replaced by d_s to indicate the distance between the transmitter node and the receiver node and d_j to indicate the distance between the jammer and the receiver node. The SJR can be now expressed as:

$$SJR = \frac{P_S G_S G_R \left(\frac{c}{4\pi d_s f} \right)^\alpha}{P_J G_J G_R \left(\frac{c}{4\pi d_j f} \right)^\alpha}$$

By cancelling some of the terms, this can be simplified to:

$$SJR = \frac{P_S G_S d_j^\alpha}{P_J G_J d_s^\alpha}$$

As the ratio is expressed in terms of power-to-power, a conversion to decibel is needed. The formula to derive decibels from the power ratio is as follows:

Equation 3 - Conversion from Power Ratio to Decibels

$$\text{Decibels (dB)} = 10 \log_{10} \frac{P_1}{P_2}$$

The formula to calculate SJR in terms of dB can be expressed as:

Equation 4 - Calculating SJR in dB

$$SJR_{dB} = P_S + G_S - P_J - G_J + 10\alpha \log_{10} d_J - 10\alpha \log_{10} d_S$$

Where P_S is the transmitting useful wireless signal power in dBm, P_J is the jamming signal power in dBm, G_S is the transmitter antenna gain in dB, G_J is the antenna gain of the jammer in dB, d_J is the distance between the receiver and the jammer and d_S is the distance in metres (m) between the transmitter and the receiver in metres (m).

To simplify things further, the same type of antennas is used for both the transmitter and the jammer. Since antenna gain depends on the antenna used, $G_S = G_J$.

While the path loss component is usually 2 (i.e. $\alpha = 2$) as is the case for free space propagation, this experiment was carried out in an indoor setting, which makes α difficult to determine due to the many physical obstacles in an indoor setting. For simplicity, the distance between the transmitter and the receiver and the distance between the jammer and receiver are both kept to 1m, which means $d_S = d_J = 1$. This cancels out the logarithm terms which also cancels the α in these terms.

P_S , which is the transmitting power of the useful wireless signal, is known to be 23 dBm as discussed in 4.2.4 Operating Parameters of Transmitter and Receiver, thus the final equation is:

Equation 5 - Simplified Calculation of SJR after setting the parameters

$$SJR_{dB} = 23 - P_J$$

4.3.3 Conducting the transmission from transmitter to receiver

The program used to receive the file is compiled on the receiver. There is one compilation warning which is not important to resolve. The compiled program is then executed without any extra parameters and left to run. The program will then be left stuck at the line “Start receiving” as shown in Fig. 30, until the transmitted file is sent.



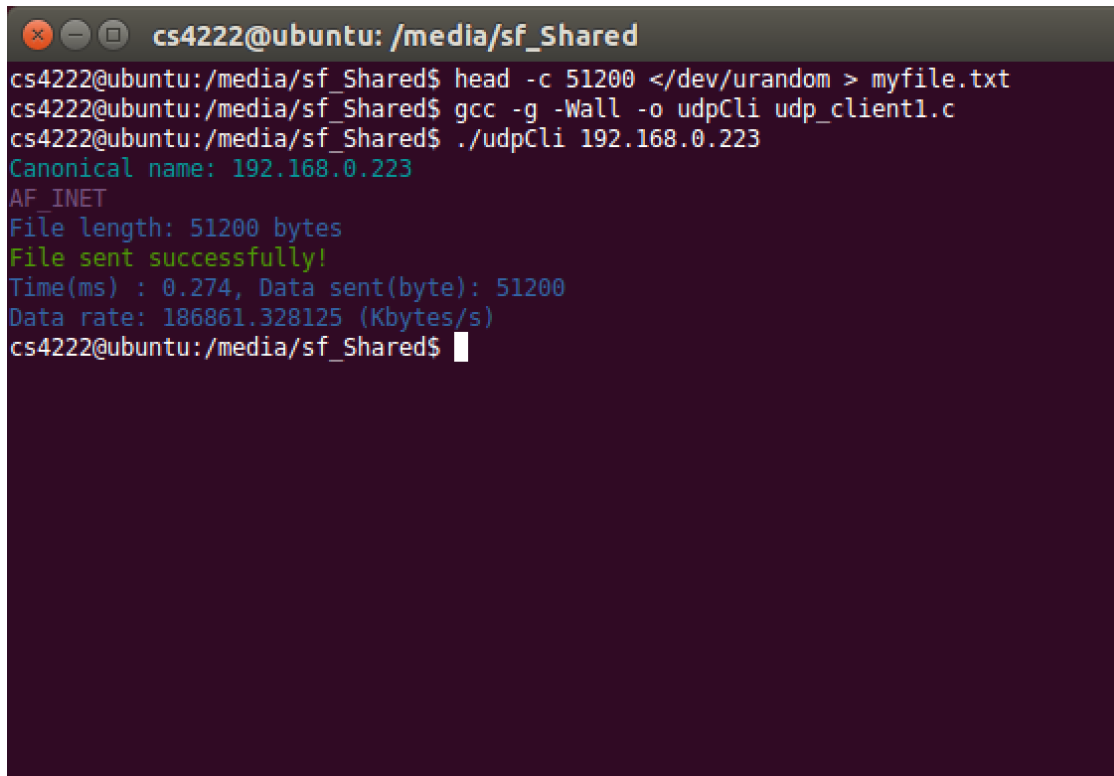
```
admin@admin: ~/Downloads
admin@admin:~/Downloads$ hostname -I
192.168.0.223
admin@admin:~/Downloads$ gcc -g -Wall -o udpSvr udp_ser.c
udp_ser.c: In function 'str_ser1':
udp_ser.c:56:81: warning: pointer targets in passing argument 6 of 'recvfrom' differ in signedness [-Wpointer-sign]
    size = recvfrom(sockfd, &recvs, DATALEN, 0, (struct sockaddr *)&addr, &addrlen);
                                                                    ^
In file included from headsock.h:2:0,
                  from udp_ser.c:15:
/usr/include/x86_64-linux-gnu/sys/socket.h:174:16: note: expected 'socklen_t * restrict {aka unsigned int * restrict}' but argument is of type 'int *'
    extern ssize_t recvfrom (int __fd, void *__restrict __buf, size_t __n,
                          ^
admin@admin:~/Downloads$ ./udpSvr
Start receiving
█
```

Fig. 30 - Compiling and Running the program in preparation to receive the file from transmitter

Next, the program used to send the file is compiled on the transmitter. The Linux terminal accepts only 2 arguments. Otherwise, the program exits. The following part of the code extracted from the full code in APPENDIX A – C Source Code used for UDP sending and receiving handles this. The first argument `argv[0]` is simply the execution of the program which was named `udpCli` as shown in Fig. 31.

```
if (argc != 2) {
    printf("Insufficient Parameters / Parameters mismatch");
    exit(0);
}
// Get Host Info
if ((sh = gethostbyname(argv[1])) == NULL) {
    printf("Error when gethostbyname");
    exit(0);
}
```


The IP Address of the receiver which was found to be 192.168.0.223 (in 4.2.3 Wireless Connection Environment Set-Up) is then used as the input field (which is argv[1]) when running the program as shown in Fig. 31.

A terminal window titled 'cs4222@ubuntu: /media/sf_Shared' showing the following commands and output:

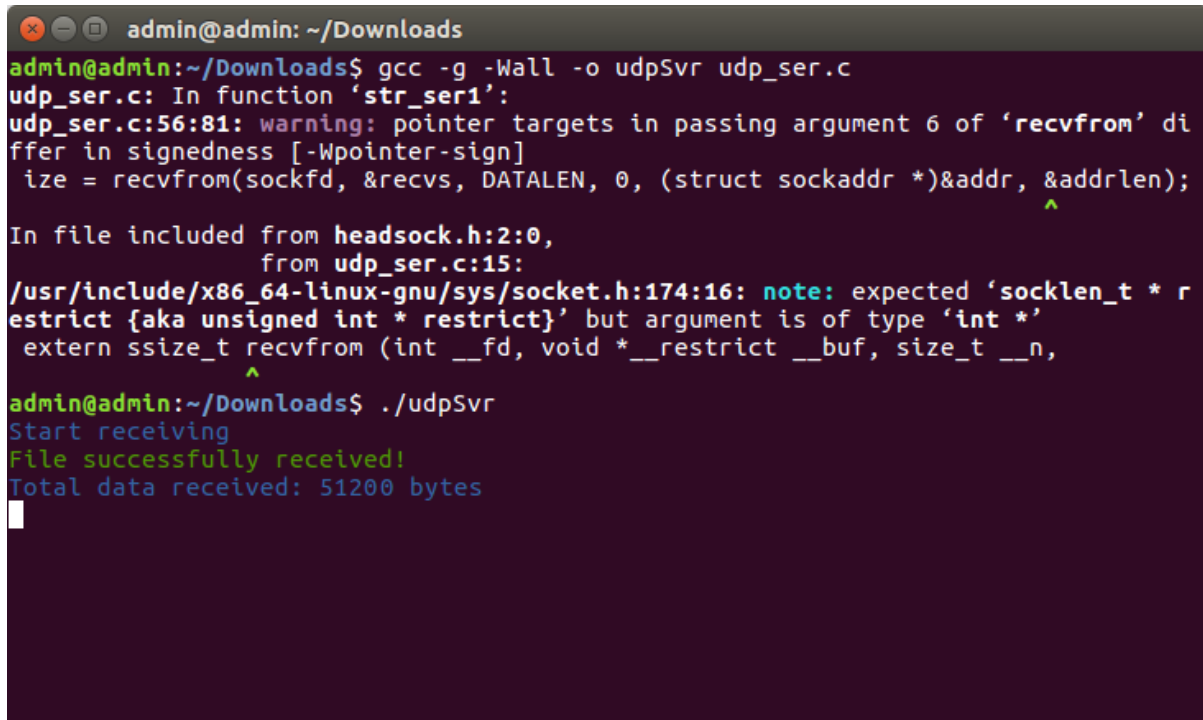
```
cs4222@ubuntu:/media/sf_Shared$ head -c 51200 </dev/urandom > myfile.txt
cs4222@ubuntu:/media/sf_Shared$ gcc -g -Wall -o udpCli udp_client1.c
cs4222@ubuntu:/media/sf_Shared$ ./udpCli 192.168.0.223
Canonical name: 192.168.0.223
AF_INET
File length: 51200 bytes
File sent successfully!
Time(ms) : 0.274, Data sent(byte): 51200
Data rate: 186861.328125 (Kbytes/s)
cs4222@ubuntu:/media/sf_Shared$
```

Fig. 31 - Compiling and Running the program to send the file to receiver

Before the program on the transmitter is being run, the 50 KB file must be named “myfile.txt” since this file is what the transmitter program looks for, otherwise the program will exit as shown in the code excerpt below:

```
// Open file
if ((fp = fopen("myfile.txt", "r+t")) == NULL) {
    printf("File does not exist\n");
    exit(0);
}
```

Once the program used to send the file is executed, since the receiver was already ready to receive the file when the receiver program was left running, the file is sent over the air. After the entirety of the 50 KB is successfully sent over the air, a message returns on both the sender and transmitter as shown in both Fig. 31 and Fig. 32. The file received on the receiver is named “myUDPreceive.txt”.



```
admin@admin: ~/Downloads
admin@admin:~/Downloads$ gcc -g -Wall -o udpSvr udp_ser.c
udp_ser.c: In function 'str_ser1':
udp_ser.c:56:81: warning: pointer targets in passing argument 6 of 'recvfrom' differ in signedness [-Wpointer-sign]
    size = recvfrom(sockfd, &recvs, DATALEN, 0, (struct sockaddr *)&addr, &addrlen);
                                                                    ^
In file included from headsock.h:2:0,
                  from udp_ser.c:15:
/usr/include/x86_64-linux-gnu/sys/socket.h:174:16: note: expected 'socklen_t * restrict {aka unsigned int * restrict}' but argument is of type 'int *'
    extern ssize_t recvfrom (int __fd, void *__restrict __buf, size_t __n,
                          ^
admin@admin:~/Downloads$ ./udpSvr
Start receiving
File successfully received!
Total data received: 51200 bytes
```

Fig. 32 - File successfully received from transmitter

4.3.3 Deriving the dependent variable

Once the transmitted file is received, it is compared against the master copy of the transmitted file which was placed in the receiver and verified to be the exact same copy of the transmitted file by means of checksum calculations.

Fig. 33 shows an example case of the corruption caused by the jammer as it is transmitted over the air. Since all data in real world applications are interpreted by bytes (since data is interpreted by characters, which is one byte long), rather than down to each individual bit, it would suffice to compare the different bytes between the 2 files by bytes and quantify BER using the number of bytes that differ from the original file. The highlighted portions on the right side of Fig. 33 show the bytes that were corrupted during the over-the-air transmission by the jammer, which are different from the highlighted bytes on the left side.

00000	2c a9 a7 24 7a da 95 fc 54 65 55 6b 42 c3 bb 18 d0 75 6c b4 c7	00000	2c a9 a7 24 7a da 95 fc 54 65 55 6b 42 c3 bb 18 d0 75 6c b4 c7
00001	dc 56 cc 86 11 db 09 21 28 b0 00 92 ed 89 0a 88 31 ff 59 73	00001	dc 56 cc 86 11 db 09 21 28 b0 00 92 ed 89 0a 88 31 ff 59 73
00002	45 d2 fe 30 30 49 b1 0c 62 cd 39 c1 95 4f 53 da 9e 69 ff ed 63	00002	45 d2 fe 30 30 49 b1 0c 62 cd 39 c1 95 4f 53 da 9e 69 ff ed 63
00003	32 1d a4 6a af 05 32 0d a7 44 11 5e 93 83 ce 56 e1 01 f6 8c	00003	32 1d a4 6a af 05 32 0d a7 44 11 5e 93 83 ce 56 e1 01 f6 8c
00004	b7 5a 76 2d fd 8e 36 30 8a c7 5b a8 10 13 95 59 6b 80 04 70 ff	00004	b7 5a 76 2d fd 8e 36 30 8a c7 5b a8 10 13 95 59 6b 80 04 70 ff
00005	ba 89 28 a3 42 08 cf 0b 48 8d fd 66 d2 e2 2d 2b 1a 36 fc dd	00005	ba 89 28 a3 42 08 cf 0b 48 8d fd 66 d2 e2 2d 2b 1a 36 fc dd
00006	1d 7c b5 76 42 83 ee 75 8f 60 fd 77 8d 04 01 4e 9b bb e5 66 24	00006	1d 7c b5 76 42 83 ee 75 8f 60 fd 77 8d 04 01 4e 9b bb e5 66 24
00007	42 3f 5e b1 3a a0 b2 2b 9a cf cd 82 22 9d 0c 72 2c ff 34 e1 96	00007	42 3f 5e b1 3a a0 b2 2b 9a cf cd 82 22 9d 0c 72 2c ff 34 e1 96
00008	b8 27 ff 7e 48 6d 11 df 8f 5f 51 95 b8 da 3d 57 c4 07 e2 3b 49	00008	b8 27 ff 7e 48 6d 11 df 8f 5f 51 95 b8 da 3d 57 c4 07 e2 3b 49
00009	f4 91 2f 1f 6b 67 c4 dc 6f bb b1 85 8e 89 18 5e 32 f1 33 a1 4c	00009	f4 91 2f 1f 6b 67 c4 dc 6f bb b1 85 8e 89 18 5e 32 f1 33 a1 4c
00010	0b 09 7c ed 69 c2 cf 1a 67 f7 51 a5 4c c3 96 d0 89 dc 06 4c 03	00010	0b 09 7c ed 69 c2 cf 1a 67 f7 51 a5 4c c3 96 d0 89 dc 06 4c 03
00011	e8 01 82 45 b4 fd 8b 06 2a c2 32 4a 9f 31 e2 bd d3 f1 ea a3 70	00011	e8 01 82 45 b4 fd 8b 06 2a c2 32 4a 9f 31 e2 bd d3 f1 ea a3 70
00012	20 39 9f 10 7d 51 24 f8 91 16 af 9a fd c9 b7 b3 07 01 a4 f4 c1	00012	20 39 9f 10 7d 51 24 f8 91 16 af 9a fd c9 b7 b3 07 01 a4 f4 c1
00013	07 c4 5f df 41 c5 81 8e a9 14 2d 54 49 e0 9f dc dc 3f 71 aa 22	00013	07 c4 5f df 41 c5 81 8e a9 14 2d 54 49 e0 9f dc dc 3f 71 aa 22
00014	d6 e1 32 0b 4f da de 1d 82 bd 62 58 41 18 75 3f 31 68 8b a3 6a	00014	d6 e1 32 0b 4f da de 1d 82 bd 62 58 41 18 75 3f 31 68 8b a3 6a
00015	92 67 06 1a f7 23 b7 23 aa 46 05 37 ff be b5 85 ee 39 72 4b	00015	92 67 06 1a f7 23 b7 23 aa 46 05 37 ff be b5 85 ee 39 72 4b
00016	20 a7 70 7a f1 a6 a1 dd 8f 91 8b 47 ad bc 39 85 1b 05 f8 32 65	00016	20 a7 70 7a f1 a6 a1 dd 8f 91 8b 47 ad bc 39 85 1b 05 f8 32 65
00017	3f 9c fa 52 e5 31 4c 53 55 3b ff 76 eb 72 bf ec 88 29 51 d0 d9	00017	3f 9c fa 52 e5 31 4c 53 55 3b ff 76 eb 72 bf ec 88 29 51 d0 d9
00018	93 19 ba 56 8a 43 85 7d ca 39 3f 40 e5 e8 0b 75 0c 75 54 8a	00018	93 19 ba 56 8a 43 85 7d ca 39 3f 40 e5 e8 0b 75 0c 75 54 8a
00019	3a 56 00 76 c1 49 79 48 93 8b a9 5f 8b a3 83 3f f3 5a ed c7 08	00019	3a 56 00 76 c1 49 79 48 93 8b a9 5f 8b a3 83 3f f3 5a ed c7 08
00020	8a 02 a0 35 05 89 b4 77 5b 1d db 7e 4f 59 e3 0e 2d c9 f9 0b 35	00020	8a 02 a0 35 05 89 b4 77 5b 1d db 7e 4f 59 e3 0e 2d c9 f9 0b 35
00021	37 ad 95 e1 5b 44 87 35 38 b1 87 8c 46 af d7 0c 06 bc c1 c5 d1	00021	37 ad 95 e1 5b 44 87 35 38 b1 87 8c 46 af d7 0c 06 bc c1 c5 d1
00022	7f 6d cd e0 3c 4a c0 31 c6 91 f4 bb 0d 53 54 0a 54 ab 44 7c 41	00022	7f 6d cd e0 3c 4a c0 31 c6 91 f4 bb 0d 53 54 0a 54 ab 44 7c 41
00023	23 6b 54 d0 9a a9 6f 02 32 c0 39 46 63 00 3f c8 00 1d fa c0 04	00023	23 6b 54 d0 9a a9 6f 02 32 c0 39 46 63 00 3f c8 00 1d fa c0 04
00024	70 23 93 d0 14 5b ff 02 a6 73 b7 98 8f 13 9a 78 ef a2 a0 50 2e	00024	70 23 93 d0 14 5b ff 02 a6 73 b7 98 8f 13 9a 78 ef a2 a0 50 2e
00025	94 84 ba 75 3c 53 14 41 58 8e 4a 65 3d b1 4b c1 21 85 4d 70 ff	00025	94 84 ba 75 3c 53 14 41 58 8e 4a 65 3d b1 4b c1 21 85 4d 70 ff
00026	8f bc fd ad c3 ec 3d 10 71 7a 59 76 d5 ed 9c d8 5b d8 08 ea 0f	00026	8f bc fd ad c3 ec 3d 10 71 7a 59 76 d5 ed 9c d8 5b d8 08 ea 0f
00027	51 82 31 33 52 62 b6 b7 cd 1d 82 e0 2f 08 87 fd c8 68 68 6d 7f 11	00027	51 82 31 33 52 62 b6 b7 cd 1d 82 e0 2f 08 87 fd c8 68 68 6d 7f 11
00028	43 13 f0 ee 87 d3 af e8 c6 ac 45 85 3f 39 7a 8d 7a 81 89 86 98	00028	43 13 f0 ee 87 d3 af e8 c6 ac 45 85 3f 39 7a 8d 7a 81 89 86 98
00029	0c 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01	00029	0c 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00030	44 88 c4 c2 76 0e b8 56 2a 91 c6 14 c4 33 ga e8 80 a2 02 26	00030	44 88 c4 c2 76 0e b8 56 2a 91 c6 14 c4 33 ga e8 80 a2 02 26
00031	18 51 e2 01 8f c4 c8 f1 1e 0a f6 94 a3 ee b4 de b0 f8 1c 52 f4	00031	18 51 e2 01 8f c4 c8 f1 1e 0a f6 94 a3 ee b4 de b0 f8 1c 52 f4
00032	4a 6b 69 12 07 66 09 8e 52 35 64 04 43 2c 37 e2 f0 e7 49 f0 65	00032	4a 6b 69 12 07 66 09 8e 52 35 64 04 43 2c 37 e2 f0 e7 49 f0 65
00033	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00033	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00034	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00034	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00035	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00035	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00036	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00036	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00037	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00037	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00038	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00038	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00039	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00039	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00040	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00040	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00041	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00041	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00042	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00042	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00043	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00043	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00044	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00044	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00045	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00045	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00046	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00046	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00047	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00047	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00048	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00048	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00049	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00049	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00050	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00050	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00051	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00051	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00052	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00052	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00053	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00053	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00054	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00054	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00055	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00055	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00056	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00056	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00057	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00057	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00058	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00058	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00059	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00059	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00060	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00060	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00061	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00061	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00062	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00062	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00063	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00063	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00064	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00064	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00065	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00065	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00066	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00066	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00067	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00067	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00068	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00068	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00069	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00069	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00070	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00070	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00071	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00071	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00072	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00072	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00073	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00073	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00074	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00074	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00075	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00075	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00076	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00076	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00077	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00077	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00078	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00078	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00079	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00079	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00080	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00080	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00081	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00081	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00082	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00082	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00083	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00083	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00084	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00084	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00085	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00085	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62
00086	cc 20 9a d8 1e e6 7f 8b 37 88 85 af 31 c0 e5 79 bf 88 5d 0f 62	00086	cc 20 9a d8 1e e6 7f 8b 37

The number of corrupted bytes can be determined by calculating the Hamming Distance between the very long string of bytes in both files, which suffices since both files are of the same byte length. A simple program written in C that calculates the Hamming Distance is written and included in APPENDIX B – C Source Code used for calculating Hamming Distance. This program returns the hamming distance, which is equivalent to the number of bytes which are different. The program takes in no parameters, and checks whether the two files, the master copy (myfile.txt) and the received file (myUDPreceive.txt) exists in the same directory of the compiled program. Otherwise, it exits the program immediately. An excerpt of the code that handles this is shown below:

```
// Open master file
if ((fp1 = fopen("myfile.txt", "r+t")) == NULL) {
    printf("Master File does not exist\n");
    exit(0);
}
// Open received file
if ((fp1 = fopen("myUDPreceive.txt", "r+t")) == NULL) {
    printf("Received File does not exist\n");
    exit(0);
}
```

The Hamming Distances calculated over all 10 trials of a single experimental setting (which consists of both the jamming signal type and the jamming power) is put back into Equation 1. Equation 1 can now be re-written as:

Equation 6 - Calculation of BER in terms of Hamming Distances

$$BER = \frac{\sum_{x=1}^{10} (\text{Hamming Distance for trial } x)}{512000}$$

CHAPTER 5 – Results

BER will be used to quantify the effects of each of the 4 jamming attacks. Since BER measures the percentage of bits that were corrupted during the over the air transmission, it is an accurate measure of the quality of the wireless signal under attack from adversaries who wishes to jeopardize communications. For low BER, the data in the wireless signal can still be reconstructed since the corrupted data can be discarded and the empty gaps left behind can be guessed. However, given a high enough BER, the gap becomes too big to make a reasonable guess and the rest of the data becomes useless.

SJR is also used as a variable for each of the jamming attacks to observe the effects of increasing and decreasing jammer powers on the quality of the wireless signal. A high jamming power is most certainly likely to cause significant enough BER to cause disruption, but would incur cost to the adversaries themselves, which may make such attacks unprofitable. Jamming can also be stealthy enough to avoid detection and corrective actions.

Equation 4 and Equation 6 provide the necessary parameters to calculate the independent variable SJR and the dependent variable BER. Thus, the columns in the table that do not address SJR and BER are used for intermediate calculations. Some of the columns are also shown to allow readers to understand that the experiment is controlled such that other variables identified when deriving the SJR and BER are not changed.

The results of the 4 types of jamming are as shown in the following 4 tables:

Table 3 - Results of BER against SJR under Barrage Jamming (BNJ)

Power of Jamming Signal used (dBm)	Power of Wireless Signal (dBm)	Distance between Jammer and Receiver (m)	Distance between Transmitter and Receiver (m)	SJR (dB)	Number of corrupted bytes over 10 trials	Number of bytes sent over 10 trials	BER (to 5 d.p.)
29	23	1.0	1.0	-6	215506	512000	0.42091
27	23	1.0	1.0	-4	204053	512000	0.39854
25	23	1.0	1.0	-2	178606	512000	0.34884
23	23	1.0	1.0	0	133611	512000	0.26096
21	23	1.0	1.0	2	89765	512000	0.17532
19	23	1.0	1.0	4	64763	512000	0.12649
17	23	1.0	1.0	6	46717	512000	0.09124
15	23	1.0	1.0	8	38298	512000	0.07480
13	23	1.0	1.0	10	33451	512000	0.06533
11	23	1.0	1.0	12	30676	512000	0.05991

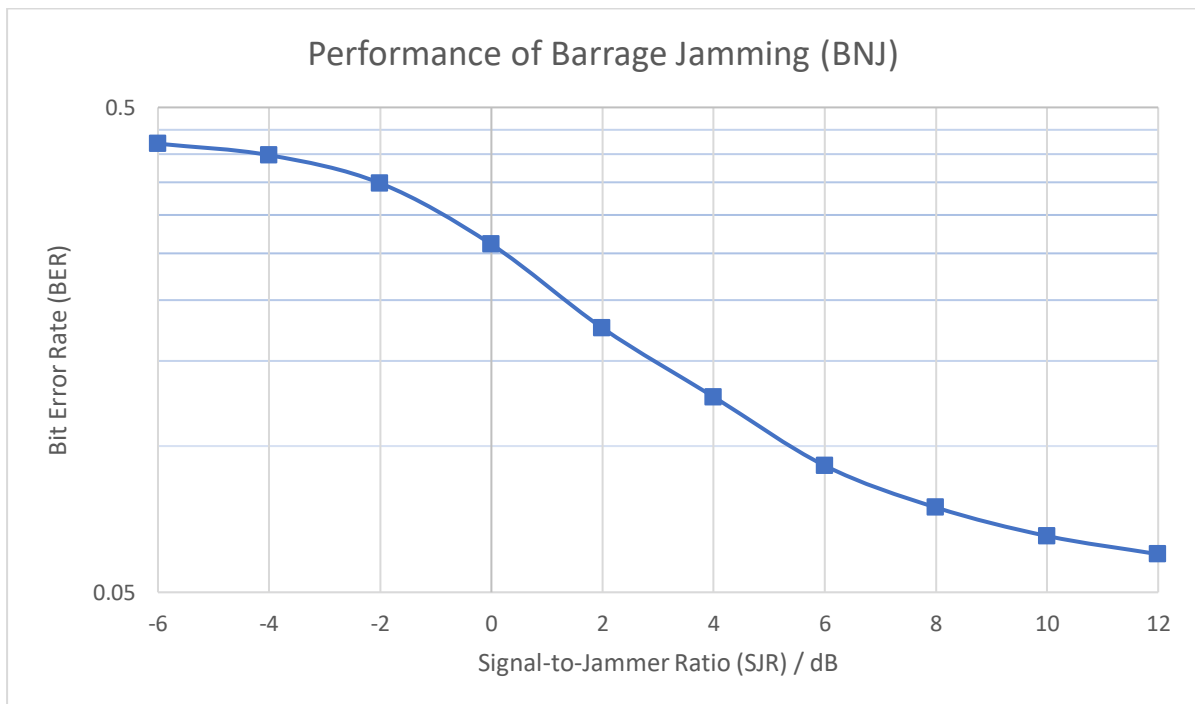


Fig. 34 - Performance of BNJ quantified in BER

Table 4 - Results of BER against SJR under Partial Band Jamming (PBJ)

Power of Jamming Signal used (dBm)	Power of Wireless Signal (dBm)	Distance between Jammer and Receiver (m)	Distance between Transmitter and Receiver (m)	SJR (dB)	Number of corrupted bytes over 10 trials	Number of bytes sent over 10 trials	BER (to 5 d.p.)
29	23	1.0	1.0	-6	193448	512000	0.37783
27	23	1.0	1.0	-4	163937	512000	0.32019
25	23	1.0	1.0	-2	107592	512000	0.21014
23	23	1.0	1.0	0	71937	512000	0.14050
21	23	1.0	1.0	2	56461	512000	0.11028
19	23	1.0	1.0	4	45638	512000	0.08914
17	23	1.0	1.0	6	36614	512000	0.07151
15	23	1.0	1.0	8	32748	512000	0.06396
13	23	1.0	1.0	10	29948	512000	0.05849
11	23	1.0	1.0	12	28785	512000	0.05622

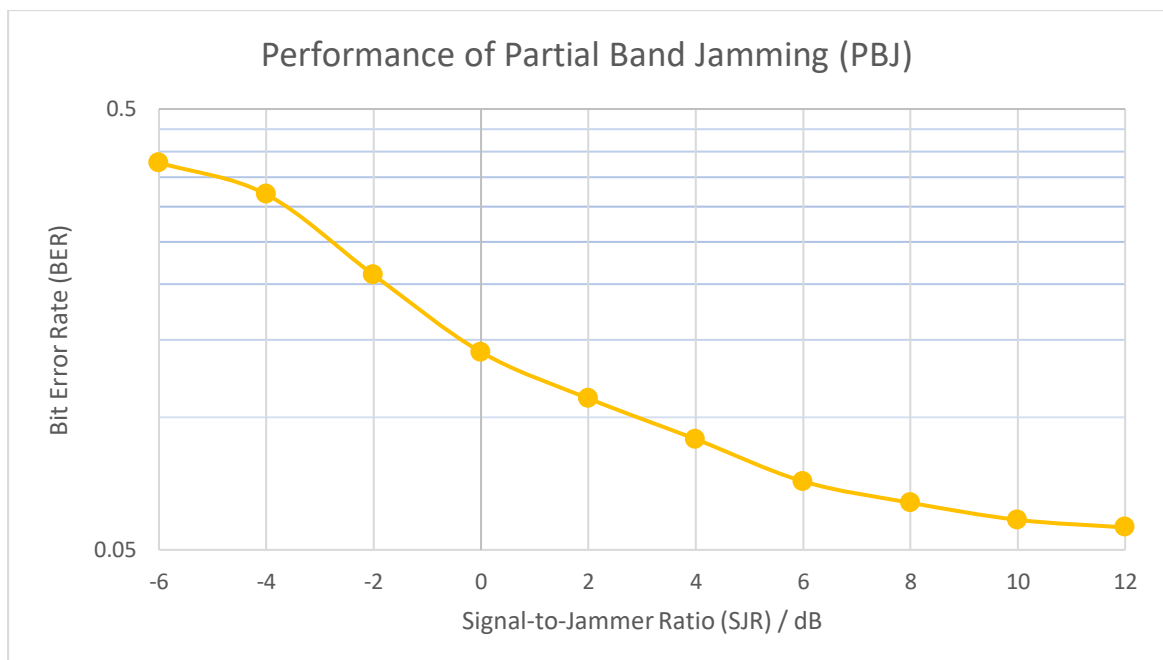


Fig. 35 - Performance of PBJ quantified in BER

Table 5 - Results of BER against SJR under Multi-Tone Jamming (MTJ)

Power of Jamming Signal used (dBm)	Power of Wireless Signal (dBm)	Distance between Jammer and Receiver (m)	Distance between Transmitter and Receiver (m)	SJR (dB)	Number of corrupted bytes over 10 trials	Number of bytes sent over 10 trials	BER (to 5 d.p.)
29	23	1.0	1.0	-6	89236	512000	0.17429
27	23	1.0	1.0	-4	69960	512000	0.13664
25	23	1.0	1.0	-2	48067	512000	0.09388
23	23	1.0	1.0	0	40279	512000	0.07867
21	23	1.0	1.0	2	35338	512000	0.06902
19	23	1.0	1.0	4	32046	512000	0.06259
17	23	1.0	1.0	6	29947	512000	0.05849
15	23	1.0	1.0	8	29066	512000	0.05677
13	23	1.0	1.0	10	28165	512000	0.05501
11	23	1.0	1.0	12	27581	512000	0.05387

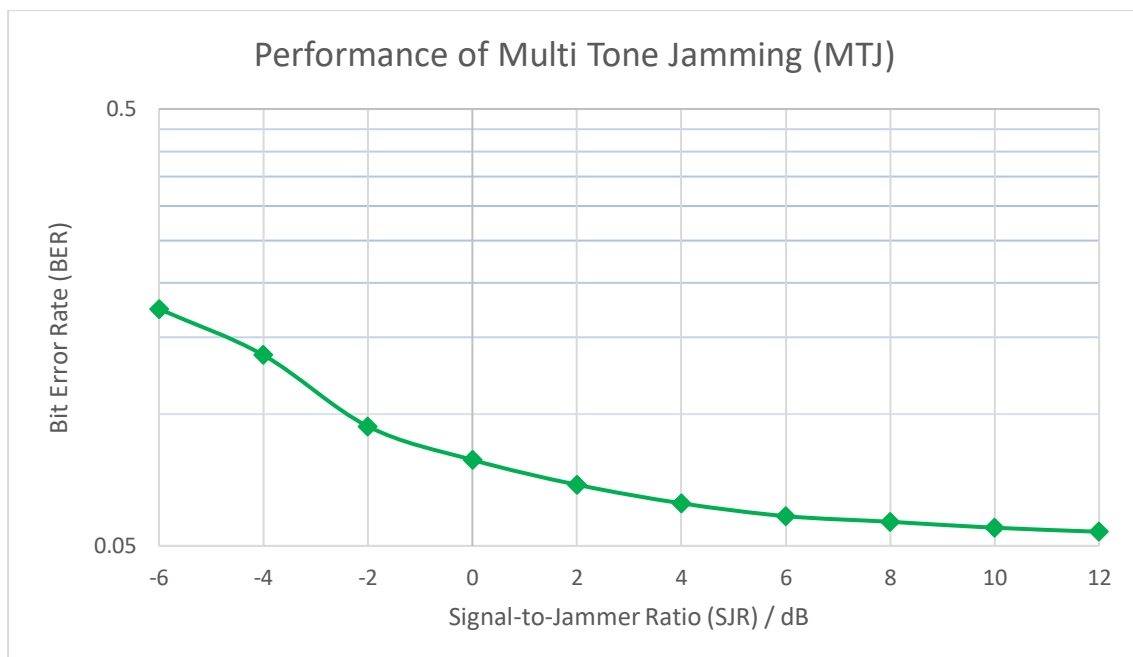


Fig. 36 - Performance of MTJ quantified in BER

Table 6 - Results of BER against SJR under Asynchronous Off-Tone Jamming

Power of Jamming Signal used (dBm)	Power of Wireless Signal (dBm)	Distance between Jammer and Receiver (m)	Distance between Transmitter and Receiver (m)	SJR (dB)	Number of corrupted bytes over 10 trials	Number of bytes sent over 10 trials	BER (to 5 d.p.)
29	23	1.0	1.0	-6	114678	512000	0.22398
27	23	1.0	1.0	-4	95939	512000	0.18738
25	23	1.0	1.0	-2	61932	512000	0.12096
23	23	1.0	1.0	0	46177	512000	0.09019
21	23	1.0	1.0	2	41011	512000	0.08010
19	23	1.0	1.0	4	35732	512000	0.06979
17	23	1.0	1.0	6	32317	512000	0.06312
15	23	1.0	1.0	8	30385	512000	0.05935
13	23	1.0	1.0	10	28790	512000	0.05623
11	23	1.0	1.0	12	27966	512000	0.05462

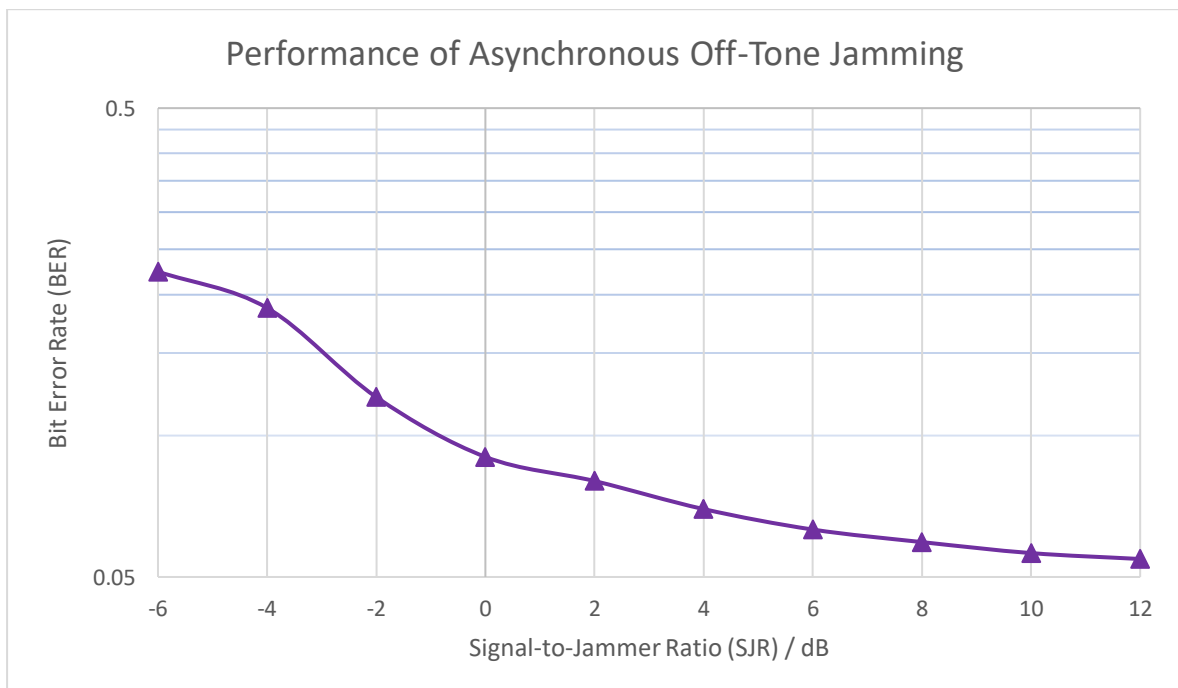


Fig. 37 - Performance of Asynchronous Off-Tone Jamming quantified in BER

Fig. 38 compares the performance of each of the jamming attacks against each other.

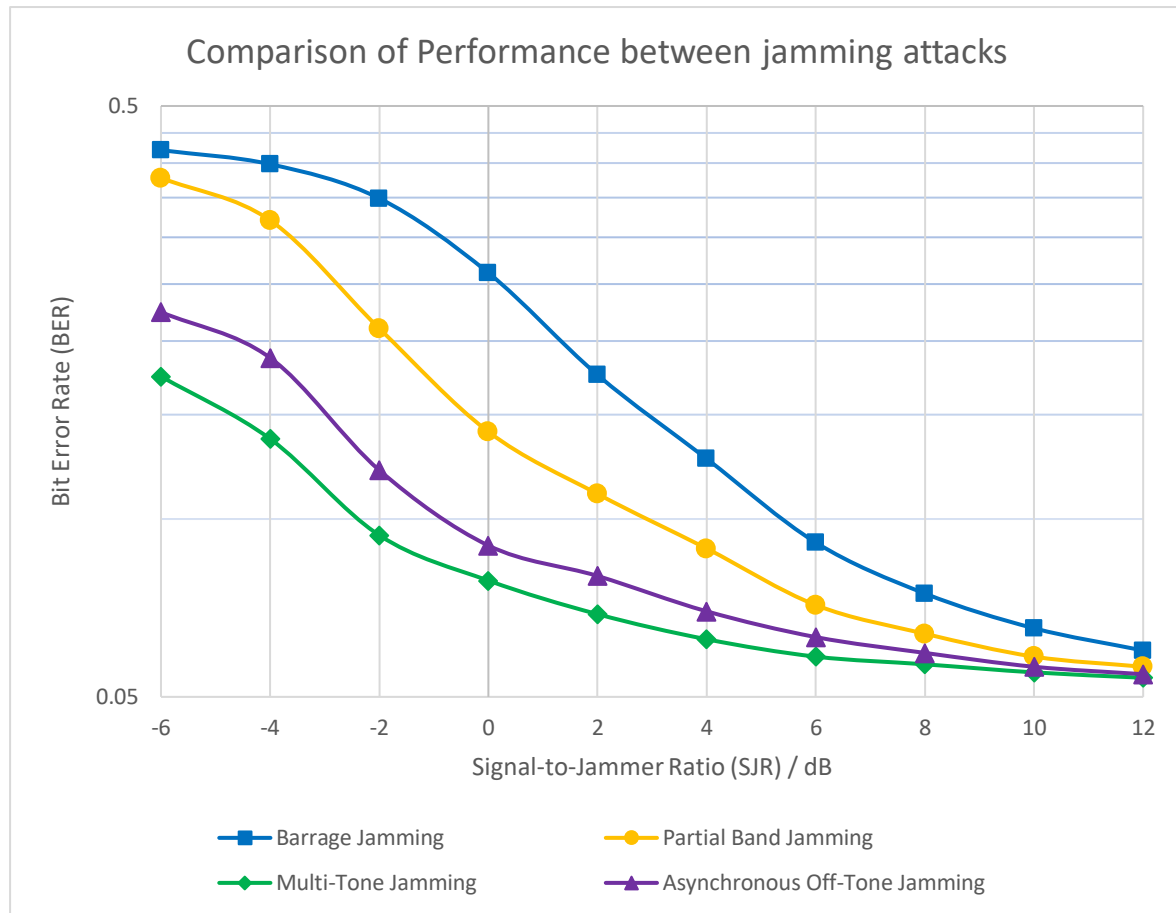


Fig. 38 - Comparison of different jamming attacks

All the jamming attacks follow a similar trend when the BER starts to drop rather quickly as the SJR increases from -4 dB to 6 dB. Thereafter, the decrease in BER becomes significantly slower. This likely means that beyond 6 dB SJR, the jamming signal becomes too weak to make much of an impact on bit corruption, and as the SJR increases beyond 12 dB, it is likely that the environment noise causes most of the bit corruptions rather than the jamming signals themselves.

However, from -6 dB to -4 dB, the decrease in BER is not as much as that of -4 dB to -2 dB or -2 dB to 0 dB.

There is an initial huge gap in the BER between PBJ and asynchronous off-tone jamming.

This is expected since only a small part of the OFDMA signal is affected as only 2 AWGN tones with the width of an OFDMA subcarrier signal (78.125 kHz) are generated, as opposed to a large AWGN signal which covers a wide bandwidth of an entire channel (20 MHz).

However, this is an impressive result from asynchronous off-tone jamming since the total power of this signal is significantly less than PBJ, but is still able to cause about 60% of the BER rate for almost all SJRs tested as PBJ despite consuming significantly less power than PBJ, since it only jams 2 out of the 256 sub-carriers which PBJ jams. This proves that the ICI caused by the asynchronous off-tone attacks are potent enough to cause a noticeable difference from normal MTJ where the jamming tones are aligned with the sub-carrier tones.

CHAPTER 6 – Conclusion

6.1 Summary

This report first started off with the scope of the research and how it is relevant to modern city life. It zooms into the specific area of research and gives readers an understanding of how the technology works. The introduction of an upgrade to the technology is then explained. The problem is introduced with the various techniques this technology in the discussion can be affected, with quantifications given to how much damage these techniques can cause to disrupt our modern lives. An experiment is designed and then conducted to see how some of these past techniques to cause damage holds against the newly upgraded technology in a commercial environment.

More specifically, barrage jamming, partial band jamming, multi-tone jamming, and asynchronous off-tone jamming were investigated in various situations against the newest revision of Wi-Fi, the 802.11ax standard, also commonly known as Wi-Fi 6. The impact of these attacks was compared to each other. Details of quantitative analysis were presented, which were backed up by experimentation and plotting the results on a graph to make a conclusion on the impact of these attacks.

This paper revealed that partial band jamming and multi-tone jamming are more power-efficient than barrage jamming for the reduced bandwidth they cover. However, they deal less damage to the quality of the wireless signal. The experimentation results also reveal that asynchronous off-tone jamming, despite consuming the same amount of power as multi-tone jamming in this experiment, is able to cause noticeably more damage than multi-tone jamming.

6.2 Future work that can be considered

Many of the other attacks mentioned in this paper were not tested against this new Wi-Fi 6 technology. Attacks such as pilot jamming, pilot nulling, false preamble timing, preamble phase warping, differential scrambling attack, hybrid acknowledgement request attack, resource allocation attack and random access channel attack are not in the scope of the experiment detailed in this paper and can be future areas to explore.

Past researchers [15], [19], [20], [21], [22], [24], [25], [26] have done well to document the effects of many of these attacks on past Wi-Fi standards. However, the new Wi-Fi 6 technology employs a new technique known as OFDMA which replaces the old OFDM which was the scope of these past researchers. Furthermore, some attacks detailed by the authors of [15] have yet to quantify the effects of the hybrid acknowledgment request attack and the resource allocation attack on OFDMA systems such as Wi-Fi 6.

REFERENCES

- [1] Wireless Broadband Alliance, “Facts and stats,” 13 April 2017. [Online]. Available: <https://worldwifeday.com/about-us/facts/>. [Accessed 17 March 2020].
- [2] R. Crist, “The iPhone 11 supports Wi-Fi 6. Here's why that matters.,” 16 September 2019. [Online]. Available: <https://www.cnet.com/how-to/the-iphone-11-supports-wi-fi-6-here-is-what-that-means-for-you/>. [Accessed 17 March 2020].
- [3] E. Khorov, A. Kiryanov, A. Lyakhov and G. Bianchi, “A Tutorial on IEEE 802.11ax High Efficiency WLANs,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 197-216, 20 September 2018.
- [4] Analytics Vidhya, “10 Real World Applications of Internet of Things (IoT) – Explained in Videos,” 26 August 2016. [Online]. Available: <https://www.analyticsvidhya.com/blog/2016/08/10-youtube-videos-explaining-the-real-world-applications-of-internet-of-things-iot/>. [Accessed 20 March 2020].
- [5] K. L. Lueth, “The 10 most popular Internet of Things applications right now,” 2 February 2015. [Online]. Available: <https://iot-analytics.com/10-internet-of-things-applications/>. [Accessed 20 March 2020].
- [6] C. White, *Data Communications and Computer Networks*, 4th ed., Boston, Massachusetts: Course Technology, 2007, pp. 140-143.
- [7] Wikipedia, “Frequency-division multiplexing,” 16 February 2016. [Online]. Available: https://en.wikipedia.org/wiki/Frequency-division_multiplexing. [Accessed 23 March 2020].
- [8] S. Vyas, “Multiplexing & Multiplexing Hierarchy,” 28 September 2014. [Online]. Available: <https://www.slideshare.net/srashtivyas7/multiplexing-ppt15-sep>. [Accessed 23 March 2020].
- [9] C. Langton, “Orthogonal Frequency Division Multiplex (OFDM) Tutorial,” 2004. [Online]. Available: <http://complextoreal.com/wp-content/uploads/2013/01/ofdm2.pdf>. [Accessed 21 March 2020].
- [10] F. Ouyang, *Digital Communication For Practicing Engineers*, Hoboken, New Jersey: John Wiley, 2019, pp. 453-456.
- [11] ElectronicsNotes, “What is OFDM: Orthogonal Frequency Division Multiplexing,” [Online]. Available: <https://www.electronics-notes.com/articles/radio/multicarrier-modulation/ofdm-orthogonal-frequency-division-multiplexing-what-is-tutorial-basics.php>. [Accessed 23 March 2020].
- [12] S. Ohno, E. Manasseh and M. Nakamoto, “Preamble and pilot symbol design for channel estimation in OFDM systems with null subcarriers,” *Wireless Communications and Networking 2011*, 3 June 2011.

- [13] Keysight Technologies, "Pilot Tracking (802.11a/g/j/p OFDM)," [Online]. Available: http://rfmw.em.keysight.com/wireless/helpfiles/89600b/webhelp/subsystems/wlan-ofdm/Content/dlg_ofdm_adv_pilottracking.htm. [Accessed 30 March 2020].
- [14] NetworkWorld, "Why is OFDMA a Magical Feature in the 802.11ax Standard?," 2018. [Online]. Available: <https://www.networkworld.com/article/3315056/why-is-ofdma-a-magical-feature-in-the-802-11ax-standard.html>. [Accessed 24 March 2020].
- [15] C. Shahriar, M. L. Pan, M. Lichtman, T. C. Clancy, R. McGwier, R. Tandon, S. Sodagari and J. H. Reed, "PHY-Layer Resiliency in OFDM Communications: A Tutorial," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 292-314, 2015.
- [16] Sivarajan, "WiFi moves to OFDMA," Alethea, 2 August 2019. [Online]. Available: <https://alethea.in/wifi-ofdma/>. [Accessed 24 March 2020].
- [17] D. Coleman, "OFDM and OFDMA Subcarriers – What Are the Differences?," 15 October 2018. [Online]. Available: <https://www.extremenetworks.com/extreme-networks-blog/ofdm-and-ofdma-subcarriers-what-are-the-differences/>. [Accessed 24 March 2020].
- [18] D. Bin, "OFDMA-BASED RESOURCE ALLOCATION FOR WIRELESS COMMUNICATION SYSTEMS," 21 July 2010. [Online]. Available: <https://scholarbank.nus.edu.sg/handle/10635/22826>. [Accessed 25 March 2020].
- [19] C. Shahriar, S. Sodagari, R. McGwier and T. C. Clancy, "Performance impact of asynchronous off-tone jamming attacks against OFDM," *2013 IEEE International Conference on Communications (ICC)*, pp. 2177-2182, 7 November 2013.
- [20] C. Shahriar, T. C. Clancy and R. W. McGwier, "Equalization attacks against OFDM: analysis and countermeasures," *Wireless Communication Mobile Computing*, vol. 16, no. 13, pp. 1809-1825, 27 November 2015.
- [21] L. Jun, J. H. Andrian and C. Zhou, "Bit Error Rate Analysis of jamming for OFDM systems," *2007 Wireless Telecommunications Symposium*, pp. 1-8, 27 April 2007.
- [22] T. C. Clancy, "Efficient OFDM Denial: Pilot Jamming and Pilot Nulling," *2011 IEEE International Conference on Communications (ICC)*, pp. 1-5, 29 July 2011.
- [23] Y. Segal, Z. Hadad, I. Kitroser and Y. Lieba, "OFDM preamble structure analysis and proposal," 3 July 2001. [Online]. Available: http://ieee802.org/16/tg3_4/contrib/80216abc-01_04.pdf. [Accessed 4 April 2020].
- [24] H. Rahbari, M. Krunz and L. Lazos, "Swift Jamming Attack on Frequency Offset Estimation: The Achilles' Heel of OFDM Systems," *IEEE Transactions on Mobile Computing*, vol. 15, no. 5, pp. 1264-1278, 1 May 2016.
- [25] M. J. L. Pan, T. C. Clancy and R. W. McGwier, "Phase warping and differential scrambling attacks against OFDM frequency synchronization," *2013 IEEE*

International Conference on Acoustics, Speech and Signal Processing, pp. 2886-2890, May 2013.

- [26] H. Alakoca, G. K. Kurt and C. Ayyıldız, “PHY based Jamming attacks against OFDM systems: A measurement study,” *2017 25th Telecommunication Forum (TELFOR)*, pp. 1-4, 21 November 2017.
- [27] J. Xu, K. Li, L. Duan and R. Zhang, “Proactive Eavesdropping via Jamming over HARQ-Based Communications,” *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1-6, December 2017.
- [28] TP-Link, “AX1500 Wi-Fi 6 Router,” [Online]. Available: <https://www.tp-link.com/sg/home-networking/wifi-router/archer-ax10/>. [Accessed 8 April 2020].
- [29] Intel, “Intel® Wi-Fi 6 AX200,” [Online]. Available: <https://ark.intel.com/content/www/us/en/ark/products/189347/intel-wi-fi-6-ax200.html>. [Accessed 8 April 2020].
- [30] die.net, “urandom(4) - Linux man page,” [Online]. Available: <https://linux.die.net/man/4/urandom>. [Accessed 12 April 2020].
- [31] IMDA, “Technical Specification - Short Range Devices,” April 2018. [Online]. Available: <https://www.imda.gov.sg/-/media/Imda/Files/Regulation-Licensing-and-Consultations/ICT-Standards/Telecommunication-Standards/Radio-Comms/IMDATSSRD.pdf>. [Accessed 7 April 2020].
- [32] P. Riihikallio, “8 reasons to turn down the transmit power of your Wi-Fi,” 21 October 2017. [Online]. Available: <https://metis.fi/en/2017/10/txpower/>. [Accessed 11 April 2020].
- [33] L. Nassef, “On the effects of fading and mobility in on-demand routing protocols,” *Egyptian Informatics Journal*, vol. 11, no. 2, pp. 67-74, 2010.
- [34] Rohde & Schwarz, “IEEE 802.11ax Technology Introduction White Paper,” [Online]. Available: <http://resources.rohde-schwarz-usa.com/c/1-ma222-0e-ieee80211>. [Accessed 12 April 2020].
- [35] Amazon, “WiFi 6 AX200 WiFi Adapter for Windows 10 64bit Chrome OS and Linux Laptop or Desktop PCs-802.11AX 2.4GHz 574Mbps or 5GHz 2.4Gbps(160MHz) with Bluetooth 5.0-Intel WiFi 6 AX200 NGW,” [Online]. Available: <https://www.amazon.com/WiFi-AX200-PCs-802-11AX-Bluetooth-NGW/dp/B07TLBNSZQ>. [Accessed 8 April 2020].
- [36] everythingRF, “R&S Signal Generator Can Now Generate Digital Satellite TV Signals up to 40 GHz,” 12 May 2017. [Online]. Available: <https://www.everythingrf.com/News/details/4111-R-S-Signal-Generator-Can-Now-Generate-Digital-Satellite-TV-Signals-up-to-40-GHz>. [Accessed 9 April 2020].

APPENDIX A – C Source Code used for UDP sending and receiving

Header file used by both programs to send and receive the file (headsock.h):

```
// headfile for TCP program
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <netdb.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>
#include <math.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <errno.h>
#include <fcntl.h>
#include <sys/time.h>
#include <arpa/inet.h>
#include <stdbool.h>

#define ANSI_COLOR_RED      "\x1b[31m"
#define ANSI_COLOR_GREEN    "\x1b[32m"
#define ANSI_COLOR_YELLOW   "\x1b[33m"
#define ANSI_COLOR_BLUE     "\x1b[34m"
#define ANSI_COLOR_MAGENTA  "\x1b[35m"
#define ANSI_COLOR_CYAN     "\x1b[36m"
#define ANSI_COLOR_RESET    "\x1b[0m"

#define NEWFILE (O_WRONLY|O_CREAT|O_TRUNC)
#define MYUDP_PORT 5350
#define DATALEN 110000 //Data unit length
#define BUFSIZE 110000
#define PACKLEN 508
#define HEADLEN 8

struct pack_so //data packet structure
{
    uint32_t num; // the sequence number
    uint32_t len; // the packet length
    char data[DATALEN]; //the packet data
};

struct ack_so
{
    uint8_t num;
    uint8_t len;
};
```

C Source Code for Sending:

```
/******
udp_client.c: the source file of the client in udp
*****/
// #define DEBUGP          // Uncomment statement to enable debugging mode

// Print macros
#ifdef DEBUGP
    #define DEBUG_PRINT(x)          printf(x)
    #define DEBUG_PRINTTWO(x,y)    printf(x,y)
#else
    #define DEBUG_PRINT(x)
    #define DEBUG_PRINTTWO(x,y)
#endif

#include "headsock.h"

void startClient(FILE *fp, int sockfd, struct sockaddr *addr, int addrlen);
void tv_sub(struct timeval *out, struct timeval *in);

int main(int argc, char *argv[]) {
    int sockfd;
    struct sockaddr_in ser_addr;
    char **pptr;
    struct hostent *sh;
    struct in_addr **addrs;
    FILE *fp;

    if (argc != 2) {
        printf("Insufficient Parameters / Parameters mismatch");
        exit(0);
    }

    // Get Host Info
    if ((sh = gethostbyname(argv[1])) == NULL) {
        printf("Error when gethostbyname");
        exit(0);
    }

    // Create Socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        printf("Error in Socket");
        exit(1);
    }

    addrs = (struct in_addr **)sh->h_addr_list;
    printf(ANSI_COLOR_CYAN "Canonical name: %s\n" ANSI_COLOR_RESET, sh-
>h_name);
    for (pptr = sh->h_aliases; *pptr != NULL; pptr++) {
        printf(ANSI_COLOR_GREEN "Aliases name: %s\n" ANSI_COLOR_RESET,
*pptr);
    }
}
```

```

switch (sh -> h_addrtype) {
    case AF_INET:
        printf(ANSI_COLOR_MAGENTA "AF_INET\n" ANSI_COLOR_RESET);
        break;

    default:
        printf("Unknown addrtype\n");
        break;
}

ser_addr.sin_family = AF_INET;
ser_addr.sin_port = htons(MYUDP_PORT);
memcpy(&(ser_addr.sin_addr.s_addr), *addrs, sizeof(struct in_addr));
bzero(&(ser_addr.sin_zero), 8);

// Open file
if ((fp = fopen("myfile.txt", "r+t")) == NULL) {
    printf("File does not exist\n");
    exit(0);
}

// Start Client transmission
startClient(fp, sockfd, (struct sockaddr *)&ser_addr, sizeof(struct
sockaddr_in));
close(sockfd);
exit(0);
}

void startClient(FILE *fp, int sockfd, struct sockaddr *addr, int addrlen)
{
    char sends[DATALEN];
    struct timeval sendt, recvt;

    // Scans file for size
    long lsize;
    fseek(fp, 0, SEEK_END);
    lsize = ftell(fp);
    rewind(fp);
    printf(ANSI_COLOR_BLUE "File length: %d bytes\n" ANSI_COLOR_RESET,
(int)lsize);

    // allocate memory to contain the whole file.
    char *buf;
    buf = (char *)malloc(lsize);
    if (buf == NULL) exit(2);

    // Copy the file into the buffer.
    fread(buf, 1, lsize, fp);
    // Append EOF byte
    buf[lsize] = '\0';

    // get time at beginning of file transfer, stored at sendt
    gettimeofday(&sendt, NULL);

    int ci = 0, slen; // File Read Index, Segment length

```

```

while (ci <= lsize) {
    // Determine segment length
    if ((lsize + 1 - ci) <= DATALEN) {
        // Remainder
        slen = lsize + 1 - ci;
    }
    else {
        slen = DATALEN;
    }

    memcpy(sends, (buf + ci), slen);    // Copy segment to payload
    DEBUG_PRINTTWO("Sent payload of %d Bytes\n", slen);
    if(sends[slen - 1] == '\0') {
        DEBUG_PRINT(ANSI_COLOR_CYAN "EOF sent\n"
        ANSI_COLOR_RESET);
    }

    // Send payload
    if (sendto(sockfd, &sends, slen, 0, addr, addrlen) == -1) {
        printf(ANSI_COLOR_RED "Send Error\n" ANSI_COLOR_RESET);
        exit(1);
    }
    // Increment file read index
    ci += slen;
    // sleep for 1 millisecond
    usleep(1000);
}
printf(ANSI_COLOR_GREEN "File sent successfully!\n"
ANSI_COLOR_RESET);

// Get time at end of file transfer
gettimeofday(&recvt, NULL);
// Get whole transfer time
tv_sub(&recvt, &sendt);

float timeInterval = 0.0;
timeInterval += (recvt.tv_sec)*1000.0 + (recvt.tv_usec)/1000.0;
float transRate = (lsize/timeInterval);
printf(ANSI_COLOR_BLUE "Time (ms) : %.3f, Data sent (byte): %ld\nData
rate: %f (Kbytes/s)\n" ANSI_COLOR_RESET, timeInterval, lsize, transRate);
}

/**
 * calculate time interval
 */
void tv_sub(struct timeval *out, struct timeval *in) {
    if ((out->tv_usec -= in->tv_usec) < 0) {
        --out->tv_sec;
        out->tv_usec += 1000000;
    }
    out->tv_sec -= in->tv_sec;
}

```

C Source Code for Receiving:

```
/******
udp_ser.c: the source file of the server in udp transmission
******/
// #define DEBUGP          // Uncomment statement to enable debugging
mode

// Print macros
#ifdef DEBUGP
    #define DEBUG_PRINT(x)          printf(x)
    #define DEBUG_PRINTTWO(x,y)    printf(x,y)
#else
    #define DEBUG_PRINT(x)
    #define DEBUG_PRINTTWO(x,y)
#endif

#include "headsock.h"

void str_serl(int sockfd); // transmitting and receiving function

int main(int argc, char *argv[]) {
    int sockfd;
    struct sockaddr_in my_addr;

    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) { //
Create socket
        printf("Socket Error");
        exit(1);
    }

    my_addr.sin_family = AF_INET;
    my_addr.sin_port = htons(MYUDP_PORT);
    my_addr.sin_addr.s_addr = INADDR_ANY;
    bzero(&(my_addr.sin_zero), 8);

    // Bind socket
    if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct
sockaddr)) == -1) {
        printf("Binding Error");
        exit(1);
    }
    printf(ANSI_COLOR_BLUE "Start receiving\n" ANSI_COLOR_RESET);

    // Keeps receiving any packets from client. Does not terminate
until Ctrl + C sent
    while (true) {
        str_serl(sockfd);
    }
    close(sockfd);
    exit(0);
}
```

```

void str_serl(int sockfd) {
    struct sockaddr_in addr;
    int addrlen = sizeof(struct sockaddr_in);
    char buf[BUFSIZE];
    FILE *fp;
    long lseek = 0;
    bool end = false;
    bool processingFile = false;

    while (!end) {
        char recvs[DATALEN];
        int payloadSize = 0;
        payloadSize = recvfrom(sockfd, &recvs, DATALEN, 0,
(struct sockaddr *)&addr, &addrlen);
        if (payloadSize) {
            processingFile = true;
            if (payloadSize == -1) { // Error Handling
                printf(ANSI_COLOR_RED "ERROR!"
ANSI_COLOR_RESET);
                exit(1);
            }
            DEBUG_PRINTTWO("Received payload of %d Bytes\n",
payloadSize);
            if (recvs[payloadSize - 1] == '\0') { // Check for
EOF
                end = true;
                payloadSize--;
                DEBUG_PRINT(ANSI_COLOR_CYAN "EOF received\n"
ANSI_COLOR_RESET);
            }
            memcpy((buf + lseek), recvs, payloadSize);
            lseek += payloadSize;
        }
    }
    if (processingFile) {
        if ((fp = fopen("myUDPreceive.txt", "wt")) == NULL) {
            printf("File does not exist\n");
            exit(0);
        }
        fwrite(buf, 1, lseek, fp); // Write data into file
        fclose(fp);
        printf(ANSI_COLOR_GREEN "File successfully received!\n"
ANSI_COLOR_RESET);
        printf(ANSI_COLOR_BLUE "Total data received: %d bytes\n"
ANSI_COLOR_RESET, (int)lseek);
        processingFile = false;
    }
}

```

APPENDIX B – C Source Code used for calculating Hamming Distance

```
#include "headsock.h"

long hammingDist(FILE *fp1, FILE *fp2);

int main(int argc, char *argv[]) {
    FILE *fp1;
    FILE *fp2;

    // Open master file
    if ((fp1 = fopen("myfile.txt", "r+t")) == NULL) {
        printf("Master File does not exist\n");
        exit(0);
    }

    // Open received file
    if ((fp2 = fopen("myUDPreceive.txt", "r+t")) == NULL) {
        printf("Received File does not exist\n");
        exit(0);
    }
    printf("Hamming Distance: %ld\n", hammingDist(fp1, fp2));
    exit(0);
}

long hammingDist(FILE *fp1, FILE *fp2) {
    char *buf1;
    char *buf2;

    long lsize;
    fseek(fp1, 0, SEEK_END);
    lsize = ftell(fp1);
    rewind(fp1);

    // allocate memory to contain both files.
    buf1 = (char *)malloc(lsize);
    if (buf1 == NULL) exit(2);
    buf2 = (char *)malloc(lsize);
    if (buf2 == NULL) exit(2);

    // Copy both files into the buffer.
    fread(buf1, 1, lsize, fp1);
    fread(buf2, 1, lsize, fp2);
    // Append EOF byte
    buf1[lsize] = '\0';
    buf2[lsize] = '\0';

    int i = 0;
    long count = 0;
    while(buf1[i] != '\0') {
        if (buf1[i] != buf2[i])
            count++;
        i++;
    }
    return count;
}
```