CROSSING THE AIR GAP — AN ULTRASONIC COVERT CHANNEL

PONTAGE DU RÉSEAU SÉPARÉ — UN CANAL ULTRASONORE CACHÉ

A Thesis Submitted to the Division of Graduate Studies
of the Royal Military College of Canada
by

Wesley Wong, B.Cmp.H.

In Partial Fulfillment of the Requirements for the Degree of
Master of Applied Science in Electrical and Computer Engineering

September, 2018

Abstract

Commonly implemented in high-security computer networks, an air gap physically isolates systems from external threats. Without Internet-connectivity, another medium is needed to transfer data to/from an air-gapped network. In this work, the possibility of inaudible acoustic channels in the ultrasonic frequency range is investigated.

An ultrasonic channel with the Goertzel algorithm, cross-correlation, pulse shaping, and a medium access control protocol was developed. Command-line interaction was demonstrated via the ultrasonic channel. With binary frequency-shift keying (BFSK), speeds of over 250 bit/s at a distance of 2 m were observed. Quadrature frequency-shift keying (QFSK) was confirmed possible within the ultrasonic range to yield speeds of about 500 bit/s. Differential phase-shift keying (DPSK) was shown to be functional. The ultrasonic channel may be hidden in audible noise such as music. The channel software was also capable of transferring data directly via auxiliary (AUX) audio cable.

Résumé

Généralement mis en œuvre dans les réseaux informatiques de haute sécurité, la séparation physique isole les systèmes des menaces externes. Sans l'accès à Internet, une autre méthode est nécessaire pour le transfert de données du réseau séparé. Dans ce travail, la faisabilité d'employer des canaux acoustiques inaudibles dans la gamme de fréquences ultrasonores est étudiée.

Un canal ultrasonore utilisant l'algorithme de Geortzel, la corrélation croisée, la conformations des impulsions, et un protocole de contrôle d'accès au support fut développé. La functionnalité de l'interface en ligne de commande a été démontrée via le canal ultrasonore. Par une modulation par déplacement de fréquence binaire (BFSK), des vitesses dépassant 250 bit/s à une distance de 2 m ont été observées. La modulation par déplacement de fréquence en quadrature (QFSK) a été confirmée possible dans le domaine ultrasonore, donnant des vitesses d'environ 500 bit/s. La modulation par changement de phase différentiel (DPSK) s'est avérée fonctionnelle. Le canal ultrasonore peut être caché dans le bruit tel que la musique. Le logiciel du canal ultrasonore est capable de transférer les données directement par un câble auxiliaire (AUX).

Contents

List of Tables

List of Figures

List of Listings

List of Equations

List of Abbreviations and Acronyms

ACK	acknowledgement
APT	advanced persistent threat
ARQ	automatic repeat request
ASK	amplitude-shift keying
AUX	auxiliary
BASK	binary amplitude-shift keying
BER	bit error rate
BFSK	binary frequency-shift keying
BIOS	basic input/output system
BPSK	binary phase-shift keying
CIA	Central Intelligence Agency
CRC	cyclic redundancy check
CSMA/CD	carrier-sense multiple access with collision detection
DBPSK	differential binary phase-shift keying
DFT	discrete Fourier transform
DOD	Department of Defense
DPSK	differential phase-shift keying
DQPSK	differential quadrature phase-shift keying
FFT	fast Fourier transform
FPSK	frequency phase-shift keying
FSK	frequency-shift keying
FTP	File Transfer Protocol
GFSK	Gaussian frequency-shift keying
HID	human interface device
HTTP	Hypertext Transfer Protocol
ICS	industrial control system
IDS	intrusion detection system
IIR	infinite impulse response

IP	Internet Protocol
IPS	intrusion prevention system
MAC	media access control
MD5	Message Digest 5
NAK	negative acknowledgement
NATO	North Atlantic Treaty Organization
NIPRNet	Non-classified Internet Protocol Router Network
NSA	National Security Agency
OOB	out-of-band
OOK	on-off keying
OSI	Open Systems Interconnection
PCM	pulse-code modulation
PLC	programmable logic controller
PSK	phase-shift keying
QAM	quadrature amplitude modulation
QASK	quadrature amplitude-shift keying
QFSK	quadrature frequency-shift keying
QPSK	quadrature phase-shift keying
RSA	Rivest–Shamir–Adleman
RX	receive
SCADA	supervisory control and data acquisition
SIPRNet	Secret Internet Protocol Router Network
SSH	Secure Shell
TCP	Transmission Control Protocol
TX	transmit
U.S.	United States
USB	Universal Serial Bus

1 Introduction

Computer networks connected to the Internet are exposed to external threats and attackers. It follows that high security computer networks operate disconnected from the Internet and external networks. This separation is known as the air gap, where secure cyberspace is physically isolated from outside connectivity. An air gap significantly minimizes the attack surface of a computer network. Despite the potential for vulnerabilities to exist in software, attacking an air-gapped network is more difficult because there are no existing communications channels to the outside. Air gaps are known as the "*sine qua non* of information systems security" [1].

Air gaps are implemented in high security environments, including military, intelligence, critical infrastructure, and corporate networks. These networks may still be breached. Malware may cross the air gap into secure networks through removable media, such as Universal Serial Bus (USB) devices. Normally, after malware is installed on the air-gapped system, it has no medium to communicate back to the attacker. A communications channel to an air-gapped network means that data can be transferred to and from the secure network. A post exploitation channel would allow an attacker to access and control systems across the air gap, acting as a medium for receiving commands and exfiltrating data off of the network.

Computer network security is enhanced with an understanding of potential attack vectors before they can be exploited by an adversary. This thesis researches the feasibility of ultrasonic covert channels in the context of air gaps, and also as a medium for general-purpose information transfer.

1.1 Motivation

Malware has been known to use the insertion of removable media into secure systems as a vector into air-gapped networks [1]. In these cases, the malware was able to spread to other internal hosts and carry out its attack automatically within the network [2]. However, exfiltration of data may have been an additional step after the insertion. Extrusion of information relies on being able to send data back out to an external host, perhaps utilizing the same channel as the ingress vector. Radio transceiver implants have been used to establish rogue communications links with air-gapped hosts [3]. A radio implant allows an external attacker within the vicinity of several miles away to communicate with the target system. The communications channel serves as a command-and-control channel used to monitor and exfiltrate data from the infected system.

Our research explores the feasibility of a communications link via inaudible high-frequency sound over common speakers and microphones. This entails the transmission of ultrasonic signals via speakers and reception through microphones in order to establish a communications link over the air gap. We consider the ultrasonic channel as an attack vector within the context of computer network security. The concept of malicious ultrasonic channels has implications to air gap security as speakers and microphones are commonly attached and built into many computing devices (e.g., laptops, smartphones, and tablets). Our research demonstrates the possibility of such a channel through the development of proof-of-concept software, which relays data ultrasonically between computing devices using speakers and microphones.

Audio in air is currently an uncommon medium for communications between computing devices. Research that features ultrasonic communications between computers in the context of network security is recent and little is known about their performance. Our original motivation in 2012 was to publish a proof of concept for ultrasonic channels as there was no other research on this subject at the time. However, as time has passed, other research has been published. In 2013, research indicated that ultrasonic channels in air have low speeds, around a bitrate of 20 bit/s [4]. A human-audible clicking noise issue was reported in modulation, detracting from the covertness of the channel [4]. Our research shows an increased speed that is usable for command-line interaction and also resolves the clicking noise issue.

This thesis examines several fundamental characteristics of ultrasonic channels: implementation of digital modulation schemes, applicable channel access methods, and generic compatibility. Reliable data transfers above the audible range were verified on consumer-grade equipment. Validation of the channel was performed by measuring the channel bitrate, frequency discrimination, and operation in noise models. This research demonstrates ultrasonic communications between computers and identifies the corresponding vulnerability for air-gapped network defence.

1.2 Operational Scenario

The ultrasonic channel is considered in the context of an attack scenario with remote control and data extrusion as the objective. The ultrasonic channel is established following from an exploit or breach. This may not require physical presence if a physical medium (e.g., USB drive) can be delivered or subverted. The channel software is installed following from an entry vector. Systems may then relay data ultrasonically to compromised Internet-connected devices in their proximity. The channel is used to receive commands via microphones and transmit data via speakers. The air in proximity of the speakers and microphones is modulated as ultrasonic signals, acting as a medium for malware to receive commands and transmit data externally. The ultrasonic channel is functional for command-line interaction and small data transfers. The channel may also be able to transmit through a Faraday cage/shield, which blocks electromagnetic emanations, but is permeable to acoustic waves.

1.3 Aim

The aim of this research was to identify the vulnerability presented by ultrasonic channels in air-gapped networks and to investigate the performance of such channels as a general-purpose communications medium. The identification of vulnerability includes an exploration and characterization of ultrasonic modulation schemes using standard computer hardware. The suitability of such channels as a communications system for malware will be investigated through integration of low-level ultrasonic communications with common high-level network protocol standards.

1.4 Research Activities

The research conducted in this thesis entailed the design and development of proof-of-concept software for relaying data ultrasonically between computers. This included the low-level modulation/demodulation of acoustic waves at ultrasonic frequencies, implementation of a suitable data-link communications protocol, and interfacing the channel with common higher-level computer communications protocols and applications in a generic manner. Verification of the channel was performed by ensuring the accurate transfer of information. Validation was performed by measuring the channel bitrate, frequency discrimination, and operation in noise models. Knowledge regarding channel characteristics was gained in development and in validation testing. This research contributes to understanding and building better ultrasonic channels.

1.5 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 covers the background context and existing technical methods applied in this research. Chapter 3 presents the design and development of our ultrasonic channel. Chapter 4 describes the validation testing conducted in order to demonstrate achievement of our aim. Chapter 5 concludes with a discussion of the work.

2 Literature Review

We have introduced the purpose of exploring the ultrasonic channel as a potential vulnerability in air-gapped networks. This chapter covers the background context to further illustrate the motivation for studying the problems in air gap security. Next, existing knowledge in acoustic communications is reviewed. Lastly, the telecommunications theory and technical methods applied in our channel are described.

2.1 Context of Study

This thesis investigates the feasibility of an ultrasonic channel to exfiltrate information over an air gap. To further understand the scenarios in which an ultrasonic channel could be employed, we have reviewed the relevant background context.

2.1.1 Computer Network Security

A computer network may be assessed by considering the *CIA triad* of information security: **C**onfidentiality, **I**ntegrity, and **A**vailability of information on the network. Confidentiality is compromised where an attacker can access and view the data. Integrity is compromised where an attacker can change the data. Availability is compromised where an attacker can disrupt the systems from normal operation. In air-gapped networks, integrity and availability can be compromised by fire-and-forget type malware. As seen with Stuxnet, malware delivered via USB was propagated through the network (including specialized devices) to reach and disable target nuclear centrifuge controllers [5]. In contrast to fire-and-forget type malware, data exfiltration requires a communications channel back out of the network. This channel serves to exfiltrate information from secure systems to an external attacker, thus compromising confidentiality.

Cybersecurity is of interest to military, intelligence, political, and commercial entities. An informational advantage is gained through the exploitation of systems where sensitive information is stored. Information gathering may be a part of foreign government intelligence routines. For example, Chinese advanced persistent threat (APT) operations involve the infiltration of air-gapped networks for the acquisition of private government information relating to socio- and geopolitical influence [6]. The motive to attack air-gapped networks may be to thwart capability that poses a threat in other domains, such as Stuxnet, which targeted nuclear capability. A channel that bridges the air gap can disable systems when they are needed the most and disrupt systems in part of larger operations. Attacks that extend outside of the cyber domain and facilitate real-world damage are described as having “kinetic” [7] effects. Cyberspace has unique properties that make the attribution of an attack difficult [7]. Our research contributes to understanding ultrasonic channels from a network attack perspective.

2.1.2 Command-and-Control Channel

A command-and-control channel can be established after a vulnerability is exploited. Malware installed on a system following an exploit may allow an attacker to maintain access to the compromised system. The communications channel serves as a command-and-control channel, used to control a compromised system by receiving instructions for it to execute. Commands are executed and the corresponding data is returned to the attacker. Such a channel may be used to monitor, alter, and disrupt the operation of systems on the network. A command-and-control channel established to a system on the network can function as a pivot point to attack other internal systems. The channel allows the attacker to maintain access to the network. Such a channel provides access to infected systems, allowing the user to adapt an attack upon a network over time.

A basic command-and-control channel can be made with a sockets relay program such as Netcat, executing a command-line interpreter such as Command Prompt (in Windows) or Unix shell to provide command-line access to the remote machine [8]. Command-and-control channels can be more sophisticated to hide the presence of the channel software on the system (e.g., through a rootkit) [9]. This thesis investigates the possibility of establishing command-and-control channels through speakers and microphones at ultrasonic frequencies. Insight regarding the existence of covert channels is necessary for their identification and detection.

2.1.3 Covert Channel

A covert channel can be considered any medium that is unintended for data transfer and hidden from detection. Covert channels provide a means of communication that is concealed from normal observation through obfuscation. Data transferred over the channel is not detected because the existence of the channel itself is hidden. In computer networks, a covert channel enables the capability to transfer data despite network restrictions (e.g., a firewall) [10]. The channel is hidden from normal traffic analysis and detection. The covert channel may be used as a command-and-control channel.

An example of a covert channel is the variation of a data field in an existing protocol in a manner such that it transmits data [10]. The covert channel would appear to be normal protocol communications in traffic analysis. Since the covert communications are hidden within normal communications, intrusion detection systems (IDS) may not be able to match the network traffic to known signatures, thus the channel would remain undetected [10]. Intrusion prevention systems (IPS) would be bypassed similarly. After a system is exploited, a covert command-and-control channel would provide continued access to target systems and networks. Data could be exfiltrated over a period of time, undetected by traffic analysis. In this study, the covert channel exists in air as ultrasonic signals between speakers and microphones. The channel does not hide within an existing protocol but rather an entirely different medium. Channels that use non-conventional mediums have been called "out-of-band" (OOB) [11] covert channels.

2.2 Air Gap

It is widely accepted that "in order to provide the highest levels of security for computer systems, they must not be connected to the Internet" [1]. The Internet and external connectivity are considered a threat in context of high security networks. The attack surface of a network connected to the Internet is greater than that of a system isolated from external connectivity. Even if incoming connections are blocked, outgoing connections would likely be allowed for internal users to retrieve information from the Internet. The Internet connection makes it possible to maintain remote access after a breach (e.g., a social-engineered client-side attack). With an air gap, the egress of information may be difficult even if a system has been compromised.

An air gap can be implemented to secure the network by removing any communications links to the outside, thus isolating the network. Air gaps are often implemented for high security networks [1]. Examples include intelligence and military networks, financial networks, industrial control systems (ICS), supervisory control and data acquisition (SCADA) systems, and critical infrastructure such as nuclear facilities and aviation systems. Companies may use *cold storage*, where data is stored offline. Cold storage may be required for legal compliance, security for storing sensitive data offline, inexpensive long-term storage of data that requires minimal access, and backups or archives [12]. Air gaps have also gained popularity for the secure storage of cryptocurrency such as Bitcoin [1]. Computers that hold sensitive research at some universities are required to be air-gapped [1]. The capability to communicate across an air gap is of relevance to cybersecurity professionals who use air gaps to secure information systems.

2.2.1 Air Gap Attacks

Several known vectors have been used in attacking air-gapped networks. The following subsections describe methods that have been used to breach air gaps.

2.2.1.1 Physical Access

The most direct type of attack is physical access. Social engineering and other physical penetration testing methods can be used to gain access to secured areas. The *evil maid* attack is an example of a physical access attack where the attacker gains physical access to tamper with systems. This type of attack can even defeat full-disk encryption by replacing the bootloader with one that records the encryption password [13]. Hardware keyloggers may also be implanted. It is also worth mentioning the insider threat. In the least, this could be in the form of a disgruntled employee or contractor who sabotages systems [14]. With physical access, an attacker completely defeats the air gap.

2.2.1.2 Removable Media

An attacker may not need to have direct physical access to systems if network users insert infected devices into their computers [1]. Network users may need external information on the air-gapped network as tasks arise that require new data. As such, users may transfer data between air-gapped and untrusted systems. Removable media storage devices are used to move data onto and off of the air-gapped network. This in-person sharing of data via removable media has been called “sneakernet” [7], where network users physically transfer data between each other via removable media. As air-gapped systems require external information and users transfer data to and from the network, malware may also utilize the same pathways to access air-gapped systems. The insertion of removable media has been a vector for malware to be transferred into secure systems [1].

Central Intelligence Agency (CIA) malware dubbed Brutal Kangaroo enables USB drives to breach air gaps [15]. A USB with Brutal Kangaroo infects the system it is plugged into, spreading to any USB drives that are plugged into the infected system [15]. This works with the aim of spreading to reach any air-gapped systems.

Similar to removable media, the personal devices of employees may also be compromised. Untrusted devices may be plugged in to an air-gapped system for a variety of reasons (e.g., charging power via USB cable) [14].

2.2.1.3 Agent.btz

An example of removable media attacks is Agent.btz, a worm that breached United States (U.S.) military air-gapped networks in 2008. Agent.btz was not specifically designed to target military networks and its origins are uncertain although suspected to be from Russia or China [16]. By first gaining access to the U.S. Department of Defense’s (DOD) Non-classified Internet Protocol Router Network (NIPRNet) which was connected to the Internet, Agent.btz could then transfer itself to USB drives [7]. A USB drive containing Agent.btz was then inserted unwittingly into a computer that was part of the air-gapped Secret Internet Protocol Router Network (SIPRNet) [7]. Air-gapped systems may not be updated/patched frequently as they are disconnected from the Internet, meaning that software vulnerabilities often exist [7]. Within several hours, Agent.btz had spread from military computers in the Middle East to systems in the U.S. Central Command; thousands of classified DOD computers were infected [7]. The infection was described "the most significant breach of U.S. military computers ever" [17]. Storage media including USB drives, optical discs, external hard drives, and floppy disks were banned shortly afterwards to prevent further spreading of the worm [18]. Efforts to remove Agent.btz and defend against such attacks led to the creation of the U.S. Cyber Command [17].

Agent.btz infected USB drives by detecting the insertion of removable media and copying itself over to them [19]. An autorun.inf file (used to specify a program to execute upon insertion) would be written to the removable media [19]. Once this USB drive is inserted into an AutoRun-enabled Windows computer, the copy of Agent.btz on the drive executes as specified by the autorun.inf file. The USB drive infected computers it was plugged into, allowing the malware to cross over onto the air-gapped network. Within the air-gapped network, any systems that mounted an infected shared network resource were also infected; similar to removable media, AutoRun would use the autorun.inf located on the mounted drive and run its copy of Agent.btz [19]. As such, Agent.btz was able to jump across air gaps and spread to other systems on the network. Agent.btz collected system information and opened backdoors as the worm propagated. AutoRun on Windows XP was disabled by default for USB media, but not for disc media [19]. Some USB drives were made to appear as optical media, allowing AutoRun to still launch [19]. AutoRun is currently a subset of AutoPlay, a feature that prompts the user with a selection of options to interact with removable media [19].

2.2.1.4 Stuxnet

Another example of removable media attacks is Stuxnet, malware used to sabotage Iranian nuclear enrichment centrifuges in 2010. The malware was believed to have been a joint American and Israeli effort to hinder nuclear capability in Iran [5]. Like Agent.btz, Stuxnet was able to cross the air gap onto the SCADA network via USB drive. As fire-and-forget malware, Stuxnet spread automatically to other hosts within the network to reach target systems. Stuxnet was specifically targeted to Siemens hardware, which was used to monitor and operate nuclear enrichment centrifuges. Stuxnet modified Siemens project files and replaced certain .dll files to subvert communications with Siemens programmable logic controllers (PLC) [2]. Stuxnet increased centrifuge speeds while displaying normal operation statistics to the monitor, forcing the centrifuges to physically tear apart. Approximately 1 000 centrifuges were destroyed at the nuclear enrichment facility in Natanz [5].

Similar to Agent.btz, Stuxnet also used AutoRun to spread to removable media [5]. Stuxnet did not only depend on the previously described AutoRun vector in Agent.btz. Additionally, Stuxnet could spread through removable media using a vulnerability in the handling of Windows shortcut files (.lnk), allowing the execution of a .dll payload on removable media [19]. The vulnerability in parsing custom icons calls the icon library's DllMain function [20]. Once within a network, Stuxnet was able to spread automatically to other hosts through several zero-day exploits.

A variant of Stuxnet, Flame, is also known to cross air gaps through the .lnk shortcut and AutoRun vulnerabilities [21]. Additionally, Flame has also been reported to have Bluetooth capability [21].

2.2.1.5 BadBIOS

BadBIOS was malware reported by Dragos Ruiu [22]. It had air gap crossing capability via ultrasonic frequencies. The malware resides in the basic input/output system (BIOS) and was said to have self-healing and updating capabilities via ultrasonic channels [22]. Samples of the malware were not made publicly available. An ultrasonic communications capability is the topic of investigation in our research.

Our ultrasonic channel had been developed when BadBIOS was announced in 2013. At the same time, other researchers published on ultrasonic channels in the context of crossing the air gap [4]. Our research confirms that ultrasonic covert channels are feasible and that software can use speakers and microphones to communicate between air gaps.

2.2.1.6 Human Interface Device (HID) Firmware

USB vulnerabilities extend beyond the AutoRun and the .lnk vulnerabilities mentioned in Agent.btz and Stuxnet. Windows, Linux, and MacOS systems have been affected by other USB exploits [20]. USB drivers are vulnerable to exploitation and USB protocols can also be subverted [20].

Other than being mass storage devices, USB is commonly used to attach computer peripherals: keyboard, mouse, webcam, etc. These are considered USB HID class examples. A USB stick meant for storage can be reflashed so that it is recognized as a keyboard, enabling it to “type” in commands. The USB Rubber Ducky is an example of a HID device which emulates a keyboard user [23]. It is possible to turn a regular USB flash drive into a USB Rubber Ducky by reflashing it [24]. When inserted into a computer, it is recognized as a keyboard. Indistinguishable from a real user on the keyboard, the device can run a script of stored keypresses, thereby running the attacker’s commands and programs. Some firmware allows the device to act as both mass storage and a HID keyboard. USBdriveby and BadUSB are examples of USB HID malware. USBdriveby turns a USB drive into a HID device that is able to quickly and covertly run commands (e.g., install a backdoor) by acting as a keyboard [25]. Through BadUSB, malware can hide in peripheral devices with reprogrammed firmware [26].

2.2.1.7 Hardware Implants

Hardware implants are devices dedicated to compromising systems; they can be well hidden, such as in a USB cable [27]. Keyloggers provide a broad functionality by capturing all keystrokes entered on a keyboard. Hardware implants can be software independent, designed to capture keystrokes regardless of the operating system. A disadvantage is that hardware implants exist in physical space, and can be visibly detected.

Devices can be physically added to the network, providing the attacker remote access. A small device can contain an entirely functional system with penetration testing tools. The Pwn Plug is an example of such a tool [28].

A radio transceiver may be a hardware implant, offering an attacker the capability to remotely access the bugged system via radio frequencies [27]. A radio channel acts as a relay to the infected host, running external commands and exfiltrating information from the infected system. The National Security Agency (NSA) has used radio implants (e.g., COTTONMOUTH) to access air-gapped hosts [27]. The NSA ANT catalog features various other hardware implants [27].

2.2.1.8 TEMPEST

Data extrusion in air-gapped networks relates to the study of compromising emanations known as TEMPEST. North Atlantic Treaty Organization (NATO) TEMPEST standards define the distances at which equipment is secured from an attacker in terms of distance (e.g., 1 m, 20 m, 100 m) [29].

Electromagnetic signals emanate from computers, which may allow data to be viewed. Electromagnetic signals are leaked from computer monitors and cables. An attacker can visually see what is on a screen at a distance of 10 m through three plasterboard walls [30]. Wireless and wired keyboards have compromising emanations that can be read at a distance of 20 m [31]. These TEMPEST attacks would not require the system to be exploited in advance.

Acoustic signals from processor vibrations can also leak information such as Rivest–Shamir–Adleman (RSA) private keys [32]. The sound from keys being pressed on a keyboard can be reliably mapped back to their keys [33]. Various other compromising emanations have been examined by Guri et al. including physical, electromagnetic, electric, magnetic, acoustic, thermal, and optical methods [34]. Power lines can be used to exfiltrate data from air-gapped networks [35]. The USBee is malware that makes USB drives emanate readable electromagnetic signals [36].

As in TEMPEST, OOB covert channels may communicate over acoustic, light, seismic, magnetic, thermal, or radio mediums [11]. A distinction can be made between compromising emanations that do not require a system to already be exploited vs. covert channels that are set up after a system is exploited, such as our ultrasonic channel.

2.2.2 Air Gap Defence

The following subsections describe methods used in defending air-gapped systems.

2.2.2.1 Physical Security

The security of an air-gapped network is primarily that of its physical security. Physical security measures may include mechanical or electronic locks, sensors and alarms, video surveillance, and security personnel. As discussed, a breach in physical security does not have to involve the direct presence of a bad actor in person. USB drives containing malware may be inserted unwittingly into secure systems, or a compromised device may be used on the air-gapped network. Malware that connects a remote attacker to a network behind physical security measures effectively bypasses them.

2.2.2.2 Trusted Removable Media

As previously illustrated, the pathways for legitimate data to be transferred may also be pathways for malware to enter. USB is a common medium for malware to be transferred over [1]. Some network administrators have opted to epoxy glue shut USB ports on sensitive systems [37]. USB sanitization is a possibility, where USB drives are plugged into a sanitization device before usage [1]. Some USB drives are designed to not be reflashable. Examples are the Kanguru FlashTrust [38] and Kingston IronKey [39]. Devices may be obtained from trusted suppliers, where the supply chain is secured.

2.2.2.3 Faraday Cage Shielding

As previously mentioned, radio-based transceivers have been used to communicate with air-gapped systems. In addition, TEMPEST attacks allow an attacker to exfiltrate electromagnetic signals. A Faraday cage can be used to block electromagnetic signals from both leaving and entering. A radio transceiver is unable to communicate outside through a Faraday cage. Although a Faraday cage blocks electromagnetic signals, acoustic signals could still go through, depending on whether the cage construction is permeable to air.

2.3 Acoustic Communications

This thesis investigates ultrasonic acoustic communications between computing devices. Acoustic waves, that is, sound waves, are vibrations in air. Sound is a mechanical wave as it requires a physical medium to propagate. Electromagnetic waves (e.g., radio, Wi-Fi) do not require a physical medium such as air to propagate. This distinction is what allows an acoustic channel to function where electromagnetic signals are blocked, such as a Faraday cage. Sound waves are gas pressure oscillations, air molecules compressed and rarefacted, spreading radially. Sound waves propagate as longitudinal waves, but will be represented as transverse waves in this thesis.

Speakers and microphones are electroacoustic transducers. Conventionally, speakers convert electrical signals into acoustic waves by displacing the speaker cone proportionally to the electrical signals [40]. Microphones convert acoustic waves into electrical signals as its diaphragm is displaced proportionally to the acoustic waves [40]. Speakers and microphones are generally designed to function in the audible frequency range, not made for use at ultrasonic frequencies; they are often rated to have frequency response from about 20 Hz to 20 000 Hz. In contrast, electromagnetic waves up to 6 GHz can be used safely for communications [41]. Acoustic waves are expected to have much lower bitrates and act at shorter distances due to the inherent characteristics of the physical air medium. With the emergence of mobile computing, there are a greater number of devices with speakers, microphones, and Internet connectivity. Speakers and microphones are common and readily attachable to computing devices.

The ultrasonic channel uses the range of acoustic waves that is above the human audible range. Sound is audible when acoustic waves stimulate the ear drum, carrying the vibrations through bones to the inner ear where hair cells move correspondingly, sending electrical signals to the brain [42]. Hearing is impaired if these hair cells are damaged [42]. If the acoustic waves occur at high enough frequencies, hair cells are not stimulated, and the sound is inaudible. Aging reduces the ability to hear higher frequencies [43]. Most adults cannot hear higher frequencies (e.g., above 17 900 Hz) [43]. It should be taken into consideration that even if frequencies are inaudible, exposure to acoustic signals in the environment has been documented in producing effects of nausea, headache, and fatigue [44].

2.3.1 Early Modems

Early modems modulated data over a limited audio frequency bandwidth. Dial-up modems used common phone lines designed for voice communication with a frequency range of about 300 Hz to 3300 Hz [45]. The first dial-up modems used full-duplex binary frequency-shift keying (BFSK) [46]. Using four separate frequency channels (two for binary downstream and two for binary upstream), they achieved speeds up to 1.2 kbit/s [46]. Later modems used separate send and receive frequency channels, applying phase-shift keying (PSK) combined with amplitude-shift keying (ASK), known as quadrature amplitude modulation (QAM) [47]. QAM with error correction and echo cancellation increased bitrates to 33.6 kbit/s [47]. Speeds were improved by detecting smaller shifts in phase and amplitude. Telephone networks became more reliable between cities with improvements such as coaxial cable, microwave radio relay, and fiber optic cable [48]. Pulse-code modulation (PCM) and data compression have been used to exceed 56 kbit/s for dial-up modems [49].

2.3.2 Existing Research

The possibility of ultrasonic covert channels has been discussed in the context of computer network security. In 2013, Hanspach and Goetz showed transfer speeds of about 20 bit/s and mentioned an audible clicking noise problem [4]. Around the same time, we had achieved speeds of about 250 bit/s and solved the clicking noise issue. In 2014, Deshotels showed transfers up to 100 ft. and over 300 bit/s at close range [50]. In 2015, Carrara showed speeds of 230 bit/s up to 11 m and presented the concept of an overnight attack, when audible signals can be used with no one around [11].

2.4 Applied Telecommunications

In order to design the channel, we examined signal processing techniques, channel access methods, and compatibility with existing network standards. Modulation schemes and protocols have been developed for electromagnetic communications. The same concepts used for electromagnetic waves can be applied to acoustic waves. Modulation schemes, signal processing techniques, and telecommunications methods were applied in the development of the ultrasonic channel.

2.4.1 Digital Modulation Schemes

Modulation is the process of varying a waveform so that information can be transferred. Data can be conveyed in a waveform by varying its frequency, phase, and amplitude. Digital modulation schemes use discrete changes in the waveform to represent binary symbols. Our research applies these modulation schemes to acoustic waves.

2.4.1.1 Frequency-Shift Keying (FSK)

FSK uses a discrete change in frequency to represent different symbols. For example, as seen in Figure 2.1, one frequency represents a symbol (0) and another frequency represents another symbol (1). Existing research on ultrasonic channels has primarily used FSK as the modulation scheme [4, 50, 11].

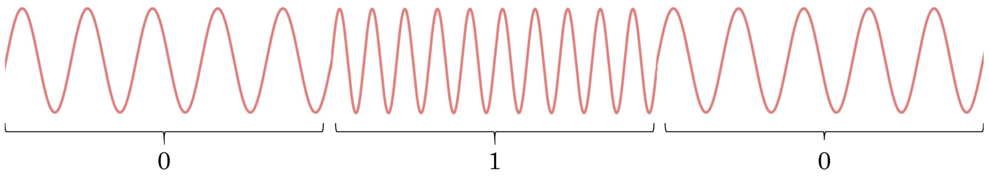

Figure 2.1 FSK modulation example

2.4.1.2 Phase-Shift Keying (PSK)

PSK uses a discrete change in phase to represent different symbols. For example, as seen in Figure 2.2, the normal phase represents a symbol (0) while a phase shifted by π represents another symbol (1).

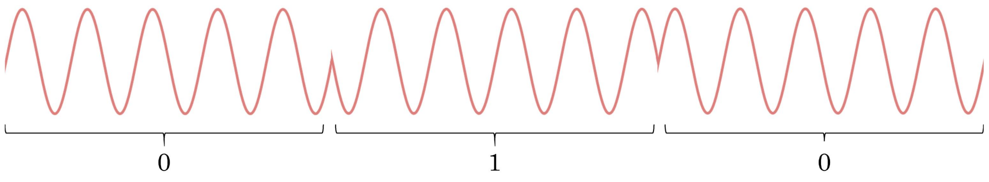

Figure 2.2 PSK modulation example

2.4.1.3 Amplitude-Shift Keying (ASK)

ASK uses a discrete change in amplitude to represent different symbols. For example, as seen in Figure 2.3, the absence of a waveform represents a symbol (0) while its presence represents another symbol (1). This is the simplest form of ASK known as on-off keying (OOK).

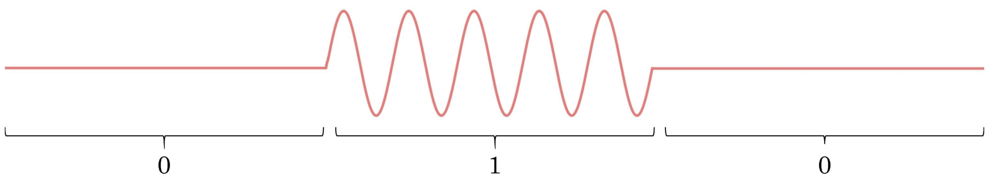

Figure 2.3 ASK modulation example

2.4.2 Signal Processing

In order to demodulate these waves, signal processing techniques are applied.

2.4.2.1 Fourier Transform

In 1822, Joseph Fourier described how the particular value of a function can be expressed by means of a definite integral containing its general value [51]. The Fourier transform converts signals in the time-domain to the frequency-domain, showing the presence of frequencies in the signal. The discrete Fourier transform (DFT) can be used on recorded samples to detect the presence of frequencies [52]. DFT allows for the frequency detection necessary in FSK. The fast Fourier transform (FFT) is a more efficient algorithm for computing the DFT. The Fourier transform was used in other ultrasonic channel research [4, 11].

2.4.2.2 Goertzel Algorithm

In 1958, Gerald Goertzel described a time-domain solution where the magnitude of a spectral line is calculated, equivalent to a complete Fourier analysis [53]. Similar to the Fourier transform, the Goertzel algorithm yields the magnitude of a frequency. The Goertzel algorithm is computationally more efficient than the FFT to detect frequency components for a small number of frequencies [52]. The Goertzel algorithm reduces processing time with fewer computational steps and requires little memory space as it uses few constants, enabling it to even be used on embedded systems [54]. Frequency detection using the Goertzel algorithm is less processing intensive than the FFT, but many engineers are unaware of it [52]. The Goertzel algorithm provides the magnitude of a target frequency. It also yields phase information. An optimized form of the Goertzel algorithm provides the frequency magnitude directly at the expense of phase information [52].

2.4.2.3 Cross-Correlation

Cross-correlation is a measure of similarity between two waveforms. Multiplying one signal by another shows the overlap of their waveforms. A greater value in cross-correlation means the signals are more similar to each other.

2.4.3 Channel Access Methods

A channel access method is the protocol by which systems use the medium to transmit and receive data. Channel access methods can also be considered medium access control (MAC) protocols. The following subsections list the channel access methods relevant to our ultrasonic channel.

2.4.3.1 Automatic Repeat Request (ARQ)

The ARQ protocol uses an acknowledgement (ACK) to confirm that data has been successfully received. An error-detecting code, such as cyclic redundancy check (CRC), enables data to be verified. Data is repeatedly resent unless an ACK is received. Variants of the ARQ protocol have been developed based on this scheme.

2.4.3.2 Random Access

The random access protocol allows any device to transmit at will. The communications are not ordered. If transmissions occur at the same time, there is a *collision*, a simultaneous interference of transmissions. Randomized delays can be used to recover from a collision, allowing one device to transmit before the other.

2.4.3.3 Master/Slave

The master/slave protocol provides ordered communications. The master initiates and controls communications with the slave.

2.4.3.4 Token Passing

The token passing protocol uses a token to order the communications. When a system has the token, it can transmit. The token is then passed to the next system. The network may be arranged in a ring formation.

2.4.4 Network Sockets

Network sockets allow software to communicate easily with other software in a standardized manner. Sockets were used to connect our software to existing computer communication tools using common computer network standards. Sockets use Transmission Control Protocol/Internet Protocol (TCP/IP) to make connections, requiring an IP address and port number.

2.5 Summary

The literature review covers the context of study, giving insight into air gap security and why ultrasonic channels are of interest. Air gaps are critical in high-security networks; however, there are still means to breach air-gapped networks. Ultrasonic channels are a potential method for exfiltrating data from air-gapped networks that needs more research. Existing ultrasonic communications research has been discussed. The telecommunications theory and techniques applied in our research were described.

3 Design

We have described air gaps and the context in which ultrasonic covert channels may be used. Through the development of such a channel, we assess the feasibility and performance of the ultrasonic medium. The channel serves as a demonstration of a covert acoustic channel between computing devices, and also acts a tool for the study of characteristics of similar channels. The channel was developed in three layers: modulation/demodulation of ultrasonic signals, implementation of a channel access method, and software interfacing with the channel. This chapter describes the design and development of the ultrasonic channel.

3.1 Proof of Concept

Our software demonstrates the possibility of ultrasonic channels between speakers and microphones. As seen in Figure 3.1, the software on the air-gapped system transfers information ultrasonically to the exfiltration device. Likewise, commands are sent ultrasonically from the exfiltration device to the air-gapped system in a two-way communications channel.

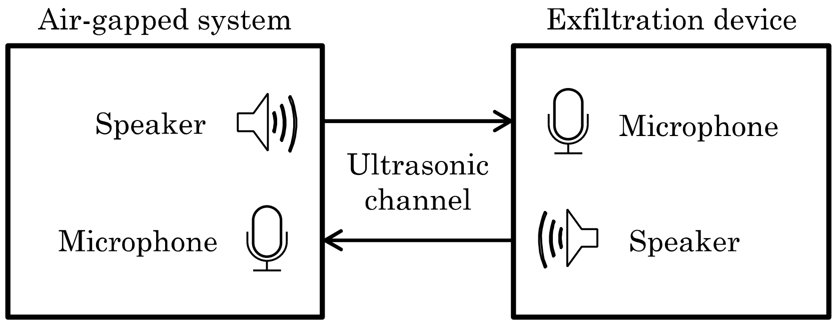

Figure 3.1 Ultrasonic channel setup

A one-way communications channel from the air-gapped system to the attacker would only allow for exfiltration of data from the secure network. However, a two-way communications channel between the air-gapped system and the attacker allows for exfiltration of data as well as interactive access to the air-gapped system. The two-way channel was required because the operational scenario includes an interactive ultrasonic command-and-control channel. Given that the data to be exfiltrated from an air-gapped system would be of high value, preserving its integrity is desirable. The two-way channel allows for error detection and correction, ensuring the fully accurate transfer of information. Common high-level network protocols such as TCP require two-way communications. The software can be used in the context of bridging an air gap, and also for general-purpose information transfer.

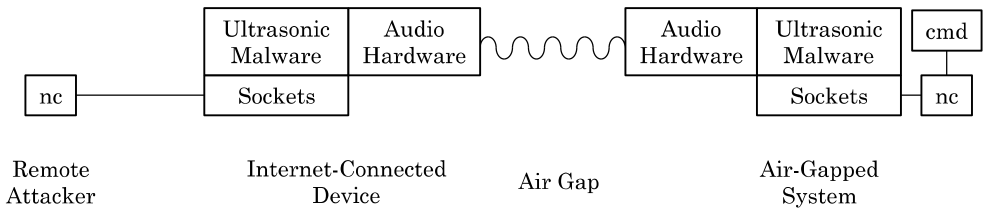

Figure 3.2 Deconstructed software model

By developing the proof-of-concept software, we identify the vulnerability that is ultrasonic channels in air-gapped networks. Figure 3.2 shows the Internet-connected exfiltration device communicating with the air-gapped system ultrasonically to relay commands from a remote attacker. Our software enables a remote attacker to use a sockets program such as Netcat (nc) to connect to an exfiltration device within acoustic proximity of the air-gapped network. In concept, this exfiltration device may be a compromised Internet-connected mobile device (e.g., smartphone) with a speaker and microphone. The malware communicates ultrasonically with the compromised air-gapped system. On the air-gapped system, a command-line interpreter (e.g., cmd/ Command Prompt) is attached, allowing command-line access to the air-gapped system.

3.2 Protocol Stack

The channel was organized into three computer network communications protocol layers following the layers of the Open Systems Interconnection (OSI) model.

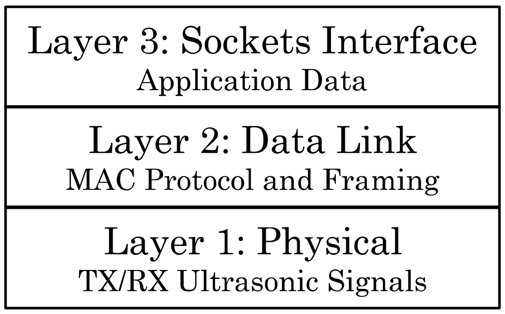

Figure 3.3 Protocol stack model

As shown in Figure 3.3, our protocol stack uses a physical layer, data link layer, and sockets interface layer.

- Sockets interface layer: forwards application-level data to/from external programs via network sockets.
- Data link layer: frame structure and channel access method for ultrasonic communications.
- Physical layer: low-level modulation, demodulation, and synchronization of ultrasonic signals; transmit (TX) and receive (RX) functionality.

The channel was developed with a layered approach. Development began with low-level functionality and proceeded through to the high-level. The following subsections proceed through the development of each layer. Our methods are described and examples of code have been provided.

3.2.1 Layer 1: Physical

The physical layer acts as a modem for ultrasonic signals. The physical layer contains our low-level acoustic relay methods, that is, the methods for modulation, demodulation, and synchronization of acoustic waves. The following subsections detail the functionality of our physical layer, including a preliminary analysis of digital modulation schemes for ultrasonic communications.

3.2.1.1 Acoustic Modulation

Initially, audio files (.wav and .mp3) containing ultrasonic frequencies were generated to roughly assess the ultrasonic transmission capability of speakers and ultrasonic reception capability of microphones. The speakers and microphones on several mobile devices were tested to transmit and receive ultrasonic signals.

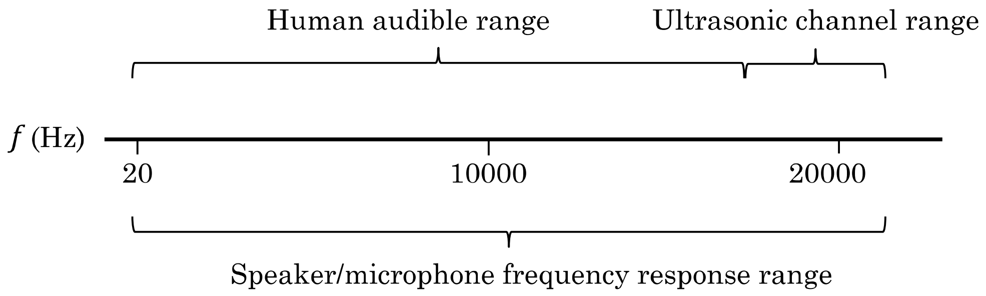

Figure 3.4 Usable ultrasonic frequency range

The ultrasonic channel needed to function above the human audible range but within the frequency response of ordinary speakers and microphones. Figure 3.4 illustrates the limited ultrasonic channel bandwidth to operate within. Since the human audible range extends up to 17 900 Hz [43], and the frequency response of audio hardware goes up to ~20 000 Hz, the ultrasonic channel is constrained between these frequencies. We found a usable bandwidth from approximately 18 000 Hz to 20 000 Hz. A more exact usable range is discussed in the validation activities.

In our channel software, waveforms were generated using a sine wave formula and plotted as a list of amplitudes in memory. The sampling rate determines the size of the list of amplitudes. The list of amplitudes directly correlates to the physical movement of the speaker cone. Ultrasonic signals are effectively played as normal sound from speakers.

$$f_n = \frac{f}{f_s}$$

$$\omega = 2\pi f_n$$

$$y = \sin(\omega x)$$

where
- f_n = normalized frequency (cycles/sample)
- f = channel frequency (Hz)
- f_s = sampling rate (Hz)
- ω = normalized angular frequency (radians/sample)
- y = amplitude (from -1 to 1)
- x = sample number

Equation 3.1 Waveform generation

In Equation 3.1, the normalized frequency is calculated by dividing the channel frequency by the sampling rate. The normalized frequency shows how many wave cycles pass per sample. The normalized angular frequency converts this to radians per sample. The waveform amplitude for each sample is plotted with the normalized angular frequency multiplier to generate the desired channel frequency.

3.2.1.2 Goertzel Algorithm Demodulation

The microphone records in order to receive ultrasonic signals. Sound is recorded at the sampling rate, the rate at which wave amplitude points are plotted. Amplitudes are recorded as a floating point number from -1 to 1. The list of wave amplitudes is processed with the Goertzel algorithm.

The DFT is processing intensive. Similar to other research [4], we had first tried using FFT. However, an inefficient algorithm would restrict the channel speed. A faster method is more practical, especially if it is to be applied to mobile or embedded devices, which have reduced processing speeds. The Goertzel algorithm was an efficient method for ultrasonic demodulation. The full Goertzel algorithm was used to detect the target frequency magnitude and phase of the recorded waveform. The optimized Goertzel algorithm was used for FSK, where phase information was unnecessary. The frequency magnitude is a relative indicator of volume at the specified frequency.

The Goertzel algorithm determines the overall frequency magnitude (and phase) of the recorded waveform over a period of time. The presence of a frequency above a magnitude threshold signifies the beginning of an ultrasonic message. The software continues to record the subsequent ultrasonic signals, which are then run through the Goertzel algorithm. The waveforms recorded are demodulated into binary according to frequency magnitude (or phase).

```
def goertzel_coefficient(frequency):
   k = int(0.5 + pulse_length*frequency/sampling_rate)
   w = 2*pi/pulse_length*k
   cosine = cos(w)
   sine = sin(w)             #only used for PSK
   coeff = 2*cosine
   return coeff
```

Listing 3.1 Pseudocode for Goertzel algorithm coefficient

The Goertzel algorithm uses a pre-calculated coefficient, shown in Listing 3.1, based on the target frequency (e.g., 19 000 Hz), sampling rate (e.g., 44 100 Hz), and pulse length (e.g., 180 samples). The pulse length is the number of samples it takes to convey one symbol. The sampling rate and pulse length are kept constant in an instance of the channel.

```
def goertzel_algorithm(amplitudes, coeff):
   Q1 = 0
   Q2 = 0
   for sample in range(pulse_length):
      Q0 = coeff * Q1 - Q2 + amplitudes[sample]
      Q2 = Q1
      Q1 = Q0
   magnitude = sqrt(Q1**2 + Q2**2 - Q1*Q2*coeff)
   return magnitude
```

Listing 3.2 Pseudocode for optimized Goertzel algorithm

As shown in Listing 3.2, the Goertzel algorithm uses several temporary variables (Q0, Q1, and Q2) to iterate over the recorded signal to yield the magnitude of the target frequency. The Goertzel algorithm acts as a digital filter to speed up the evaluation of a finite trigonometric series for individual terms in the DFT [55, 56]. The Goertzel algorithm can be seen as a series of second-order infinite impulse response (IIR) filters [54]. Samples are processed over a set time interval recursively using its previous two outputs, the temporary variables Q1 and Q2. After a few samples have been read, the Goertzel algorithm converges on the target frequency information [57].

3.2.1.3 Cross-Correlation Synchronization

Ultrasonic signals were initially received without cross-correlation, using a window time length that matches the length of a pulse. Reading was not accurate and errors in reading were sometimes present. Without cross-correlation the entire transmission can be misread due to the position of the ultrasonic signals in the recording. For example, the beginning portion of an ultrasonic signal may be detected at the end of a recording, causing the first bit to be read in twice and the rest of the frame shifted by one bit. Not knowing exactly where the signal begins results in misreading symbols due to reading in between the symbols. Imprecise reading distorts the demodulated phase and frequency information.

This gave rise to the need to track modulated pulses when they are received. This tracking or synchronization was achieved with cross-correlation, which provided a more specific position in the recording over which to run the Goertzel algorithm, making demodulation much more precise. Cross-correlation eliminates errors in reading.

Cross-correlation was used to find an offset where the ultrasonic signals begin in a recording. This greatly improved the accuracy of demodulation. The offset allows us to run the Goertzel algorithm directly over the location of the signals in the recording.

```
def cross_correlation(amplitudes, reference):
   optimal_offset = 0
   max_correlation = 0.0
   for offset in range(pulse_length):
      correlation = 0.0
      for sample in range(pulse_length):
         correlation += amplitudes[offset+sample] *
                        reference[sample]
      correlation = abs(correlation)
      if correlation > max_correlation:
         optimal_offset = offset
         max_correlation = correlation
   return optimal_offset
```

Listing 3.3 Pseudocode for cross-correlation

As seen in Listing 3.3, we iterate over the recording at each offset, finding a correlation of similarity to the reference signal. Cross-correlation is the sum of the received and reference signal multiplied. The maximum correlation is the best match of similarity, yielding the offset where the signal begins in the recording.

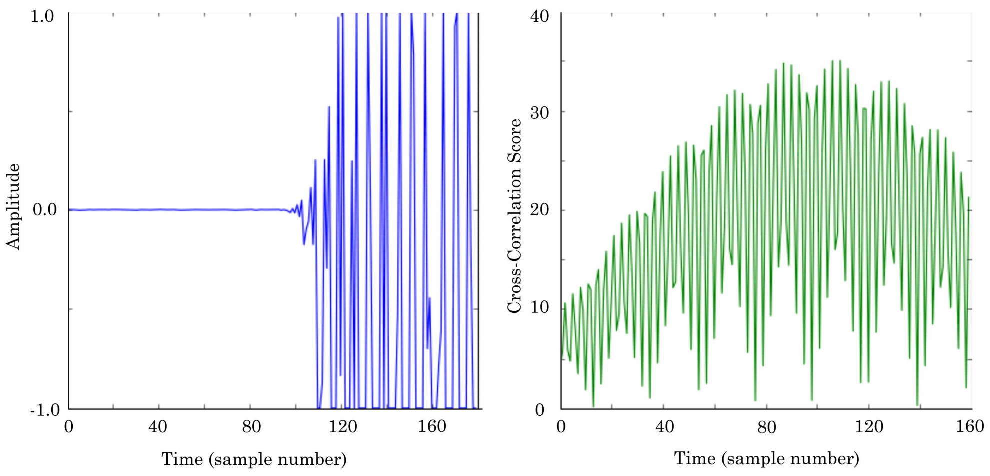

Figure 3.5 Signal recorded (left); Cross-correlation scoring (right)

In Figure 3.5, the maximum cross-correlation occurs at about 105 samples (seen on the right). The offset is located at 105 samples, which is where the signal visibly begins (seen on the left). The precision provided by cross-correlation allows for accurate demodulation, especially necessary for shorter pulse lengths.

3.2.1.4 Pulse Shaping

A pulse shape improves cross-correlation by making it easier to recognize the position of a symbol in the recording. It physically delimits each symbol with an overall recognizable shape, allowing the receiver to match this overall shape instead of only its contents. The pulse shape ameliorates cross-correlation as the overall pulse shape yields more accurate cross-correlation scores compared to matching only the low-level frequency contents. Pulse shaping increased the accuracy of finding the offset where the signal starts.

Initially, the amplitude was linearly increased and decreased, creating diamond shapes. However, a rounder, fuller pulse shape contains more amplitude presence for better demodulation. The Gaussian distribution can also be applied, effectively making Gaussian FSK (GFSK). For simplicity and practicality, we decided to use a cosine multiplier in order to create the pulse shape.

$$y = \sin(\omega x) \cdot \frac{(1 - \cos(\frac{2\pi}{l}x))}{2}$$

where
y = amplitude (from -1 to 1)
x = sample number
ω = normalized angular frequency (radians/sample)
l = pulse length (samples)

Equation 3.2 Waveform generation with pulse shaping

In Equation 3.2, the original waveform equation from Equation 3.1 was multiplied with a pulse shape that appropriately positions and scales the pulse.

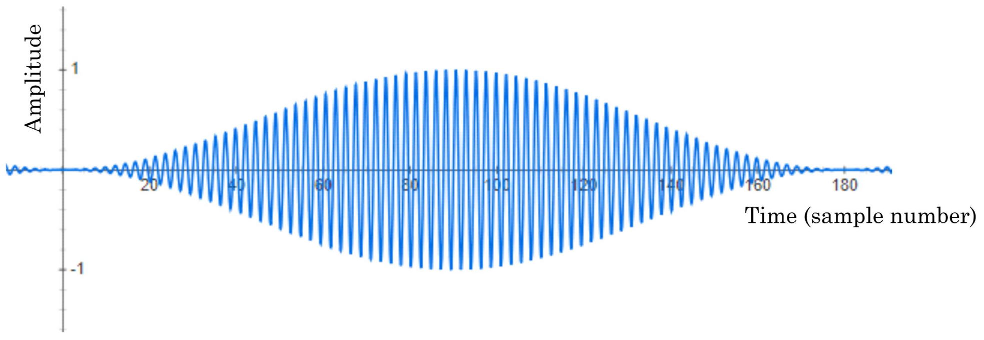

Figure 3.6 Pulse-shaped waveform plotted

Figure 3.6 is a graphical representation of the equation used for pulse shaping from Equation 3.2. Pulse shaping solved the audible clicking noise issue mentioned in other research [4]. Without pulse shaping, the jump from one symbol to another causes a distinct audible click. Since the pulse shape reduces the ends of a symbol in amplitude, the speaker can transition to the next symbol without any audible clicking noise. Elimination of the clicking noise is crucial for covert ultrasonic transmissions.

3.2.1.5 Preamble Pulse

There was still slight inaccuracy after pulse shaping in detecting precisely where the signal starts. This was due to matching the reference pulse at one frequency, while the first bit could have been either of the channel frequencies. Cross-correlation over a longer pulse (e.g., two pulse lengths) did not have much gain in accuracy. To increase the accuracy of cross-correlation, a preamble pulse was set to a designated frequency to be transmitted before the data pulses. The preamble pulse is more distinctly recognizable in cross-correlation, providing a more accurate offset from which to start reading the rest of the message.

The preamble frequency can be set to one of the channel frequencies. However, cross-correlation with a completely different frequency gave a slightly more accurate offset. As such, the preamble pulse uses a frequency that is different from the two channel frequencies. The preamble frequency can be set to be between the data channel frequencies to conserve bandwidth. Lower frequencies were more reliable in demodulation, especially at further distances. Accordingly, the preamble frequency may be set to a lower frequency for increased accuracy.

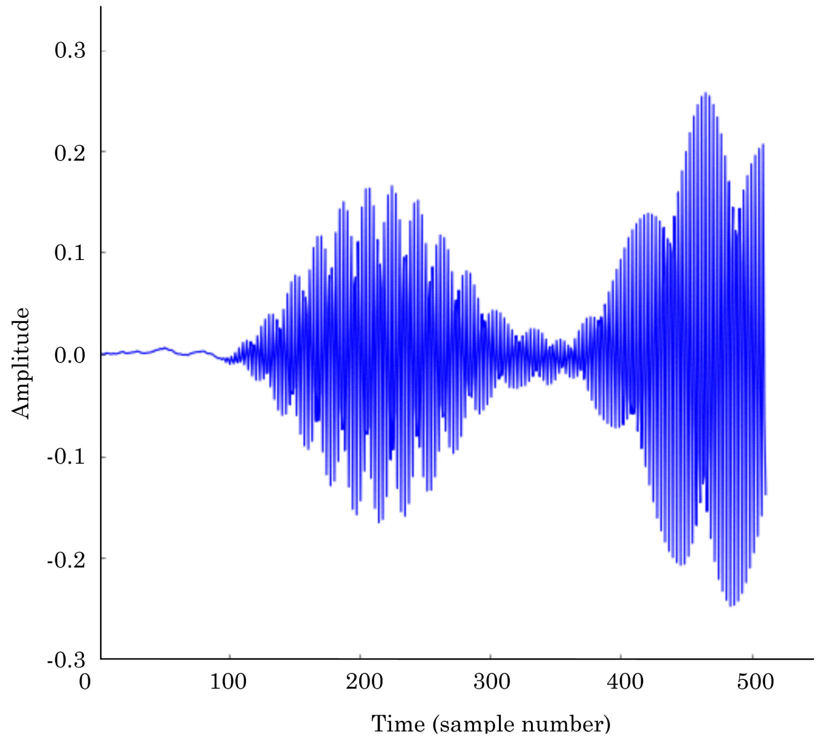

Figure 3.7 Preamble pulse

The preamble pulse can also be called a starter pulse; it signifies where a transmission begins. In Figure 3.7, the preamble pulse can be seen between 100 and 300 samples. The data pulses follow after the preamble pulse. The preamble pulse triggers the recording of subsequent signals (over a specified length of time) to be parsed. With pulse shaping and a preamble pulse, cross-correlation finds an exact offset for demodulating the symbols in the transmission.

3.2.1.6 Preliminary Analysis of Digital Modulation Schemes

After proving functionality at the physical layer, we explored alternate possibilities for optimality in the different modulation schemes. The modulation scheme determines how the original waveform is shifted to convey discrete symbols. Each of the digital modulation schemes was applied to shift the frequency, phase, and amplitude.

Quadrature shift keying has twice the bitrate of binary shift keying by conveying two bits in 1 symbol (i.e., 00, 01, 10, or 11). The following subsections report our preliminary findings on digital modulation schemes for ultrasonic transfer.

3.2.1.6.1 Binary Frequency-Shift Keying (BFSK) and Quadrature Frequency-Shift Keying (QFSK)

FSK was the primary modulation scheme used in the development of the ultrasonic channel. In BFSK, a pulse plotted at a certain frequency designates 0, while a pulse at a different frequency designates 1. These frequencies were separated by a spectral difference sufficient to discriminate between the frequencies in demodulation. The normalized angular velocity, ω (see Equation 3.1), was recalculated to generate the different frequencies.

From brief testing, about 500 Hz of separation can be used for FSK. A more exact measure of frequency discrimination is described in the validation activities. The 500 Hz of separation can be implemented around 19 000 Hz. Development began with audible frequencies. FSK tested to be functional up to around 20 000 Hz. A more precise upper frequency response is discussed in the validation activities.

QFSK uses two more frequencies. QFSK with four frequencies easily follows if it can fit within ultrasonic frequency bandwidth. We noted that the four frequency channels in QFSK can fit in ultrasonic ranges that most people cannot hear. At 500 Hz of frequency separation, four frequency channels would require 1 500 Hz of bandwidth (i.e., 500 Hz × (4-1)). 1 500 Hz bandwidth fits between 18 500 Hz and 20 000 Hz.

3.2.1.6.2 Binary Phase-Shift Keying (BPSK) and Quadrature Phase-Shift Keying (QPSK)

After confirming that our demodulation methods were functional with FSK, we experimented with PSK as the modulation scheme. While the optimized Goertzel algorithm allowed for faster processing of FSK at the expense of phase information, the full Goertzel algorithm is required for PSK. Therefore, PSK is slightly more computationally intensive than FSK. A phase shift of theta (θ) is added to the original wave equation (i.e., $\sin(\omega x + \theta)$) to represent different symbols.

```
real = Q1 - Q2*cosine
imaginary = Q2*sine
theta = degrees(atan2(imaginary,real))
```

Listing 3.4 Pseudocode for phase in Goertzel algorithm

Listing 3.4 shows the calculations that are added to the end of the Goertzel algorithm (see Listing 3.2) to return phase instead of magnitude. The full Goertzel algorithm yields the real and imaginary components of the magnitude. The previous cosine and sine variables calculated in the Goertzel coefficient are required (see Listing 3.1). The real and imaginary components are used to find the phase shift, θ, of the waveform.

In BPSK, 360° is halved such that θ from 0° to 180° represents one symbol (0) and θ from 180° to 360° represents the other symbol (1). For QPSK, there are four different phase states, where 360° is quartered instead of halved. Similar to QFSK, having four symbols allows the representation of two bits per symbol, therefore doubling the bitrate.

With PSK there was an issue of a *phase drift*. An extraneous addition to the expected value of θ was observed, interfering with correct demodulation. The phase drift was a continual θ shift of about 15° between pulses. Since the phase drift makes a reference phase unusable, we had to use differential PSK (DPSK) instead of regular PSK. DPSK compares the current phase to the previous phase instead of using a reference phase. For the initial phase, the phase of the preamble pulse can be used. Differential BPSK (DBPSK) was successfully implemented and functional around 20 000 Hz. Differential QPSK (DQPSK) is also functional, although it is not as precise as DBPSK due to the reduced symbol space. Quadrature modulation schemes have a higher bit error rate (BER) than binary modulation schemes. The phase drift was noticeably less for lower frequencies (less than 15°) but the issue persists.

The advantage of phase-shift keying is that phase information can be transferred on the same frequency without occupying more spectral bandwidth within the ultrasonic range. The other frequency, instead of being used to signal the other symbol, can be used for an extra phase-shift keyed channel. This may allow full-duplex communications.

A problem with PSK may be sound reflection (i.e. echoes). In a real world application, phase information is the most likely to be distorted. Acoustic reverberations over distance can cause phase to become highly distorted. Issues in propagation, loss, and reflection especially affect phase information. These are more of an issue in acoustic waves than electromagnetic waves. PSK was more prone to errors than FSK. As such, the channel was developed using BFSK for its reliability.

3.2.1.6.3 Binary Amplitude-Shift Keying (BASK) and Quadrature Amplitude-Shift Keying (QASK)

An obvious form of modulation is ASK. The simplest form of ASK is the presence vs. absence of pulse, called OOK, which is the form of BASK we considered. A pulse represents 1, while absence represents 0. Demodulation with the Goertzel algorithm calculates whether the received magnitude of a certain frequency is above or below a threshold. For QASK, an amplitude multiplier is used (i.e., $\mathrm{A} \cdot \sin(\omega x)$) to constrain the maximum wave amplitudes.

An advantage of BASK/QASK is that a period of silence acts as a symbol itself, instead of having to transmit at another frequency. This saves from having to use another frequency if the spectrum bandwidth were more restricted because of audio hardware limitations. At the minimum, one frequency within the ultrasonic range should be usable for a BASK modulation scheme, which has the same bitrate as with two frequencies in BFSK. Alternatively, the absence of a frequency above a magnitude threshold can be used to end the transmission.

ASK was not tested in our channel for several reasons. In BASK, if the offset is not exactly accurate, it is likely to read a small amount of frequency presence where there is supposed to be silence, resulting in error. Implementation of QASK needs to have multiple magnitude thresholds. There would be three levels of amplitude and silence to convey the four symbols. In development, we observed that magnitude was a highly variable number, dependent on distance and hardware. Reducing the amplitude for QASK also reduces the distance of transmission because quieter sound cannot travel as far. Compared to electromagnetic waves, acoustic waves attenuate significantly more over a shorter distance.

3.2.1.6.4 M-ary Shift Keying

Beyond binary and quadrature schemes, digital modulation can have M-ary shift keying. The continuation follows from binary and quadrature schemes by further dividing the modulation space, extending to 8 symbols, 16 symbols, and so on. Using 8 symbols means that 4 bits can be transferred per pulse, thereby increasing the binary scheme's bitrate by four times. Using 16 symbols means that 8 bits can be transferred per pulse, thereby increasing the binary scheme's bitrate by eight times.

In M-ary FSK, more frequency channels would be used. For an ultrasonic channel between speakers and microphones, the number of frequencies usable is determined by the ultrasonic frequency range and how closely frequency channels can be placed. 8 frequency channels do not easily fit in the ultrasonic range between 18 000 Hz and 20 000 Hz. The 4 frequency channels in QFSK are the most we could place within the ultrasonic range with 500 Hz of frequency separation. However, if the audible range of frequencies could be used, we could certainly have 8, 16, or even 32 frequency channels. Such schemes can be called FSK8, FSK16, and FSK32 respectively. At 500 Hz of frequency separation, 32 frequency channels require 15 500 Hz of bandwidth (i.e., 500 Hz × (32-1)). This would fit between 20 Hz and 20 000 Hz. Using 32 symbols would mean that 16 bits can be transferred per pulse, thereby increasing the binary scheme's bitrate by 16 times. Although entirely feasible, it was not necessary to demonstrate this; our software was validated with BFSK.

In M-ary PSK, more phase shifts would be used. Since QPSK was already not very accurate, PSK8 would have too many errors. In M-ary ASK, more amplitudes would be used. Since QASK was already not optimal due to the previously mentioned issues, ASK8 would have been even more problematic.

3.2.1.6.5 Combined Digital Modulation Schemes

Frequency, phase, and amplitude are independent of one another. Because these wave properties can carry information independently, it is possible to combine digital modulation schemes to send more data simultaneously. For example, PSK applied at different frequencies may be called frequency-PSK (FPSK). We confirmed that FPSK is possible. However, it is prone to error due to the phase drift issue mentioned in experimenting with PSK. FSK was combined with PSK, but the phase drift persisted. In FPSK, there were sequences that could not be transferred. For example, multiple of the same symbol in a transmission would lead to a perceived phase shift when there is none (e.g., $f_0\theta_0$, f_1, f_1, $f_0\theta_0$). Phase was compared to the previous phase of the same frequency. As such, FPSK was not implemented for the channel. Certain sequences could not be received if the phase drift exists.

Ideally, an implementation of QPSK over M-ary FSK channels would transfer the most information simultaneously.

Another way of combining modulation schemes is the following. In ASK, a period of silence can be used to represent a symbol. Taking this concept from ASK, it is possible to implement QFSK with only three frequencies and a silence, thus conserving spectrum bandwidth by using one less frequency channel.

3.2.1.6.6 Summary of Digital Modulation Schemes

Table 3.1 summarizes the theoretical bitrate and expected performance of the different digital modulation schemes that were investigated in our preliminary analysis.

Modulation Scheme	**Theoretical Bitrate (bit/s)**	**Expected Performance**
BFSK	250	Reliable
QFSK	500	Reliable, fits in ultrasonic bandwidth
M-ary FSK	1000+	Reliable, requires audible bandwidth
PSK	N/A	Not functional due to phase drift issue, reference phase unusable
DBPSK	250	Functional
DQPSK	500	Functional, more error prone than DBPSK
M-ary DPSK	N/A	Not functional due to reduced symbol space
BASK	250	Functional
QASK	N/A	Not functional due to variable magnitudes, reduced transfer distance
M-ary ASK	N/A	Not functional
FPSK	500	Possible but prone to error

Table 3.1 Summary of digital modulation schemes

3.2.2 Layer 2: Data Link

We returned to our most reliable form of modulation, BFSK, to develop the remainder our channel. The physical layer had transmit/receive functions that were usable by the data link layer. We developed a channel access method suitable for two-way half-duplex communications between the air-gapped system and the exfiltrating device.

The channel access method uses a frame to transfer data. This frame contains overhead fields that ensure data is transferred accurately in the protocol. This frame is effectively the sequence of bits pulsed in a transmission. At the receiver, the bits are received and interpreted according to the frame structure. The preamble pulse precedes the frame.

This section describes the frame structure (see Figure 3.8) for communications, followed by a preliminary analysis of communications protocols for ultrasonic transmission.

3.2.2.1 Frame Structure

ACK	Duplicate	Port	Length	Payload	CRC
1 bit	1 bit	16 bits	6 bits	0-512 bits	16 bits

Figure 3.8 Frame diagram

3.2.2.1.1 Acknowledgement

The acknowledgement is a bit that acknowledges the frame transmitted was successfully received. The ACK (value of 1) tells the sender that the previous frame sent was received. Otherwise, the previous frame is continually resent until the sender receives acknowledgement. A negative acknowledgement (NAK; value of 0) tells the sender that the previous frame was not received and must be resent.

Instead of using a serialized acknowledgement number to keep track of frames, the acknowledgement here is a simplification of this concept and reduces the frame length required by a serialized number. This was an efficiency introduced given the slow bitrate of the channel.

3.2.2.1.2 Duplicate Identifier

The duplicate identifier is a bit that identifies whether the frame is new or a retransmission of the previous frame. This was part of ensuring reliable transfer. Transmissions may be duplicated if there is an error causing the sender to retransmit. If the same data is received a second time, it must be discarded. Additionally, sometimes the payload data may be exactly same as the previous frame. Even if the payload data sent is exactly the same, the duplicate identifier tells the receiver that the frame is new information meant to be received. This differentiates between new data and a retransmission of the previous frame.

Similar to the acknowledgement, instead of using a serialized number that keeps track of frames, the duplicate identifier is a simplification of this concept and reduces the frame length required by a serialized number. This was an efficiency introduced given the slow bitrate of the channel.

3.2.2.1.3 Port Number

The port number is the TCP port that data is meant to be forwarded to. The corresponding application is connected to this port, ready to accept the data that has been transferred ultrasonically.

3.2.2.1.4 Payload Length

The payload length determines the length of a frame. We used a payload length field to facilitate experimentation with different frame sizes demonstrated later in the validation activities. This field is not necessary when frame length is set and consistent. The payload length field should be entirely omitted to reduce frame length if a consistent payload length is used.

3.2.2.1.5 Payload

The payload is the actual data to be transferred ultrasonically. This is the application-level data in binary. The payload length can be anywhere from 0 to 512 bits. A reasonable upper bound for the payload length is 512 bits because greater lengths may be prone to error in practical application. Longer transmission lengths have a higher chance of a single bit being incorrect in the entire frame.

3.2.2.1.6 Cyclic Redundancy Check (CRC)

CRC is the error-detecting code used to verify the integrity of data transmitted. CRC provides error detection so that the data link layer can correct for errors with retransmission. The CRC field is matched to the CRC computed on the contents of the frame. A match means that the frame is correct, and a mismatch means that there was some error. If the CRC doesn't pass, the entire frame is discarded since the data is flawed.

In choosing the bit length of the CRC, there were 8, 16, and 32 bit options. CRC8 has 256 hash values, while CRC16 has 65 536 hash values, and CRC32 has about 4 billion hash values. CRC16 (crc-16-dds-110) was an appropriate choice given our payload length of up to 512 bits, saving the excessive frame length of CRC32. Although we used CRC16, CRC8 can suffice to decrease the frame overhead with a low chance of inaccuracy.

3.2.2.2 Preliminary Analysis of Channel Access Methods

We have previously described the structure of our frame. The frame was used in our channel access method. Our channel access method used an ARQ methodology to correct for errors in transmission. This ensured the reliable transfer of information. In ARQ, transfers proceed linearly, where the previous frame must be successfully received in order for the next to be accepted. The next frame is not sent until an acknowledgement is received. The protocol ensures that 100% of the data is transferred correctly, guaranteeing the accurate transfer of data. The following figures (Figure 3.9 to Figure 3.11) illustrate the ARQ protocol using devices *Alice* and *Bob*.

$$\begin{tikzcd}
\textbf{Alice} & \textbf{Bob} \\
\text{A1} \arrow[r] & \text{CRC} \checkmark \\
\text{CRC} \checkmark & \text{B1, ACK} \arrow[l] \\
\text{A2, ACK} \arrow[r] & \text{CRC} \checkmark \\
\text{CRC} \checkmark & \text{B2, ACK} \arrow[l]
\end{tikzcd}$$

Figure 3.9 ARQ protocol (normal)

In Figure 3.9, the data payload is represented by the first letter of the sender, followed by a sequence number (e.g., Alice sends A1 data, then A2 data). Checkmarks represent successful CRC checks done by the receiver. This figure shows Alice and Bob communicating normally without error. Alice sends her A1 data. It is successfully received by Bob, who replies with his B1 data and an acknowledgement, meaning that A1 has been received. Alice receives acknowledgement and sends A2 along with an acknowledgement for B1. Bob receives acknowledgement and sends B2 along with an acknowledgement for A2, and so on.

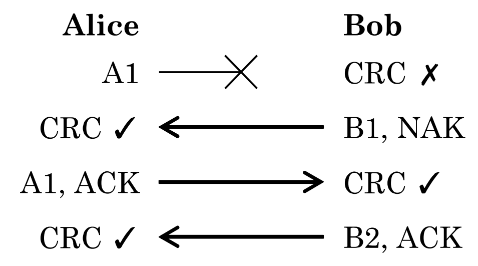

Figure 3.10 ARQ protocol (error)

In Figure 3.10, an error has occurred in Alice's transmission of A1 data, causing Bob's CRC check to fail. Bob replies with his B1 data, along with a negative acknowledgement because he did not receive the data correctly from Alice. Alice resends her A1 data along with acknowledgement of Bob's B1 data.

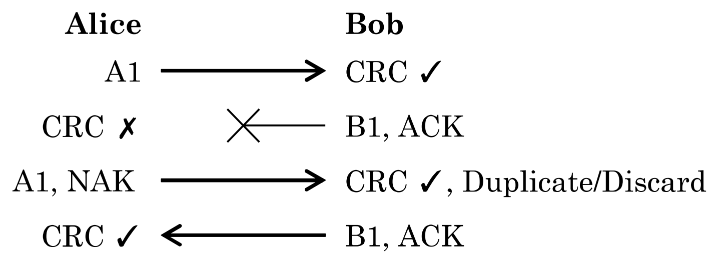

Figure 3.11 ARQ protocol (duplicate/discard)

In Figure 3.11, Alice successfully sends A1 data to Bob. However, Bob's reply fails, causing Alice to retransmit A1. Since Bob has already received A1, he detects that it is a duplicate and discards it.

In addition to the ARQ protocol, we used another means to model the order of communications in our channel access method. Originally, a random access protocol was implemented. A random access protocol was possible due to the capability of detecting starter ultrasonic frequencies. This is similar to carrier-sense multiple access with collision detection (CSMA/CD), which is able to detect collisions and uses random waiting times to recover from collisions. When a starter frequency is detected, the remainder of the frame is then recorded. Any device can initiate transmission. A machine address field can be added without the ordering of communications. A solution to collisions in the random access protocol is to use randomized waiting times before retransmission. In Figure 3.12, the devices transmit simultaneously, resulting in a collision. They are set to receive for a random amount of time before transmitting again, at which point one will likely transmit before the other.

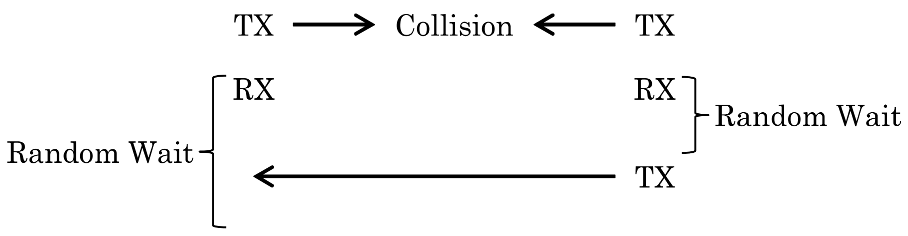

Figure 3.12 Random access protocol collision recovery

Even more applicable than a random access protocol, a master/slave protocol was implemented in order to suit the operational scenario. In the master/slave protocol, only the Internet-connected exfiltration device initiates communications with the air-gapped system, which also avoids transmission collisions. The order of the master/slave protocol avoids errors where multiple hosts attempt to transmit simultaneously. The random access protocol was more like a "free-for-all" where there were collisions. Given the potential inefficiencies with heavy loads and long waiting times, the random access protocol was not as extensible to a network because all of the devices would be trying to transmit at the same time, jamming each other, especially since transmissions are slow. The channel becomes inefficient when considering scalability up to a network level.

A network can be formed through an ordered scheme such as master/slave or token passing. The token passing protocol is similar to master/slave protocol. In token passing, more devices would be used along with a means of addressing. The master/slave protocol with implicit token passing on a one-to-one channel can be extended to token passing for multiple systems. Symmetry was considered in the protocol so that the code for the master is mostly similar for the slave.

A primary reason for using the master/slave protocol is that the air-gapped slave device would not be beaconing out constantly if data was received on the socket. An attacker would want to connect in only when needed and not have a system broadcast out repeatedly, making the covert channel potentially more easily detectable. The master device only beacons out when communication is desired. When the attacker is connected to the master device, it can then beacon out to nearby systems. In this protocol, the ultrasonic channel may be disconnected most of the time. The slave does not beacon out and remains silent, listening for signals. The master connects to the slave when in range.

3.2.3 Layer 3: Sockets Interface

External applications were connected to the ultrasonic channel via TCP/IP sockets. With sockets connections, the channel was effectively “interfaced” with external applications in a generic manner. A simplified overview of our protocol stack implementation appears as local port forwarding over an ultrasonic channel. Application-level data is sent through sockets.

The sockets interface was implemented considering compatibility and extensibility. The sockets network standard is ubiquitous in contemporary computer communications, commonly used as the interface to network protocols. Sockets are useful because they can connect existing, generic sockets-based programs together. Sockets enable the program to be connected to other programs as part of a larger set of attack functionality. Programs connected by sockets can easily pass information between each other, locally and across the Internet. These programs may be developed independently, performing unique functionality, and then connected together to relay information between each other.

3.2.3.1 Generic Sockets Relay

Our software from the outside appears to be a sockets relay over ultrasonic communications. Once external applications are connected with sockets, information can be transferred through the ultrasonic channel. Sockets also enable remote connections (over the Internet) to the exfiltration device.

The program is set to use specified ports. The Internet-connected exfiltration device locally binds to these ports and accepts external application connections. Separately, the air-gapped system repeatedly tries to connect to its copy of the external applications on the specified ports. When an external application has a socket connection to the program, the socket is added to the list of sockets.

Application-level data received on a socket is converted from a bytes string into a binary list, put into a frame, and sent according to our channel access method protocol. At the receiver, the binary payload bits are converted back into a bytes string and forwarded to the appropriate port.

3.2.3.2 Queued Delivery

Given the slow speeds of ultrasonic transfer, queues are useful to hold the application-level data before ultrasonic transfer. Queue servicing threads are run concurrently. The transmit queue (TxQueue) and receive queue (RxQueue) are accessible to the data link layer. These queues have push & pop methods, and provide a separation of concern. Having mutual exclusion, the queues are ensured against thread concurrency errors.

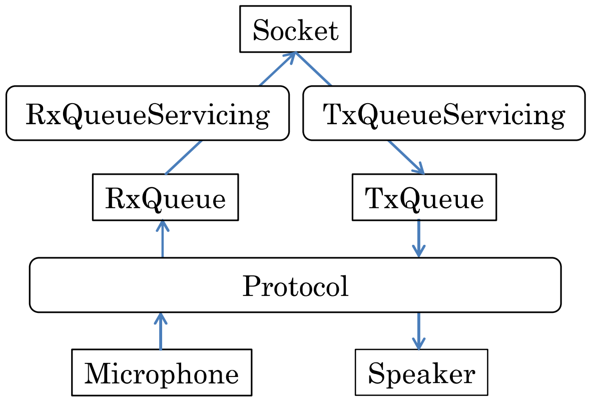

Figure 3.13 Program data flow with socket

As seen in Figure 3.13, data flows from the microphone to the socket, and from the socket to the speaker. A bytes string received from a socket is converted into a binary list and placed into a frame structure. The frame is put into the TxQueue by the TxQueueServicing thread. This frame in the TxQueue is popped off by the channel access method when ready to transmit ultrasonically.

The bits received from a CRC-verified and non-duplicate ultrasonic transmission are placed into a binary list in a frame structure. The frame structure is put into the RxQueue and popped off by the RxQueueServicing thread, which converts the binary list into a bytes string and forwards it to the appropriate socket.

The sockets interface was designed such that each socket has its own TxQueueServicing thread. Meanwhile, only one RxQueueServicing thread forwards data from the RxQueue to all of the connected sockets.

This architecture is faster than a previous architecture where each socket used its own TxQueueServicing and RxQueueServicing threads.

3.2.3.3 Port Mapping External Applications

Multiple external applications can be connected simultaneously to the ultrasonic channel. The program can use multiple ports, enabling multiple sessions. The same port numbers are used on both devices. However, the port numbering may be obfuscated by changing the port numbers. Different port numbers can be used to communicate ultrasonically (i.e., port A on the master is sent to port B on the slave).

3.2.4 Functional overview

Our software was developed in three layers: physical, data link, and sockets interface. The protocol stack mapped onto our layered development process. The following listings display the methods developed for our software.

```
physical
class Audio()
   def tx_sound(tx_frame)
   def rx_sound(seconds) return rx_frame
   def graph_pulse_display()
   def _plot_pulse(frequency) return amplitudes
   def _prepare_pulses()
   def _goertzel_coefficient() return coeff
   def _prepare_goertzel_coefficients()
   def _goertzel_algorithm(amplitudes,f0_coeff,f1_coeff)
       return magnitudes[]
   def _determine_pulse_bit() return boolean
   def _cross_correlation(amplitudes,reference)
       return offset
   def _read_subsequent_pulse(amplitudes) return bits[]
   def _detect_preamble_pulse(amplitudes,coeff)
       return magnitude
   def _read_pulse(stream) return amplitudes
   def _listen() return bits[]
```

Listing 3.5 Physical layer functional overview

In Listing 3.5, the functionality of our physical layer is shown. The Audio class contains the methods for transmitting/receiving acoustic signals, the Goertzel algorithm, cross-correlation synchronization, pulse shaping, the preamble pulse, and pulse visualization.

```
datalink
class Frame()
   def _pad_list_zeros(list) return padded_list
   def _string_to_raw_string(string) return raw_string
   def _binary_list_to_integer(binary_list) return integer
   def _integer_to_binary_list(integer) return binary_list
   def _binary_list_to_string(binary_list) return string
   def _string_to_binary_list(socket_data)
       return binary_list
   def _update_frame(new,index)
   def _calculate_crc() return crc
   def _check_crc() return boolean
   def _set_crc()
   def _get_ack() return boolean
   def _set_ack(ack)
   def _get_duplicate_id() return boolean
   def _set_duplicate_id(duplicate_id)
   def _get_port() return integer
   def _set_port(socket)
   def _get_length() return integer
   def _set_length(socket_data)
   def _get_payload() return raw_string
   def _set_payload(socket_data)
class Protocol()                              #thread
   def master_protocol(rx_queue, tx_queue)
   def slave_protocol(rx_queue, tx_queue)
   def _print_results()
```

Listing 3.6 Data link layer functional overview

In Listing 3.6, the functionality of our data link layer is shown. The Frame class contains the methods for reading and setting the frame data, and changing data types. The Protocol class contains the channel access method for the master and slave devices.

```
sockets
class Sockets()
   def create_socket(port)                     #thread
   def _rx_queue_to_sockets()                  #thread
   def _socket_to_tx_queue(socket)             #thread
```

Listing 3.7 Sockets interface layer functional overview

In Listing 3.7, the functionality of our sockets interface layer is shown. The Sockets class contains the methods for forwarding data between sockets and TX/RX queues.

3.3 Program Features

Program features entail the controls for the software. Command-line flags control these features of the program. Issues encountered in developing the ultrasonic channel were remedied with the adjustable program features.

3.3.1 Application Settings

These are the program settings required for external applications to use the ultrasonic channel.

3.3.1.1 Master/Slave Selection

The selection for setting what the device functions as: master or slave. The Internet-connected exfiltration device is the master. The air-gapped system is the slave.

3.3.1.2 Port Mapping

The selection of which ports are used by the program.

3.3.2 Channel Settings

These are program options that facilitated usage and testing. They adjust the variables for the channel and helped debug it. These settings highlight some of the issues encountered in ultrasonic transfer and aspects of the hardware/environment.

3.3.2.1 Pulse Visualization

Displays a visual representation of the wave recorded. This was helpful in debugging, especially to see the preamble pulse. Pulse visualization allows visual assessment of what the program used as the cross-correlation offset. We can visibly see where pulses start and end.

3.3.2.2 Preamble Pulse Magnitude Threshold

We encountered an issue where detection of the preamble pulse was too late. If recording begins too late, the offset is shifted incorrectly for reading the remainder of the pulses. This was solved by setting the preamble pulse magnitude threshold detection lower so that less of a presence of the preamble pulse initiates receiving earlier. Conversely, the preamble pulse magnitude threshold may also be set higher for noisier environments so that it isn't falsely started on noise that was not a starter pulse.

The magnitudes returned by the Goertzel algorithm are relative to the hardware. Different microphones receive sound at different amplitudes. Depending on the hardware, magnitude is usually a value between 0 and ~15. The magnitude threshold default is 0.5. At normal volumes, the laptop microphone received target frequencies around magnitude 2, while the webcam microphone received target frequencies around magnitude 5.

3.3.2.3 Cross-Correlation Synchronization Window Length

The cross-correlation synchronization window length can be adjusted to ensure that the start of the preamble pulse is detected as the correct offset. Initially, the channel had a chance of missing the preamble pulse and instead matching the subsequent data pulse. Shortening the cross-correlation window size eliminates the chance of this occurring.

3.3.2.4 Pre-Transmission Pause

A slight pause before transmission of a frame ensures that the receiving device has begun recording. This prevents the start of the signal from being continually cut off due to potential system-specific processing time.

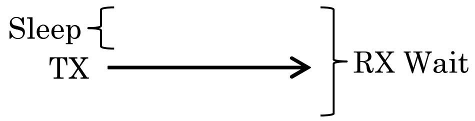

Figure 3.14 Pre-transmission pause

Figure 3.14 illustrates the adjustable sleep time before transmission. The transmission occurs before the receiver times out. The sleep time can be 0 or greater as required, but as small as possible to not add delay to the channel.

3.3.2.5 RX Wait Time

The RX wait time sets how long a device waits for a response. This is illustrated in Figure 3.14. The RX wait time must be longer than the pre-transmission pause in order to ensure reception given the delay.

3.3.2.6 Post-Transmission Empty Padding

We observed an issue with demodulating incorrectly near the end of frames. It was necessary to transmit a tail of amplitude 0 after the pulse, otherwise the speaker stops outputting and the audio stream can get cut short before the last bits are sent. This was likely due to inaccuracy in the audio library control of the hardware. Such an error can make transmissions fail entirely. There must be sufficient zeroes padding in the tail. The number of zeroes required is hardware specific and our default is 8 zeroes, which should be enough for most devices.

Stopping the speaker abruptly can ruin the end of a transmission, distorting the last bits. This was solved by stopping output to the speaker gently, having a buffer at the end before releasing the speaker.

3.3.3 Validation Test Settings

These are program options that allow for testing the physical channel capability. They were used in our validation activities as they correlate to the variables tested in next chapter. This assisted our testing of characteristics of ultrasonic channels. The settings can also optimize the channel depending on its usage requirements (e.g., transfer distance).

3.3.3.1 Payload Length

Setting the optimal frame length depends on how much data can usually be ultrasonically transferred before errors are present. The payload length can be any value. We chose an upper bound of 512 bits. In the validation activities, we discuss an optimal value for payload length.

3.3.3.2 Pulse Length

Setting the optimal pulse length depends on how small a pulse can be emitted by the speaker and still be read correctly by the microphone. The bitrate of the channel is inversely proportional to the pulse length.

As the cross-correlation algorithm has complexity $O(n^2)$, reading in pulse lengths that are too long will cause delay in processing. We started to notice processing delays when pulses were about 1 000 samples long.

3.3.3.3 Sampling Rate

Setting the optimal sampling rate depends on the fastest sampling rate achievable without errors occurring. The bitrate of the channel is directly proportional to the sampling rate.

3.3.3.4 Frequency Channels

Setting the frequencies used by the ultrasonic channel (i.e., the two frequencies in BFSK) depends on the upper frequency range of the equipment used. Lower quality equipment may require lower frequencies for accurate transfer without errors.

$$f = \frac{f_s}{l} n$$

where
f = channel frequency (Hz)
f_s = sampling rate (Hz)
l = pulse length (samples)
n = whole number (cycles)

Equation 3.3 Whole number frequency cycles

Using Equation 3.3, the frequencies were adjusted to have a whole number of cycles. Although its effects were not significant due to pulse shaping, this was performed to perfect the number of cycles and to help rule out alignment issues. By default, 19 110 Hz and 19 600 Hz are used as the channel frequencies since they are ultrasonic frequencies that have a whole number cycles at the 44 100 Hz sampling rate with a pulse length of 180 samples.

3.3.3.5 Goodput Trial

Performs a test of overall perceived throughput, that is goodput, the amount of payload data delivered over time. In the validation activities, goodput is used to measure the practical performance of the channel.

3.3.4 Auxiliary (AUX) Audio Cable Transfer

After development of the ultrasonic channel, we tested transferring data directly via the 3.5mm AUX audio cables. Although the ultrasonic channel software was not designed to transfer data directly via audio cables, we found that the software functioned identically. It was possible to remove the speakers and microphones from the channel entirely, directly transferring data via audio cables.

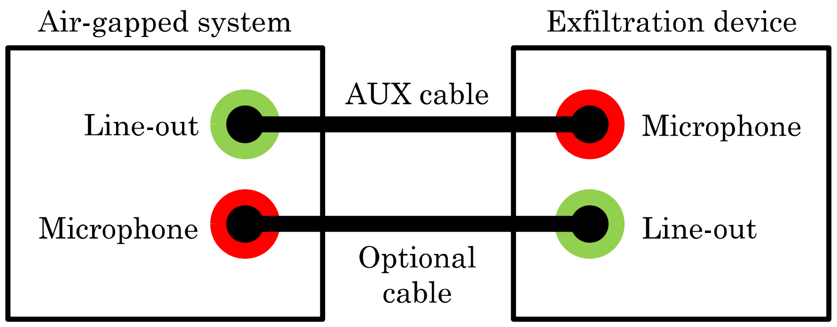

Figure 3.15 AUX cable data transfer

Figure 3.15 shows the line-out jack outputting to the microphone jack by AUX cable instead of via acoustic waves in air. Frequency waves in the audio cable propagate much more reliably than in air. The pulse was visibly more accurate in the AUX cable. This means that frames can be longer, pulses can be shorter, and the sampling rate increased. A two-way channel can be established with two AUX cables. If the two-way channel is unavailable, a one-way channel can transfer data very reliably (line-out to microphone) without error-correcting feedback to the air-gapped system.

Audio cables can, of course, use the entire frequency spectrum silently instead of having to stay within the ultrasonic range to avoid audible detection. As mentioned in M-ary modulation schemes (see section 3.2.1.6.4), this would allow for significantly higher bitrates. Audio cables can also use higher frequencies in the ultrasonic range. Phase and amplitude modulation would be more reliable in cable than in air. As such, combined modulation schemes such as QAM could increase bitrates.

3.4 Summary

The ultrasonic channel designed consists of a three-layered protocol stack model which organizes the functionality of the program. Layer 1 (Physical Layer) entails our methods for the modulation of acoustic signals, demodulation with the Goertzel algorithm, cross-correlation synchronization, and pulse shaping. Layer 2 (Data Link) implements the framing of data and an appropriate channel access method. Layer 3 (Sockets Interface) uses sockets to connect the ultrasonic channel to external programs. The program features detail the settings of our program. Lastly, the ability of acoustic software to directly transfer over AUX audio cables was presented.

4 Validation Activities

In the previous chapter, we described the design and development of our channel in terms of a three-layered protocol stack. Additionally, we listed program features that allow for validation testing of the channel. Validation testing was performed to demonstrate that the channel effectively bridges an air gap by acting as a medium for the transfer of information between computers. The characteristics of the channel that were investigated include usable bitrate, frequency discrimination, and operation in noise. The observable performance of the channel may then be used to assess its effectiveness and limitations, and help characterize the ultrasonic channel.

4.1 Experimental Setup

Our ultrasonic channel was developed and tested in a simulated air-gapped environment comprised of a desktop computer and a laptop. Our equipment included several speakers and microphones. By using low-grade hardware, we tested the ability of an ultrasonic channel to function over common consumer-grade speakers and microphones. We observed the capability of acoustic channels to transfer data ultrasonically on inexpensive hardware not designed to be used at ultrasonic frequencies, meaning the ultrasonic channel is very likely to work on most hardware. The validation test settings described in section 3.3.3 were used to test the channel.

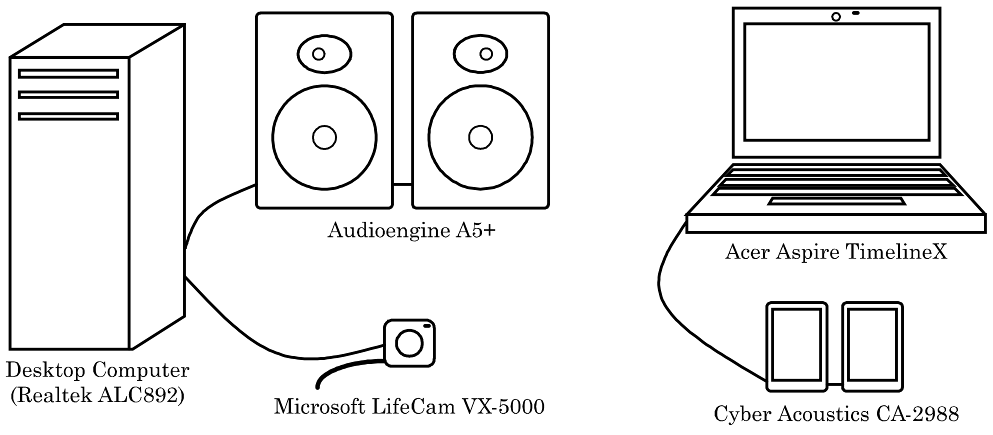

Figure 4.1 Experimental setup diagram

Figure 4.1 illustrates the primary equipment configuration we used in development and validation testing.

Equipment	**Type**
Acer Aspire TimelineX 3830TG-6412	Laptop (with microphone)
Realtek ALC892	Audio chipset
Audioengine A5+	Speakers
Cyber Acoustics CA-2988	Speakers
Logitech Z130	Speakers
Microsoft LifeCam VX-5000	Webcam (with microphone)
Logitech C200	Webcam (with microphone)
Logitech C525 HD	Webcam (with microphone)
Genius MIC-01A	Desktop microphone
iCan OV-M380A	Desktop microphone

Table 4.1 Equipment list

The ultrasonic channel was tested to be functional over all of the equipment shown in Table 4.1. The channel was mainly developed over the Audioengine A5+, Microsoft LifeCam VX-5000, Acer Aspire TimelineX 3830TG-6412, Cyber Acoustics CA-2988, and Realtek ALC892 (see Figure 4.1). The Microsoft and Logitech webcam microphones are higher quality and performed better than the inexpensive Genius and iCan desktop microphones. Weaker microphones had to be closer to the speakers and were more likely to have errors in the bit sequences.

In development, the channel was sensitive to the positioning of the microphone from the speaker. When the channel was fully developed, we were able to test at further ranges. The directionality of the microphone was generally in line with the speaker, but this was not required. The placement of the microphone can be angled from the speakers.

We also assessed operation in noise, the transmission of ultrasonic frequencies despite noise. Several noise models were used to validate this capability. Operation in noise (e.g., computer fans) was simulated with a program that generated white noise [58].

Demonstrating the feasibility of ultrasonic channels on one platform sufficed; the design concepts in this thesis can be ported to other systems. Preliminary testing with spectrum analyzer applications showed that the transmission and reception of ultrasonic frequencies on mobile devices was completely feasible.

4.2 Functional Testing

Functional testing ensured that the methods were correct as they were developed. Each layer was proven functional before developing the next. Data was first shown transferrable at the physical level. As the data link layer was developed, the handling of ultrasonic signals was improved on the physical layer. The data link protocol ensured reliable ultrasonic transfers. The channel access method protocol was verified to transfer correctly. We tested the channel by ensuring that randomly generated strings could be transferred reliably. After the sockets interface layer was implemented, the software was tested externally. We ensured successful transfer through the layers of the protocol stack.

As sockets enabled external applications to connect to the ultrasonic channel, Netcat was used to transfer data. The primary program used to test the ultrasonic channel was Netcat, acting as a TCP/IP connection program to test the sending of characters. File transfers through Netcat were also performed. These transfers were verified to be correct with Message Digest 5 (MD5) checksums.

Sockets enabled application layer protocols to run over our ultrasonic channel. Entire application-level protocols were also transferred ultrasonically. Secure Shell (SSH) functioned, but was quite slow due to the encryption requiring larger amounts of data. A simple Python Hypertext Transfer Protocol (HTTP) server was set up for testing. HTTP was confirmed functional over the ultrasonic channel. File Transfer Protocol (FTP) was also functional, although it was more difficult to configure because of its separate data and command channels. The ultrasonic channel was tested successfully with large transfers using Netcat.

4.3 Validation

The following validation activities assess the performance of the channel under different variables. This serves as a rough assessment of ultrasonic channel characteristics. Transfers at close range (<2 m), without any noise or anomalies, would go through successfully 100% of the time. Any errors were caused by the inability to interpret the signals due to the intended stressor (e.g., distance, noise). Errors result in CRC checks failing, forcing the channel to retransmit. There are different points at which the channel can be hindered: hardware, equipment (speakers and microphones), processing, and the acoustic medium. Given the constraints of our experimental setup, we have demonstrated stress-testing of the acoustic medium.

We use throughput as a generic term to describe the bitrate of the channel. Goodput is the effective speed of transfer of the channel, not counting the frame overhead bits transferred as well. Goodput only considers the payload data delivered. Goodput is a suitable measure given the large overhead. In validation, 10 to 30 transfers were performed per goodput trial (with the setting in section 3.3.3.5). Although a measure of throughput would show an even higher bitrate, we only consider the payload data as valuable data transferred.

4.3.1 Channel Bitrate

Although the aim of the research was only to build a functional ultrasonic channel, the channel bitrate exceeded our original expectations. A binary modulation scheme (BFSK) was used, serving as a reference for base performance. It follows that a quadrature modulation scheme implementation would have twice the bitrate. A means of increasing the channel throughput would be additional frequency channels.

4.3.1.1 Variable Frame Length

The payload length was tested (with the setting in section 3.3.3.1) in relation to goodput to find an optimal frame length. A payload length that is too short does not effectively make use of the frame overhead required. However, a reason to use shorter payload lengths is that frames of excessive length are more prone to failure, as a single incorrect bit ruins the entire frame. A reasonable frame length is better in practice given the likeliness of error and response waiting time.

For stable experimentation, we used a sampling rate of 44 100 Hz and 180 samples/pulse at a distance of <2 m. With this sampling rate and pulse length, we observed a throughput of about 250 bit/s. The frame overhead was 40 bits. These were the default settings used in the validation tests for our channel. The starting payload length was 1 byte. We increased the number of payload bytes to observe the corresponding goodput.

$$\frac{44\,100\text{ samples}}{1\text{ second}} \cdot \frac{1\text{ pulse}}{180\text{ samples}} \cdot \frac{1\text{ bit}}{1\text{ pulse}} \cdot \frac{8\text{ payload bits} \cdot \text{payload length}}{40\text{ frame bits} + 8 \cdot \text{payload length}}$$

Equation 4.1 Theoretical goodput

Equation 4.1 shows the calculation of goodput given our default settings. Throughput is calculated by dividing the sampling rate by the pulse length and multiplying by the number of bits per pulse. Goodput is calculated by multiplying the throughput by the ratio of payload data in each frame. The units cancel out, leaving the final unit of bit/s. If a quadrature modulation scheme were used, the calculation would use 2 bits/pulse (instead of 1 bit/pulse), multiplying the bitrate by two times.

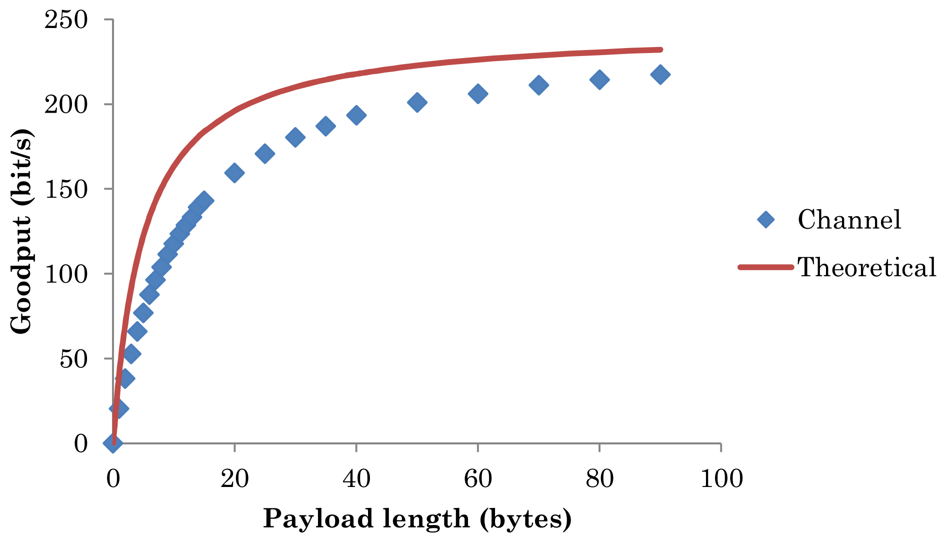

Figure 4.2 Goodput vs. Payload length

In Figure 4.2, the theoretical curve is what the goodput would be ideally, without any waiting time between transmissions. The channel curve is the actual performance of the channel. The channel goodput is slightly lower than the theoretical goodput due to practical delays in the channel software (e.g., pre-transmission pause).

A reasonably optimal payload length is found where the curve begins to plateau. We can see that after a payload length of 30 bytes, the channel does not gain much goodput with further increases to the payload length. We selected 30 bytes as the payload length to use in the remainder of the validation tests. As such, goodput is expected to be about 180 bit/s in the following tests.

It should be noted that the channel can be faster with more optimal settings such as a smaller pulse length, faster sampling rate, and reduced overhead (e.g., error-detecting code CRC8 instead of CRC16).

4.3.1.2 Variable Pulse Length

The pulse length was tested (with the setting in section 3.3.3.2) in relation to goodput at a distance of <2 m. A shorter pulse length results in an increased channel bitrate. However, the channel is unable to demodulate correctly if the pulse length is too short. We tested for the minimum pulse length required.

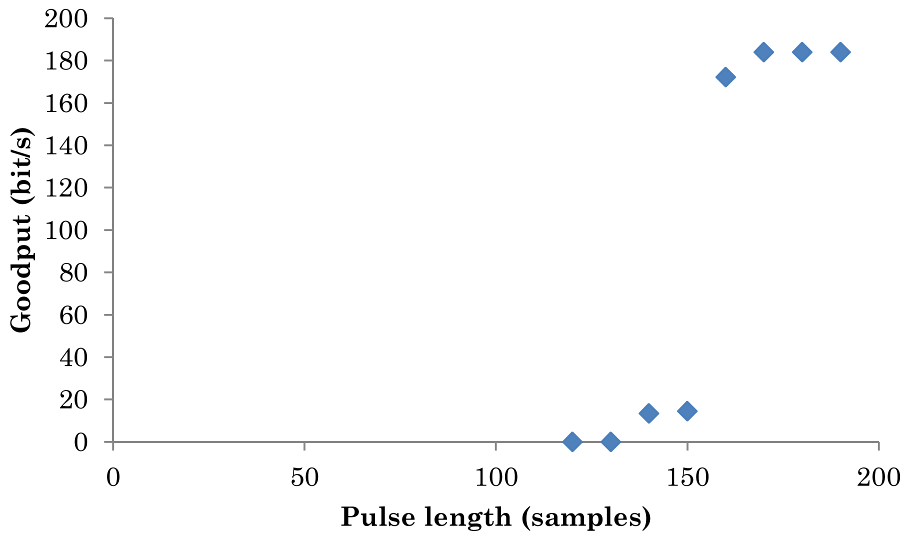

Figure 4.3 Goodput vs. Pulse length

Figure 4.3 shows that at least 160 samples were required for a pulse to be demodulated correctly and achieve the expected goodput of about 180 bit/s. The channel default was set a little longer, to 180 samples, for increased stability at further ranges. A pulse length of 180 samples at the 44 100 Hz sampling rate takes approximately 4.08 ms. The pulse length can be shortened to increase the speed of the channel. A pulse length of 180 samples yields a throughput bitrate of 245 bit/s (44 100 Hz ÷ 180 samples), while a pulse length of 170 samples yields a throughput bitrate of 259 bit/s (44 100 Hz ÷ 170 samples).

4.3.1.3 Variable Sampling Rate

The sampling rate was tested (with the setting in section 3.3.3.3) in relation to the channel defaults. A sampling rate that is too slow is unable to demodulate the pulses correctly. Slower sampling rates require a longer pulse length to demodulate pulses correctly. A faster sampling rate correlates to higher goodput. However, a sampling rate that is too high is unstable.

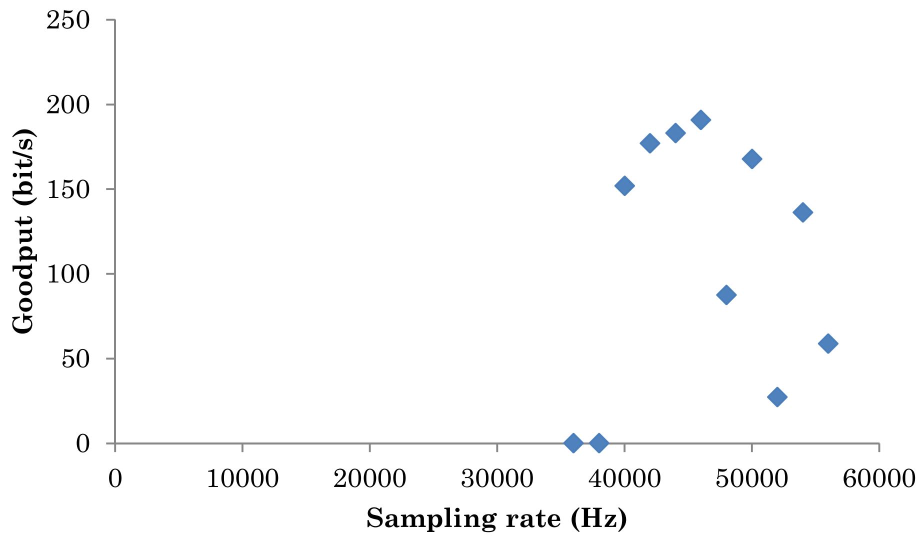

Figure 4.4 Goodput vs. Sampling rate

Figure 4.4 shows that a sampling rate over 40 000 Hz was required to demodulate pulses 180 samples long. Throughput increases along with an increase in sampling rate to about 45 000 Hz. When the sampling rate is above 45 000 Hz, the alignment of pulses is sometimes inaccurate and frames may be misread. The channel was unreliable after 45 000 Hz with default settings. The points plotted after 45 000 Hz were highly variable, and these examples were plotted to show their variability. Faster sampling rates would require the pulse length to be proportionally larger. We did not increase the pulse length, but kept it constant at 180 samples/pulse to isolate for the change in sampling rate. The default sampling rate was set to 44 100 Hz.

4.3.2 Frequency Discrimination

The concept of frequency discrimination was previously presented in the design of our channel. The frequencies used in FSK had to be placed a sufficient spectrum distance apart in order to be demodulated correctly as distinct frequencies. The frequency distance required to tell frequencies apart determines how many frequency channels can fit into the ultrasonic range. We tested (with the setting in section 3.3.3.4) how closely the frequencies can be placed together and still be consistently demodulated as distinct frequencies. This was performed by setting the two frequencies used in our software.

Frequency magnitudes were smaller and less comparably different from each other at higher frequencies. Magnitudes are smaller at frequencies above 18 000 Hz, leading to more demodulation errors. The weaker microphones we used had lower magnitudes returned from the Goertzel algorithm and therefore less ability to discriminate between frequencies. It is preferable to have a larger usable frequency bandwidth for better discrimination.

Frequency discrimination determines the channel bitrate as the ability to fit two more frequency channels means that QFSK is possible in the ultrasonic range. QFSK provides double the bitrate of BFSK.

4.3.2.1 Ultrasonic Frequency Response

The ultrasonic frequency response of the microphone was tested to find the usable upper bound frequency. We noticed that inexpensive microphones detected frequencies higher than advertised in their product specifications. One microphone specification stated frequency response up to 16 000 Hz, but it was able to detect up to 19 500 Hz. Figure 4.5 shows the ultrasonic frequency response of the primary equipment used in development (i.e., laptop microphone).

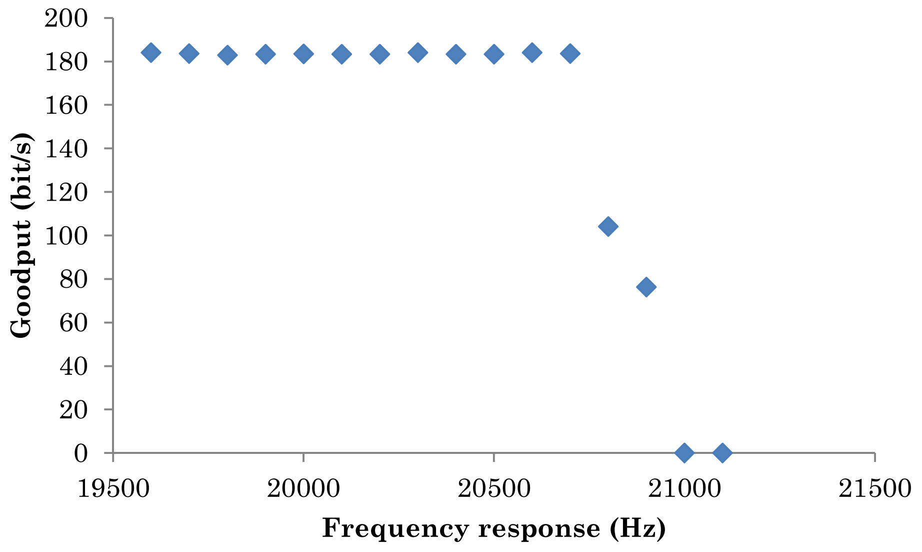

Figure 4.5 Goodput vs. Frequency response

Since the laptop microphone was weaker than the Microsoft LifeCam VX-5000, it was the device limiting ultrasonic frequency response. The laptop microphone was able to detect ultrasonic frequencies reliably up to approximately 20 700 Hz.

4.3.2.2 Frequency Discrimination

The default frequencies used were 19 110 Hz and 19 600 Hz. These two frequency channels used in BFSK were brought closer together until the channel was unable to discriminate between the frequencies. Figure 4.6 shows how closely the frequencies can be placed and still maintain full goodput.

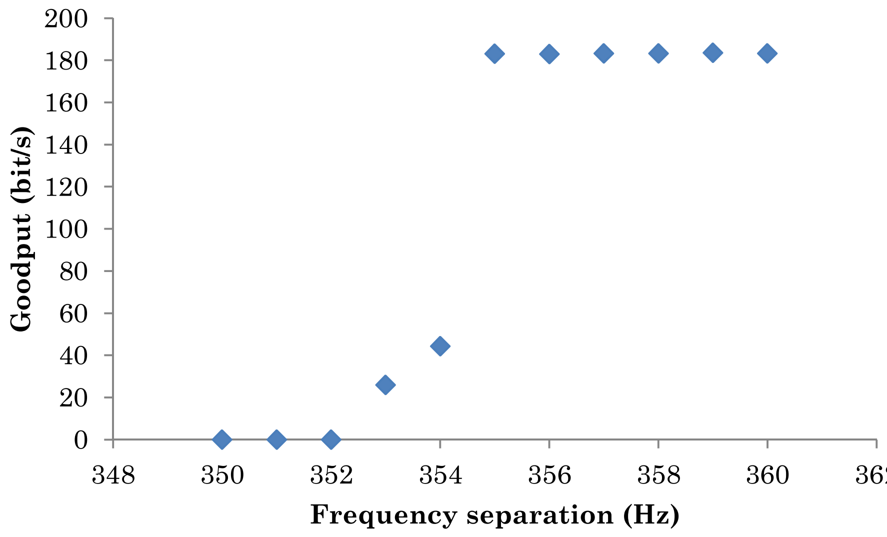

Figure 4.6 Goodput vs. Frequency separation

The frequency discrimination was smaller than expected from the visible signal bandwidth seen in spectrum analysis. Previously, we described using 500 Hz of separation in our design. The frequency separation may be decreased to 355 Hz or greater if the usable ultrasonic range were more limited.

We observed that two more channels can fit within the ultrasonic range, allowing for QFSK. Given the previously stated ultrasonic frequency response up to 20 700 Hz and frequency discrimination of 355 Hz, we can fit four frequency channels in the ultrasonic range. QFSK is possible within the ultrasonic range, enabling double the speed of BFSK. As we have previously demonstrated bitrates of about 250 bit/s with BFSK, it follows that QFSK would have 500 bit/s.

4.3.2.3 Frequency Discrimination with White Noise

The same frequency discrimination test was repeated along with white noise being generated in the background.

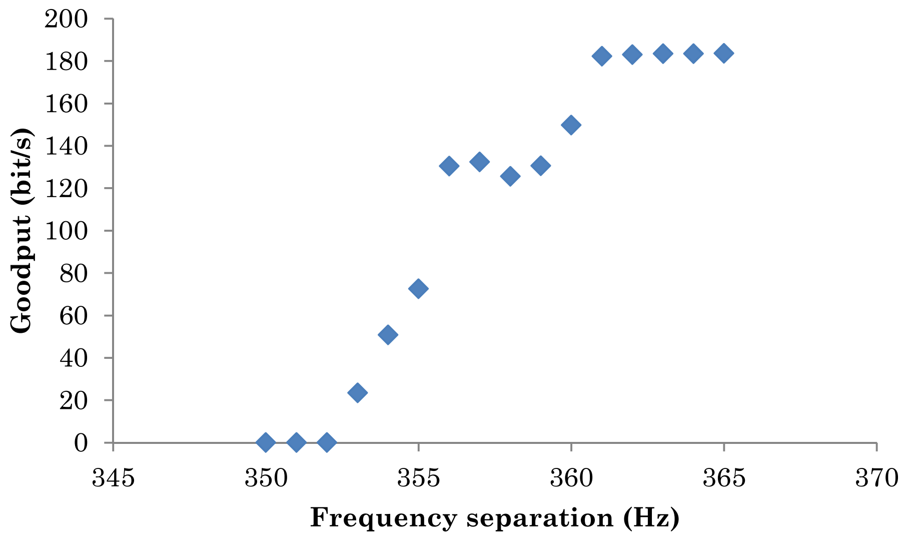

Figure 4.7 Goodput vs. Frequency separation with noise

In Figure 4.7, we can see that it is slightly more difficult to discriminate between frequencies in the presence of noise. The volume of white noise generated from an external source ~3 m away was measured to be 70 dB at the channel. Errors occurred with less frequency separation; therefore, the channel required greater frequency separation to be stable. Imperfections in the graph are due to practical testing inaccuracy.

4.3.2.4 Frequency Discrimination in Audio Cable

The same frequency discrimination test was repeated in the 3.5 mm AUX audio cable. It can be expected that transfers in the AUX cable are more stable than in air.

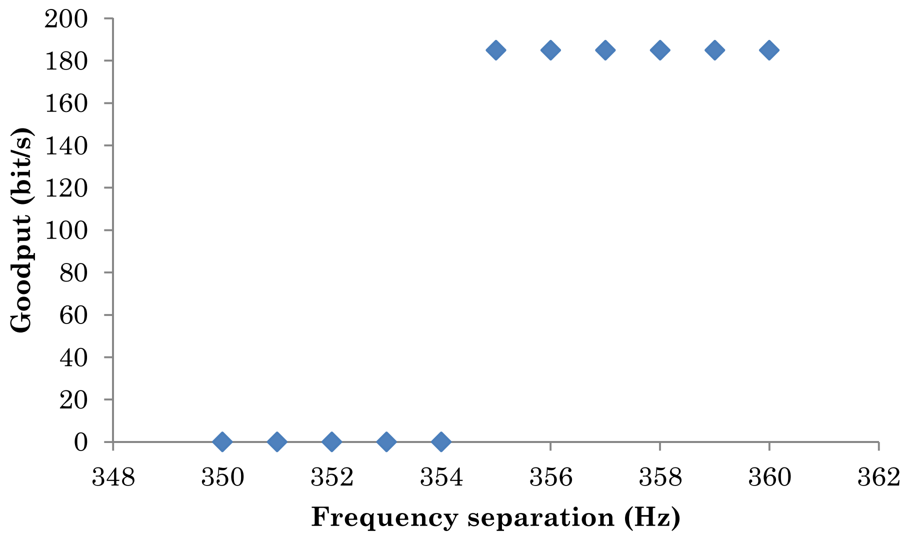

Figure 4.8 Goodput vs. Frequency separation in AUX cable

As seen in Figure 4.8, we observed that 355 Hz separation was required for frequency discrimination. This confirms the same minimum frequency separation as in air.

4.3.3 Distance and Noise Models

Noise impairs correct demodulation. The ultrasonic channel was measured in the presence of noise to find the level of noise required to prevent demodulation. The default channel volume was about 87 ± 2 dB. Ambient noise before applying white noise was 45 dB in volume.

Since loud noises have great amplitude, their signal bandwidth spills into the range of high frequencies. For example, a loud enough cough or clapping noise would distort the ultrasonic channel. A loud noise can exceed the preamble pulse magnitude threshold to initiate recording. Loud noises also affect the correct offset position in cross-correlation.

White noise was produced with a program [58] randomly generating amplitudes from -1 to 1. We used the white noise program to jam the ultrasonic channel. We tested for the extent of noise that distorts the channel.

4.3.3.1 Distance

The distance at which the channel performs was tested. When the speaker to microphone distance is too far, data can no longer be transferred. The channel was measured at increasing distances. Some equipment functioned at further distances more reliably than others. As expected, the channel can transfer further with a higher speaker volume.

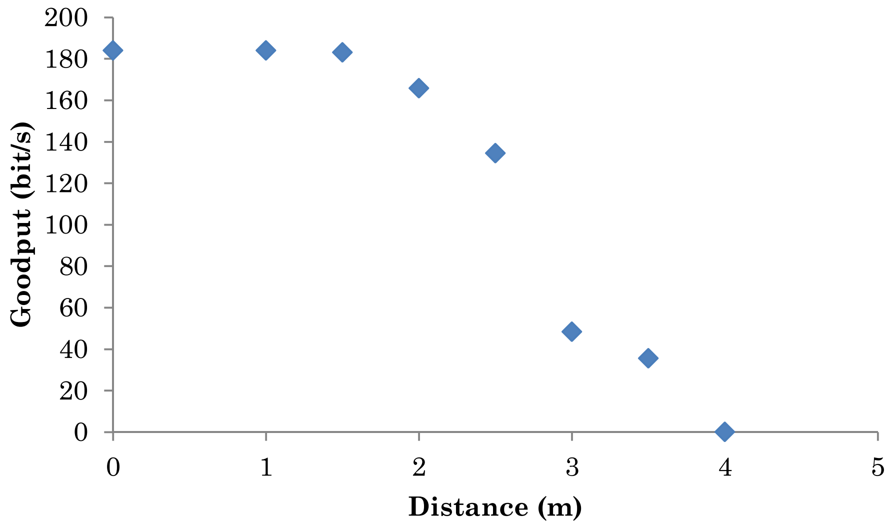

Figure 4.9 Goodput vs. Distance

As shown in Figure 4.9, the channel with default settings was impaired after 2 m and did not work beyond 4 m.

4.3.3.2 White Noise from External Source

White noise was generated from an external source to find the level of noise required to disrupt the channel. The external noise source was placed ~3 m away from the ultrasonic channel. Noise testing was conducted in ambient noise (not in an anechoic chamber) as an estimate for practical implementation.

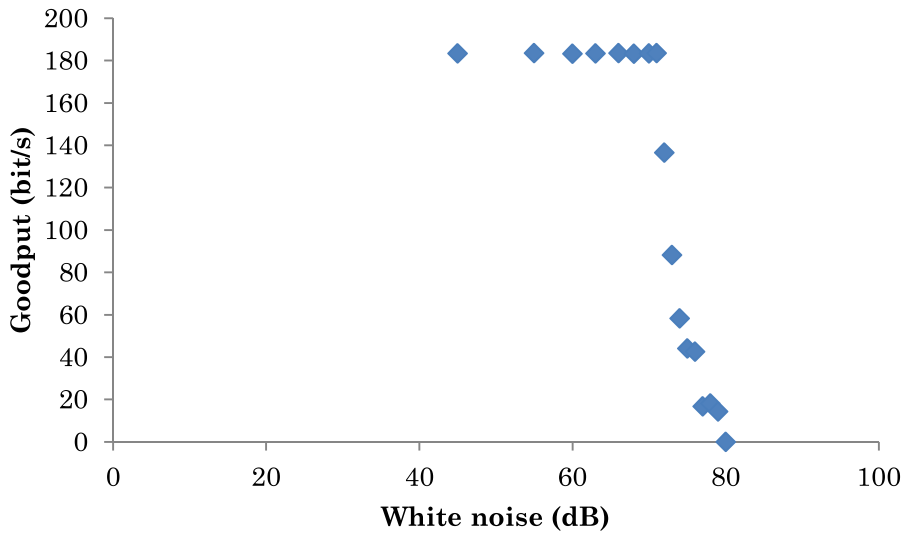

Figure 4.10 Goodput vs. White noise from external source

As seen in Figure 4.10, the channel was disrupted after 70 dB of noise and completely unable to function after 80 dB of noise. As such, the volume of noise required to impair the channel was about 17 dB lower than the channel volume at 87 dB.

4.3.3.3 White Noise from Self

White noise was generated from the speakers used in the ultrasonic channel itself to find the level of noise required to disrupt our channel.

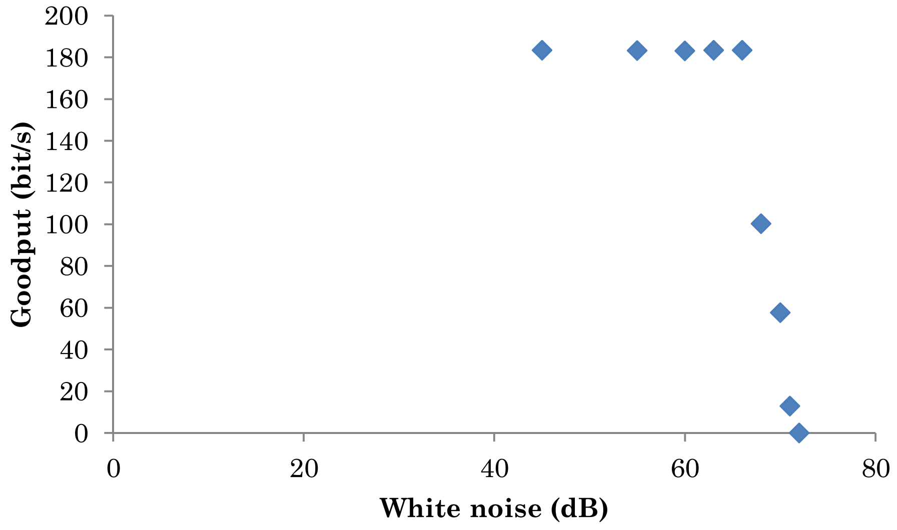

Figure 4.11 Goodput vs. White noise from self

As shown in Figure 4.11, after 66 dB of noise the channel was disrupted, and completely unable to function after 72 dB of noise. Effectively, the volume of noise required to impair the channel was about 21 dB lower than the channel volume at 87 dB. As expected, the channel was disrupted with less noise than from an external source.

4.4 Summary

The experimental setup has been described along with functional testing of the software during development. The validation activities show the performance of our ultrasonic channel and also provided some insight into the characteristics of ultrasonic channels. Additionally, the validation activities highlight further potential for increasing the performance of ultrasonic channels.

5 Discussion

The literature review described our motivation for studying ultrasonic covert channels in the context of computer network security. The proof-of-concept ultrasonic channel was developed in order to demonstrate the feasibility of such a channel. As part of the design process, we performed a preliminary analysis of digital modulation schemes. The validation activities showed the performance of our channel. Characteristics of the channel that were measured include channel bitrate, frequency discrimination, and operation in noise. The implementation of a sockets-layer interface showed the suitability of the channel for generic high-level computer communications. This chapter analyzes the effectiveness and limitations of the ultrasonic covert channel, and concludes the thesis with comments on future work.

5.1 Channel Effectiveness

This research confirms that ultrasonic covert channels are possible. Following from the breach of an air-gapped network, an ultrasonic channel may be used to exfiltrate information. Such a channel can allow sustained access to the network over time. The channel may not need the physical presence of an intruder. As discussed in the literature review, air gap breaching malware has shown the capability to spread via removable media. Ultrasonic channel malware running across multiple devices would improve the chances of successfully exfiltrating data. In practice, the malware installation would be more covert (e.g., kernel-level hiding) [9]. Ultrasonic channels are not commonly expected and detected in computer network security. The channel software shows that the compromised machines do not have to beacon out. Only a master device must try and contact out as desired or at regular intervals. This would be difficult to detect because the channel does not need to beacon out constantly. Once the channel is established, all of the necessary transfers can take place. Our ultrasonic channel software was developed as an example of this operation.

Our ultrasonic channel design demonstrated accurate transfers. We observed 100% successful transfers at close range, where the setup was not disturbed. The channel bitrate was sufficient for transferring small amounts of data, such as command-line interaction and text. The Goertzel algorithm was effective for demodulation. Processing the Goertzel algorithm was not a bottleneck since it is not processing intensive. The usage of higher bitrates provided an assessment of the acoustical medium itself. There would not have been a significant gain with faster processing of the Goertzel algorithm and cross-correlation in demodulation. The limiting factor was the physical medium and audio hardware. As such, the performance of our ultrasonic channel was observed.

In terms of speed, the potential of ultrasonic channels was demonstrated. We have explored different methods to achieve even higher bitrates. Each digital modulation scheme (FSK, PSK, ASK) was investigated in binary and quadrature implementations. Frequency discrimination only required 355 Hz of separation, meaning that that four frequency channels can be used within the ultrasonic range. Realistically, we would want to set a greater frequency separation to reduce the chance of error, such as 380 Hz. A quadrature modulation scheme provides twice the bitrate of a binary modulation scheme. The measured bitrate of 250 bit/s for BFSK would be 500 bit/s in QFSK. With a frequency separation of 380 Hz, only 1 140 Hz of bandwidth (i.e., 380 Hz × (4-1)) is required to convey four frequencies, as opposed to our initial design estimate of 1 500 Hz. It is also possible to use 3 frequencies and a pause to represent the four symbols if ultrasonic bandwidth is limited, which would require 760 Hz of bandwidth (i.e., 380 Hz × (4-2)). It may even be possible to fit FSK8 within the ultrasonic range using 7 frequencies and a pause, which would require 2 280 Hz of bandwidth (i.e., 380 Hz × (8-2)). FSK8 would have an estimated speed of 1 kbit/s. DBPSK allows for the transfer of phase modulated information on a single frequency, saving the frequency bandwidth required. DQPSK was also implemented, although it was not as accurate as DBPSK. PSK may be combined with FSK (i.e., FPSK) to achieve higher bitrates, although it is prone to error. Significantly higher bitrates can be achieved if the audible frequency bandwidth can be used. In the audible range of frequencies, 32 frequency channels can be used with M-ary FSK. 32 frequency channels would allow the transfer of 16 bits/pulse, thus increasing the binary modulation scheme's bitrate by 16 times (16 × 250 bit/s = 4 kbit/s). Relative to the classic 56 kbit/s dial-up modem, this is very fast for acoustic transfers. It is likely that audible frequencies may also require less Hz of separation for frequency discrimination because lower frequencies are more easily discernable. Audible frequencies retain information better and transfer further than frequencies in the ultrasonic range.

We have demonstrated bitrates of about 250 bit/s, which is higher than the 20 bit/s reported by Hanspach and Goetz [4]. As previously discussed, implementation of QFSK would achieve a bitrate of 500 bit/s. This would demonstrate a higher bitrate at close range than reported by Deshotels (300 bit/s) [50] and Carrara (230 bit/s) [11].

Considering the fast bitrate and small size of the pulses, we have shown a measure of bitrate at close range. Longer pulses would be required for greater distances. There is a tradeoff between bitrate and distance. Closer range allows for higher speeds, which was what our channel showed. Transfers at further distances need to have slower bitrates in order for the information to be retained acoustically. This was evident in other research that showed transfers at a greater distance at the expense of slower bitrate (i.e., 20 bit/s at 19.7 m) [4]. Our focus was on the speed of the channel, how small the pulses can be made while still demodulating reliably. In terms of distance, the ultrasonic channel requires devices to be at fairly close range. However, the software can be set to transfer over greater distances at slower speeds. A concept would be to make the channel self-optimize speed according to distance. It would run a test periodically that determines signal strength. For example, the channel starts with slower transfers using long, slow pulses to enable transfers at the furthest distances. Shorter pulses and faster transfers are then used for closer distances. The channel would test for and adjust the pulse length, thereby achieving the highest bitrates given the distance.

We demonstrated data transfers in 3.5 mm AUX audio cables. The entire audible frequency bandwidth and as many frequencies possible in the ultrasonic range can be used (e.g., with M-ary FSK) in AUX cables, providing much faster bitrates, around 4 kbit/s. Because phase and amplitude modulation are more reliable over a cable than in air, modulation schemes combined with FSK could increase bitrates over AUX cables. Since transfers in AUX cables were much more accurate than in air, correctional feedback is not essential and data can be transferred reliably one-way. We confirmed that audio jacks are a means of transferring data out of a system.

The channel is able to broadcast on top of noise/speech/music. Because the Goertzel algorithm finds the magnitude of specific frequencies, it is sensitive to the ultrasonic frequencies the channel uses and resilient to noise such as music. We observed that playing music and speaking do not disturb the channel significantly because they tend to be below ultrasonic frequencies, whereas white noise spans the entire frequency spectrum randomly. The channel works when playing music through the same speakers that are transmitting the ultrasonic channel. As such, ultrasonic channels may be hidden in the background while broadcasting other audio such as music or speech. Figure 5.1 is a spectral analysis performed while running the channel simultaneously with audible noise in the background. The ultrasonic channel resides at about 20 000 Hz, while audible noise is visible lower in the frequency spectrum.

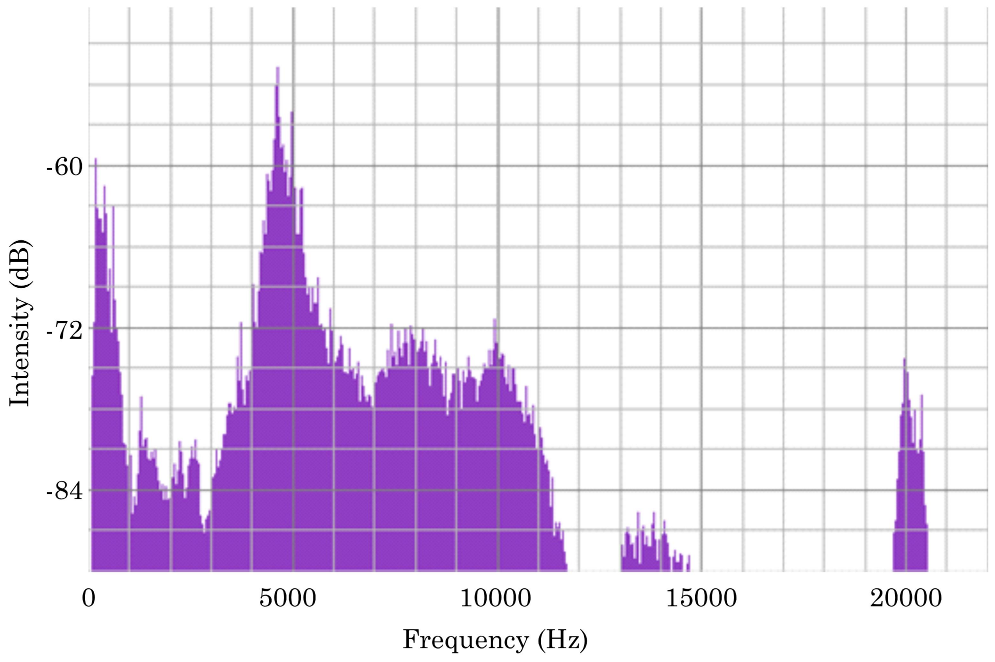

Figure 5.1 Ultrasonic channel hidden in audio

The channel is effective where speakers are in place for the purpose of transmitting audio. Microphones are common on mobile devices.

Existing audio channels (i.e., for voice communication) may be used. Depending on the scenario requirements, the channel can be changed to be one-way with forward error correction, where correctness is not guaranteed. Such a channel does not require both speakers and microphones on both devices, but rather only one speaker and one microphone. With this setup, commands could not be sent and the channel would be used for exfiltration only. Acoustic channels were considered in the context of crossing the air gap, but there may be other general-purpose applications.

Acoustic signals can be used where electromagnetic signals are blocked (i.e., with a Faraday cage/shield). Ultrasonic signals pass through electromagnetic shielding because they are acoustic signals. Devices placed on either side of a Faraday cage would enable ultrasonic transfers to penetrate directly through the Faraday cage.

5.2 Channel Limitations

In terms of malware, an ultrasonic channel requires an initial vector or exploit through which it can be installed on the air-gapped system. It also requires compromised devices with speakers and microphones to be in place in a relatively close range setup. Where speakers/microphones are not on the network, the channel cannot be used. If the air-gapped system does not have a microphone, the ultrasonic channel can be converted to be one-way, beaconing out constantly, and can only exfiltrate data but not receive new commands. Additionally, other research has shown the capability of speaker-to-speaker transfers, where speakers were turned into microphones (with jack retasking) to record ultrasonic signals at a low bitrate of 10 bit/s [59]. An acoustic channel is dependent on the volume of speakers. Speakers physically set to a low volume would not enable a channel to transfer very far. The ultrasonic channel attack has a somewhat limited application but works well where necessary, such as in air-gapped systems.

A limitation is the audio hardware, the ultrasonic frequency response and quality of speakers and microphones. Better equipment may improve the channel proportionally by improving frequency discrimination and transmission range. Tweeters, speakers designed for high frequencies, would be able to leverage more frequency channels in the ultrasonic range. Specialized speakers and microphones with a higher ultrasonic frequency response would permit using a larger range of ultrasonic frequencies, allowing for higher speeds and greater distances of transfer.

Despite having speeds higher than that of previous research, the ultrasonic channel is still slow in comparison to conventional communications channels. For general-purpose information transfers, the slow speed hinders multimedia ability. It is mostly confined to text/numerical data and small media.

In the preliminary analysis of modulation schemes, the difficulties with PSK and ASK were discussed. PSK had a phase drift problem and phase information was not as reliable as the frequency of a wave. ASK uses a reduction in amplitude, which would reduce the distance that the acoustic wave can travel. Although QAM is a popular modulation scheme in wired and electromagnetic communications, it is not suitable for acoustic communications due to the aforementioned issues with PSK and ASK. In FSK, the number of M-ary channels implemented is determined by the ultrasonic frequency bandwidth usable. Loud noises distort ultrasonic signals when large amplitudes cause sufficient distortion in the ultrasonic frequency bandwidth.

Defensive strategies were considered. The existence of ultrasonic channels may suggest caution for devices with speakers/microphones in air-gapped networks. Although it may impair the quality of audio, speakers may be limited to audible frequencies to prevent covert ultrasonic channels. AUX audio ports may be removed or epoxy glued shut where they are not needed. It is also possible to jam inaudible frequencies, however, there may be effects on health to consider. Ultrasonic frequencies can be detected and viewed with spectrum analysis. Ultrasonic detectors may be placed where there are speakers, treating frequencies above the audible range as a potential for ultrasonic channels.

5.3 Future Work

In future research, a deeper investigation into digital modulation schemes for acoustic channels would be beneficial, with the aim of increasing the channel bitrate at the lowest level. In particular, PSK is of interest due to its ability to transfer information at the same frequency, thus conserving frequency bandwidth. If the phase drift issue were solved, then PSK could be combined with FSK to achieve FPSK.

Other methods of exfiltrating information over an air gap would be valuable. Future research may reveal other non-conventional means through which information can be transferred, along with assessment of their performance. The general-purpose applicability of such channels is also of interest.

5.4 Conclusion

We have identified the ultrasonic channel as a potential vulnerability in air-gapped networks. Our software acts as an ultrasonic channel proof-of-concept. It served as an artifact or tool that demonstrates the capability of acoustic channels. The low-level characteristics of ultrasonic channels were examined, where our methods increased the performance of acoustic channels. This thesis serves as a guide for the development of ultrasonic channels and provides a basis for the study of similar channels. Notable observations from our research are mentioned in the following contributions.

5.4.1 Contributions

The Goertzel algorithm was implemented in our ultrasonic channel to increase processing performance. The clicking noise issue mentioned in previous research [4] was solved by pulse shaping. Command-line interaction was demonstrated over the ultrasonic channel. All of the basic digital modulation schemes have been compared in a preliminary analysis. With BFSK, we measured speeds of about 250 bit/s at a distance of 2 m. Using QFSK in the ultrasonic bandwidth was confirmed possible by frequency discrimination, allowing for speeds of 500 bit/s. DBPSK was shown to be functional. DQPSK is also functional, although not as precise. A combined modulation scheme, FPSK, was shown to be possible, although prone to error. We have operated the ultrasonic channel in noise, such as hiding in music. We described the concept of self-optimizing speeds according to distance, where faster speeds can be achieved at closer distances. We showed that acoustic channel software can transfer data directly over AUX audio cables, where the entire audible frequency bandwidth can be used silently. Lastly, we showed the suitability of such channels to support common high-level network protocols by demonstrating multi-session support for existing network applications using generic sockets.

6 References

[1] G. Pajari, 'USB Flash Storage Threats and Risk Mitigation in an Air-Gapped Network Environment', in *CanSecWest Applied Security Conference*, Vancouver, BC, 2014.

[2] N. Falliere et al., 'W32.Stuxnet Dossier - Symantec Security Response', 2011. [Online]. Available: https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf [Accessed: 1-Aug-2018].

[3] D. Sanger and T. Shanker, 'N.S.A. Devises Radio Pathway Into Computers - The New York Times', 2014. [Online]. Available: https://www.nytimes.com/2014/01/15/us/nsa-effort-pries-open-computers-not-connected-to-internet.html [Accessed: 26-Jul-2018].

[4] M. Hanspach and M. Goetz, 'On Covert Acoustical Mesh Networks in Air', *Journal of Communications*, vol. 8, no. 11, pp. 758-767, 2013.

[5] P. Mueller and B. Yadegari, 'The Stuxnet Worm', 2012. [Online]. Available: https://www2.cs.arizona.edu/~collberg/Teaching/466-566/2012/Resources/presentations/topic9-final/report.pdf [Accessed: 27-Jul-2018].

[6] M. Bigueur, 'Chinese APT Analysis “APT30”', 2017. [Online]. Available: https://miguelbigueur.com/2017/10/26/chinese-apt-analysis-apt30/ [Accessed: 16-Aug-2018].

[7] R. Clarke and R. Knake, *Cyber War: The Next Threat to National Security and What to Do About It*. New York, NY: HarperCollins, 2010.

[8] E. Skoudis, 'Netcat Cheat Sheet'. [Online]. Available: https://www.sans.org/security-resources/sec560/ netcat_cheat_sheet_v1.pdf [Accessed: 25-Sep-2018].

[9] C. Chapman, 'USBCat - Towards an Intrustion Surveillance Toolset', M.A.Sc. thesis, Royal Military College of Canada, Kingston, ON, 2013.

[10] E. Couture, 'Covert Channels', SANS Institute, 2010. [Online]. Available: https://www.sans.org/reading-room/whitepapers/ detection/covert-channels-33413 [Accessed: 25-Sep-2018].

[11] B. Carrara, 'Air-Gap Covert Channels', Ph.D. thesis, University of Ottawa, Ottawa, ON, 2016.

[12] A. Mendoza, 'Cold Storage in the Cloud: Trends, Challenges, and Solutions - Intel', 2013. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/w hite-papers/cold-storage-atom-xeon-paper.pdf [Accessed: 12-Aug-2018].

[13] B. Schneier, '"Evil Maid" Attacks on Encrypted Hard Drives', 2009. [Online]. Available: https://www.schneier.com/blog/archives/2009/10/evil_maid_attac.ht ml [Accessed: 25-Sep-2018].

[14] B. Perelman, 'Air Gap or Not, Why ICS/SCADA Networks Are at Risk | SecurityWeek', 2016. [Online]. Available: https://www.securityweek.com/air-gap-or-not-why-icsscada-networks-are-risk [Accessed: 16-Jul-2018].

[15] K. Collins, 'Wikileaks: The CIA can remotely hack into computers that aren’t even connected to the internet - Quartz', 2017. [Online]. Available: https://qz.com/1013361/ [Accessed: 16-Jul-2018].

[16] The Economist, 'The worm turns', 2008. [Online]. Available: https://www.economist.com/united-states/2008/12/04/the-worm-turns [Accessed: 21-Apr-2018].

[17] W. Lynn III, 'Defending a New Domain: The Pentagon's Cyberstrategy', *Foreign Affairs*, vol. 89, no. 5, pp. 97-108, 2010.

[18] N. Shachtman, 'Under Worm Assault, Military Bans Disks, USB Drives | WIRED', 2008. [Online]. Available: https://www.wired.com/2008/11/army-bans-usb-d/ [Accessed: 16-Jul-2018].

[19] B. Anderson and B. Anderson, *Seven Deadliest USB Attacks*. Burlington, MA: Syngress, 2010.

[20] J. Larimer, 'Beyond Autorun: Exploiting vulnerabilities with removable storage', in *Black Hat*, Washington, DC, 2011.

[21] K. Zetter, 'Meet “Flame,” The Massive Spy Malware Infiltrating Iranian Computers | WIRED', 2012. [Online]. Available: https://www.wired.com/2012/05/flame/ [Accessed: 21-Apr-2018].

[22] D. Goodin, 'Meet “badBIOS,” the mysterious Mac and PC malware that jumps airgaps | Ars Technica', 2013. [Online]. Available: https://arstechnica.com/information-technology/2013/10/meet-badbios-the-mysterious-mac-and-pc-malware-that-jumps-airgaps/ [Accessed: 14-Jul-2018].

[23] Hak5, 'USB Rubber Ducky'. [Online]. Available: https://hakshop.com/products/usb-rubber-ducky-deluxe [Accessed: 25-Sep-2018].

[24] A. Zhukov, 'Turning a Regular USB Flash Drive into a USB Rubber Ducky'. [Online]. Available: https://hackmag.com/security/rubber-ducky/ [Accessed: 25-Sep-2018].

[25] S. Kamkar, 'USBdriveby: exploiting USB in style', 2014. [Online]. Available: http://samy.pl/usbdriveby/ [Accessed: 16-Jul-2018].

[26] K. Nohl and J. Lell, 'BadUSB - On accessories that turn evil', in *Black Hat*, Las Vegas, NV, 2014.

[27] NSA, 'NSA ANT Catalog - USB', 2008. [Online]. Available: https://nsa.gov1.info/dni/nsa-ant-catalog/usb/index.html [Accessed: 16-Jul-2018].

[28] R. McMillan, 'The Pwn Plug is a little white box that can hack your network | Ars Technica', 2012. [Online]. Available: https://arstechnica.com/information-technology/2012/03/the-pwn-plug-is-a-little-white-box-that-can-hack-your-network/ [Accessed: 16-Aug-2018].

[29] Holland Shielding Systems, 'Tempest solutions against stealing information - Faraday Cages'. [Online]. Available: https://www.faradaycages.com/eavesdropping [Accessed: 16-Jul-2018].

[30] M. Kuhn, 'Compromising emanations: eavesdropping risks of computer displays', Technical Report UCAM-CL-TR-577, University of Cambridge, Cambridge, U.K., 2003.

[31] M. Vuagnoux and S. Pasini, 'An improved technique to discover compromising electromagnetic emanations', in *2010 IEEE International Symposium on Electromagnetic Compability*, Fort Lauderdale, FL, 2010.

[32] D. Genkin et al., 'RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis', in *CRYPTO*, 2014, pp. 444-461.

[33] L. Zhuang et al., 'Keyboard acoustic emanations revisited', *ACM Transactions on Information and System Security*, vol. 13, no. 1, pp. 3:1-3:26, 2009.

[34] M. Guri, 'BeatCoin: Leaking Private Keys from Air-Gapped Cryptocurrency Wallets', 2018. [Online]. Available: https://arxiv.org/pdf/1804.04014.pdf [Accessed: 16-Jul-2018].

[35] M. Guri et al., 'PowerHammer: Exfiltrating Data from Air-Gapped Computers through Power Lines', 2018. [Online]. Available: https://arxiv.org/pdf/1804.04014.pdf [Accessed: 16-Jul-2018].

[36] M. Guri et al., 'USBee: Air-gap covert-channel via electromagnetic emission from USB', in *14th Annual Conference on Privacy, Security and Trust*, Auckland, New Zealand, 2016.

[37] S. Stolfo, 'IBM and thumb drives: epoxy or beacons?', 2018. [Online]. Available: https://www.csoonline.com/article/3270971/physical-security/ibm-and-thumb-drives-epoxy-or-beacons.html [Accessed: 25-Sep-2018].

[38] Kanguru, 'Kanguru FlashTrust™ Secure Firmware USB Flash Drive'. [Online]. Available: https://www.kanguru.com/storage-accessories/kanguru-flashtrust-secure-firmware.shtml [Accessed: 21-Apr-2018].

[39] Kingston Digital, 'IronKey S1000 Secure USB Drive', 2018. [Online]. Available: https://www.kingston.com/datasheets/IKS1000_en.pdf [Accessed: 13-Aug-2018].

[40] C. Woodford, 'How loudspeakers work - Explain that Stuff', 2018. [Online]. Available: https://www.explainthatstuff.com/loudspeakers.html [Accessed: 25-Sep-2018].

[41] Consumer and Clinical Radiation Protection Bureau, 'Limits of Human Exposure to Radiofrequency Electromagnetic Energy in the Frequency Range from 3 kHz to 300 GHz', Safety Code 6, Health Canada, Ottawa, ON, 2015. [Online]. Available: https://www.canada.ca/content/dam/hc-sc/migration/hc-sc/ewh-semt/alt_formats/pdf/consult/_2014/safety_code_6-code_securite_6/final-finale-eng.pdf [Accessed: 12-Aug-2018].

[42] Dangerous Decibels, 'How do we hear?'. [Online]. Available: http://www.dangerousdecibels.org/virtualexhibit/2howdowehear.html [Accessed: 26-Apr-2018].

[43] B. Lawton, 'Damage to human hearing by airborne sound of very high frequency or ultrasonic frequency', Contract Research Report 343, Health & Safety Executive, London, U.K., 2001, p. 77. [Online]. Available: http://www.hse.gov.uk/research/crr_pdf/2001/crr01343.pdf [Accessed: 12-Aug-2018].

[44] B. Smagowska and M. Pawlaczyk-Łuszczyńska, 'Effects of Ultrasonic Noise on the Human Body - A Bibliographic Review', *International Journal of Occupational Safety and Ergonomics*, vol. 19, no. 2, pp. 195-202, 2013.

[45] L. Peterson and B. Davie, *Computer Networks: A Systems Approach*. Burlington, MA: Morgan Kaufmann, 2012, pp. 74-75.

[46] Electron18, 'Exchange interface Bell 202', 2010. [Online]. Available: http://www.softelectro.ru/bell202_en.html [Accessed: 25-Sep-2018].

[47] D. Lawyer, 'Modem-HOWTO: Appendix A: How Analog Modems Work', 2007. [Online]. Available: https://www.tldp.org/HOWTO/Modem-HOWTO-21.html [Accessed: 25-Sep-2018].

[48] S. Hochheiser, 'Telephone Transmission', 2015. [Online]. Available: https://ethw.org/Telephone_Transmission [Accessed: 25-Sep-2018].

[49] M. Pellegrini, 'Testing 56k Modems', 1998. [Online]. Available: https://www.evaluationengineering.com/testing-56k-modems [Accessed: 25-Sep-2018].

[50] L. Deshotels, 'Inaudible sound as a covert channel in mobile devices', in *8th USENIX Workshop on Offensive Technologies (WOOT '14)*, Berkeley, CA, 2014, p. 16.

[51] A. Dominguez, 'Highlights in the History of the Fourier Transform - IEEE Pulse', 2016. [Online]. Available: https://pulse.embs.org/january-2016/highlights-in-the-history-of-the-fourier-transform/ [Accessed: 27-Jul-2018].

[52] K. Banks, 'The Goertzel Algorithm | Embedded', 2002. [Online]. Available: http://www.embedded.com/design/configurable-systems/4024443/The-Goertzel-Algorithm [Accessed: 21-Apr-2018].

[53] J. Penketh, 'An Efficient Method for the Use of Overlapped Analysis', in *London Communications Symposium*, London, U.K., 2002. [Online]. Available: http://www.ee.ucl.ac.uk/lcs/previous/LCS2002/LCS043.pdf [Accessed: 27-Jul-2018].

[54] P. Mock, 'Add DTMF Generation and Decoding to DSP-μP Designs', Application Report SPRA168, Texas Instruments, 1989. [Online]. Available: http://www.ti.com/lit/an/spra168/spra168.pdf [Accessed: 25-Sep-2018].

[55] G. Goertzel, 'An Algorithm for the Evaluation of Finite Trigonometric Series', *The American Mathematical Monthly*, vol. 65, no. 1, pp. 34-35, 1958.

[56] D. Jones, 'Goertzel's Algorithm - Digital Signal Processing: A User's Guide - OpenStax CNX', 2006. [Online]. Available: https://cnx.org/contents/kw4ccwOo@5/Goertzel-s-Algorithm [Accessed: 25-Sep-2018].

[57] G. Schmer, 'DTMF Tone Generation and Detection: An Implementation Using the TMS320C54x', Application Report SPRA096A, Texas Instruments, 2000. [Online]. Available: http://www.ti.com/lit/an/spra096a/spra096a.pdf [Accessed: 25-Sep-2018].

[58] B. Walker, *Noise_Generator.py*, 2012. [Online]. Available: https://code.activestate.com/recipes/578350-platform-independent-white-noise-generator/download/1/ [Accessed: 1-Sep-2018].

[59] M. Guri et al., 'MOSQUITO: Covert Ultrasonic Transmissions between Two Air-Gapped Computers using Speaker-to-Speaker Communication', 2018. [Online]. Available: https://arxiv.org/pdf/1803.03422.pdf [Accessed: 29-Sep-2018].