Exploiting the IoT Through Network-based Covert Channels

THESIS

Kyle S. Harris, Captain, USAF

AFIT-ENG-MS-22-031

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-22-031

Exploiting the IoT Through Network-based Covert Channels

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

Kyle S. Harris, BS

Captain, USAF

March 24, 2022

AFIT-ENG-MS-22-031

Exploiting the IoT Through Network-based Covert Channels

THESIS

Kyle S. Harris, BS
Captain, USAF

Committee Membership:

Lt Col Wayne C. Henry, Ph.D.
Chair

Maj Richard Dill, Ph.D.
Member

Scott R. Graham, Ph.D.
Member

Abstract

Information leaks are a top concern to industry and government leaders. The Internet of Things (IoT) is a rapidly growing technology capable of sensing real-world events. IoT devices lack a common security standard and typically use lightweight security solutions, exposing the sensitive real-world data they gather. A practical method for exfiltrating data from these devices is by covert channel.

This research designs a novel IoT Covert Timing Channel (CTC) by encoding data within preexisting network information, namely ports or addresses. Seven different encoding methods are implemented between two IoT protocols, Transmission Control Protocol/Internet Protocol (TCP/IP) and ZigBee. The TCP/IP covert channel is created by mimicking a Ring smart doorbell and implemented using Amazon Web Services (AWS) servers to generate traffic. The ZigBee channel is built by copying a Philips Hue lighting system and executed on a Local Area Network (LAN). Additionally, the CTC can be implemented in two different modes: Stealth and Bandwidth. Performance is measured using throughput and detectability. The Stealth methods mimic legitimate traffic captures to make them difficult to detect while the Bandwidth methods forgo this approach for maximum throughput. Detection results are presented using four statistical-based detection tests: the Kolmogorov-Smirnov (KS) test, the Shape test, the Regularity test, and the Similarity test.

The Stealth results have a throughput of 4.61 bits per second (bps) for TCP/IP and 3.90 bps for ZigBee. They also evade detection tests. The Bandwidth methods average 81.7 Kbps for TCP/IP and 9.76 bps for ZigBee, but are evident in detection tests.

Acknowledgements

I would like to thank Lt Col Wayne Henry, my advisor, for his continued guidance and support throughout my thesis research. I also want to thank my parents for their continued love and encouragement. Lastly, I am grateful to my classmates for keeping me sane in the middle of this pandemic.

Kyle S. Harris

Table of Contents

Page

List of Figures

Figure Page

List of Tables

Exploiting the IoT Through Network-based Covert Channels

I. Introduction

1.1 Background and Motivation

According to the National Intelligence Council, the Internet of Things (IoT) is one of the top six technologies with potential impacts on US interests and national power [8]. The IoT was first proposed in 1999; however, it is still a rapidly evolving technology with approximately 30 billion devices today [9, 10]. These devices sense and collect information based on real-world events around them, making the data they collect particularly sensitive, especially for military use. These applications include enhanced base security, personnel accountability, automation, and data analytics [11].

The IoT's rapid evolution has led to several security issues. Since these devices often have limited resources, they must use lightweight protocols and security solutions. These limitations and the rapid iteration of devices have led to a lack of IoT security standards [12, 13]. Although the IoT can significantly increase the efficiency and capability of an organization, it is not without risk. The more IoT devices across the Department of Defense (DoD), the higher the risk, especially in data security. However, the same risks apply to competing nations. Understanding the vulnerabilities of IoT protocols and devices, how to exploit them, and how to secure them is a high priority. One practical method of exfiltrating data from these devices is via covert channels.

1.2 Hypothesis and Research Objectives

This thesis creates a novel Covert Timing Channel (CTC) hypothesizing that exfiltrating data from IoT devices through network-based encoding is more effective than by inter-packet delay-based methods. The effectiveness of each CTC is measured by its throughput and detectability. To properly test this hypothesis, the following research objectives are identified:

- Develop a CTC capable of encoding data within preexisting network-based information.
- Implement the CTC within commonly used IoT protocols.
- Optimize the IoT CTC to have higher throughput or lower detectability.
- Analyze and evaluate the system's detectability and throughput.

1.3 Approach

This thesis creates a novel CTC within two IoT communication protocols, Transmission Control Protocol/Internet Protocol (TCP/IP) and ZigBee. The TCP/IP implementation encodes data within any part of a socket: source ports, destination ports, source Internet Protocol (IP) addresses, or destination IP addresses. The ZigBee approach encodes data within the device's network address. Furthermore, these encoding methods are implemented in a *Stealth* mode or a *Bandwidth* mode. The Bandwidth mode focuses on sending traffic as quickly as possible and is designed to be implemented when few or no detection methods are present. The Stealth mode mimics a legitimate traffic baseline, making it difficult to detect. This baseline is taken from a single device for each protocol. A Ring smart doorbell is used for TCP/IP, and a Philips Hue system is implemented for ZigBee.

These CTCs are evaluated based on their detectability and throughput. Throughput is based on the number of encoded bits per second the channel can exfiltrate. Detectability is measured from the results of four statistical tests.

1.4 Research Contributions

To the researcher's knowledge, this is the first work that proposes encoding data the addresses and ports of IoT devices. This method is more likely to go undetected since CTCs typically encode data in predefined timing intervals. It also sends traffic faster since these timing intervals limit throughput. This approach can covertly exfiltrate or securely transmit sensitive data across IoT devices. Finally, the CTC is evaluated and compared with other network-based and IoT CTCs.

1.5 Thesis Overview

The remainder of this research is organized into four chapters. Chapter II provides background information on covert channels and the IoT, in addition to current research on IoT covert channels. Chapter III presents the experimental methodology used to develop and evaluate the system. Chapter IV presents and analyzes the experimental results. Finally, Chapter V summarizes the research, presents conclusions, and discusses directions for future work.

II. Background and Literature Review

2.1 Overview

This chapter presents background information used in this research. First, background on Internet of Things (IoT) applications, security, and commonly used protocols are explained. The following section discusses the background of covert channels, focusing on covert timing channels. Then, related research in covert channels is presented. Finally, the chapter is summarized.

2.2 Internet of Things

The IoT is a network of heterogeneous devices capable of detecting and perceiving real-world events occurring around them to enable the integration of the cyber and physical worlds [14]. IoT architecture is decomposed into three different layers, as shown in Figure 1: perception, network, and application layers [15]. The perception layer, also known as the sensing layer, consists of individual IoT devices. These devices contain sensors that are capable of detecting and perceiving real-world events occurring around them (e.g., a motion sensor or a camera) and transmitting that information through the network layer [16]. The network layer is responsible for transmitting data between each IoT device in the perception layer and its overarching function in the application layer. It consists of networking mediums, such as the Internet, cloud computing, and protocols used to transmit information between devices [17]. The application layer incorporates sensor data into the application [18]. It is also where the user interacts with the data.

IoT applications can vary widely, including healthcare, manufacturing, and home automation. Many of these technologies can also be applied to military use. As explained by Chiaramonte, et al. [11], security could be monitored from a centralized

point on the base via a sensor and drone network, and smart wearables can monitor and identify personnel. Kott, et al. [19], propose that IoT devices will soon be part of every aspect of military conflict due to the informational advantage they provide. These applications include battlefield vision, status and location sensors on weaponry, and wearable devices on warfighters. Additionally, because military IoT devices will become prevalent, adversaries will fight over IoT cyberspace terrain to control them in the physical world.

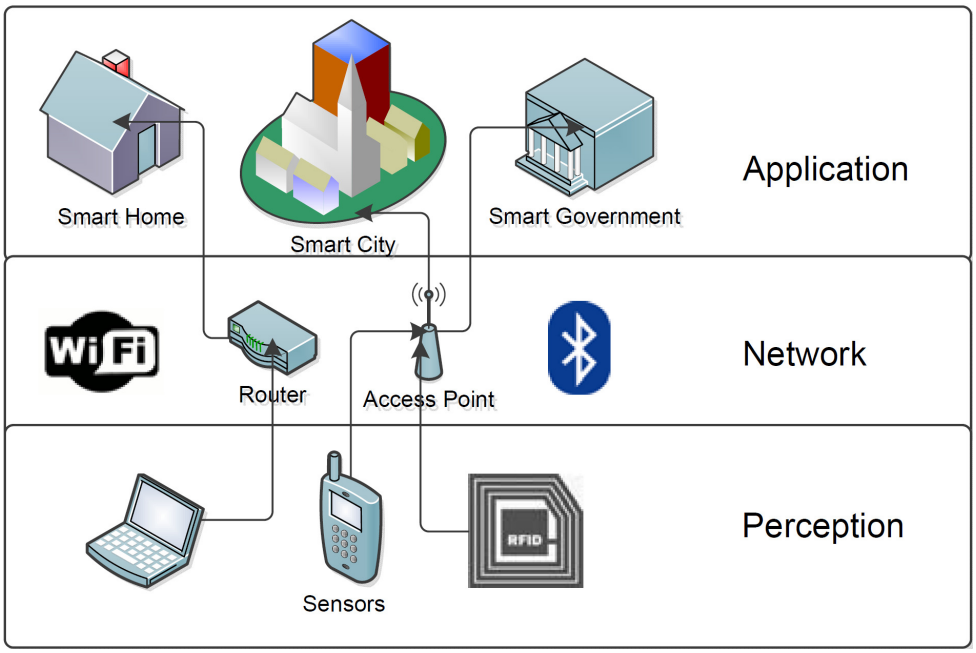

Figure 1: Classification of IoT applications [1]

2.2.1 Protocols

There are a wide variety of protocols to accommodate different types of IoT devices. This section focuses mainly on Transmission Control Protocol/Internet Protocol (TCP/IP) and ZigBee, the two communication protocols implemented in the covert channel.

Communication protocols enable and administer the transfer of information be-

tween systems. One of the most common IoT communication protocols, ZigBee, is a wireless, low-cost, low-power protocol built on the IEEE 802.15.4 standard for use in Wireless Sensor Networks [20]. It was developed by the Connectivity Standards Alliance (CSA), formerly known as the ZigBee Alliance, and is used in several applications, including security and home automation. As depicted in Figure 2, the ZigBee protocol stack is composed of four core layers: physical, Media Access Control (MAC), network, and application. Institute of Electrical and Electronics Engineers (IEEE) standards control the physical and MAC layers, and the CSA manages the application and network layers.

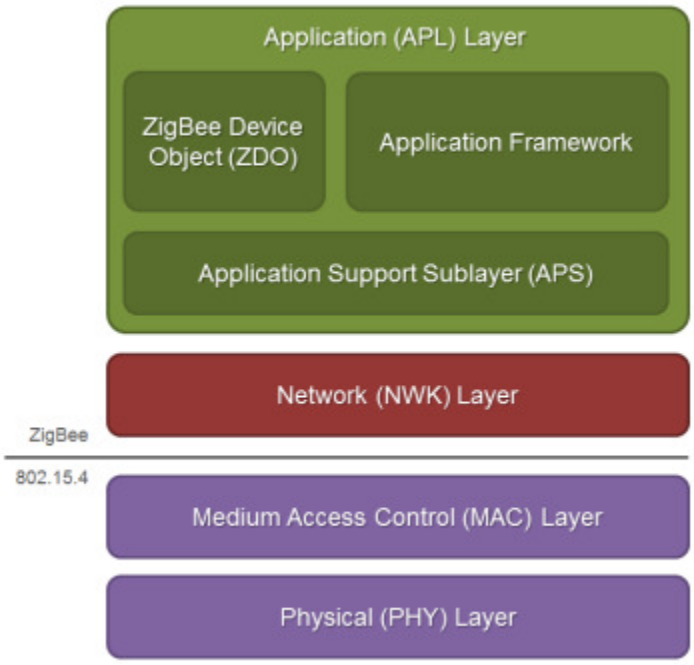

Figure 2: The ZigBee protocol stack [2]

ZigBee implements network addresses in the form of a four-digit (16-bit) hex number, as shown by the Wireshark capture in Figure 3. These addresses are present in both the MAC and network layers.

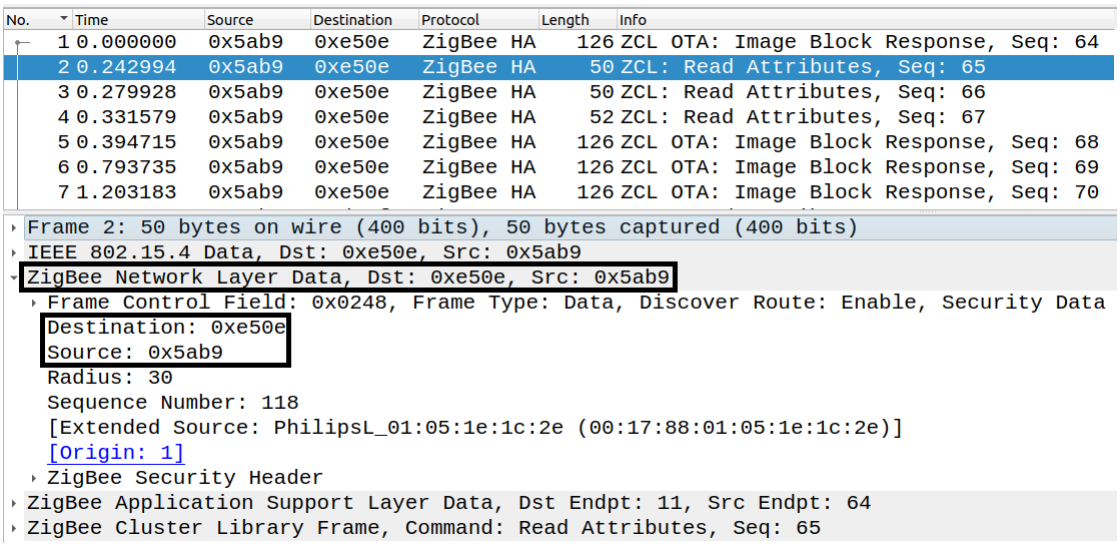

Figure 3: A Wireshark screenshot of a ZigBee packet

As shown in Figure 4, a ZigBee network typically consists of a single coordinator, multiple routers, and many end devices. The coordinator is the root of the entire network, connecting each router in the system [21]. Routers connect end devices to the network and pass messages between them. They are also commonly connected for network optimization. Devices can also be connected directly to the coordinator in smaller networks, like those present in this research.

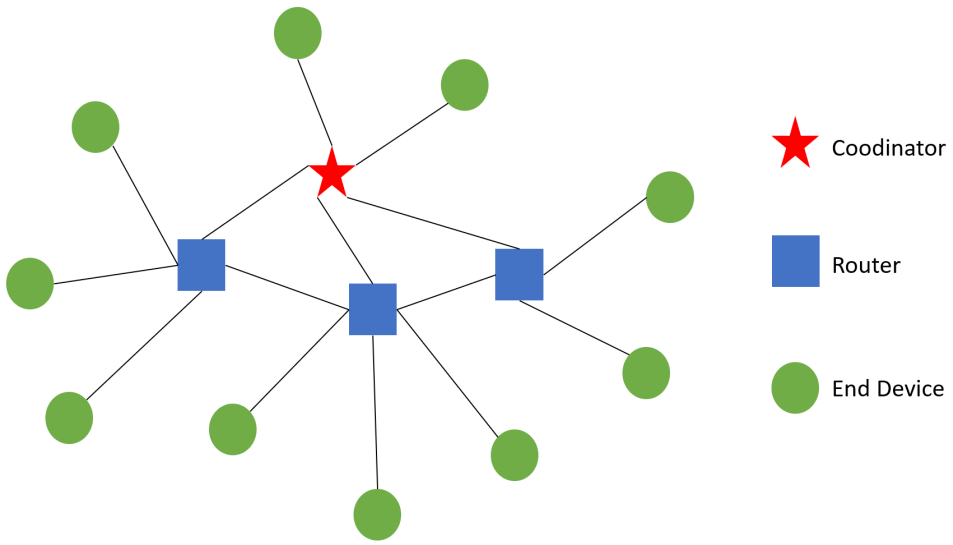

Figure 4: An example ZigBee network layout

Most IoT devices focus on conserving resources and use lightweight protocols like ZigBee and Bluetooth Low Energy (BLE). However, many IoT applications still use the traditional 802.11 Wi-Fi standards. In 2017, IEEE introduced 802.11ah, known as Wi-Fi HaLow, to be more compatible with other infrastructure and applications due to Wi-Fi's prevalence. Table 1 compares the baseline specifications of these protocols. Wi-Fi HaLow, ZigBee, and BLE have lower power consumption and bandwidth than traditional Wi-Fi. Sacrificing bandwidth for IoT applications is acceptable because they typically do not require high throughput. Wi-Fi HaLow also has similar power consumption to BLE and ZigBee, but with a much greater range [22].

Table 1: A comparison of common IoT communication protocols [6]

Feature	**IEEE 802.11 (n/ac)**	**IEEE 802.11ah**	**ZigBee/802.15.4e**	**BLE**
Frequency band (GHz)	Unlicensed 2.4, 5 GHz	Unlicensed 900 MHz	Unlicensed 868/915 MHz 2.4 GHz	Unlicensed 2.4 GHz
Data Rate	6.5–6933 Mbps	150 kbps–346 Mbps	<250 kbps	<1 Mbps
Coverage range	<200 m	<1.5 km	<100 m	<50 m
Power consumption	Medium	Low	Low	Low
Number of devices supported	2007	8000	65,000	Unlimited *

* BLE supports an unlimited number of devices, this depends on the configured address space.

This research focuses primarily on ZigBee and Wi-Fi because they are two of the most common IoT protocols; however, it is still relevant to briefly discuss other common IoT communication protocols:

- **BLE:** One of the most common IoT communication protocols. This protocol features an extended range, lower latency, and lower power requirements than traditional Bluetooth. It is commonly used for communication in mobile devices, wearables, and vehicles [23, 24].

- **Long-Term Evolution Advanced (LTE-A):** A standard wireless communication protocol originally designed for mobile phones; it has high throughput, long-distance coverage, and a high power requirement [22].

- **IPv6 over Low power Wireless Personal Area Network (6LoWPAN):** A network layer protocol designed to implement IPv6 more efficiently by compressing packet headers. It was originally designed for use with IEEE 802.15.4, but can also be implemented on top of other protocols, like IEEE 802.11ah and BLE, to efficiently implement Internet Protocol (IP) compatibility [22, 23, 25].

2.2.2 Security

The IoT is inherently insecure due to the wide variety of devices and rapid evolution without a universal security standard [12, 13]. Additionally, it primarily consists of embedded devices, which cannot support standard, complex security solutions [26]. These devices are often left unattended, making them vulnerable to physical tampering. Since IoT devices are almost entirely wireless, eavesdropping on traffic, even when it is encrypted, is also a privacy concern [27].

One example of improper security is the use of default symmetric keys in IoT devices [28]. Since the identical key exists in every device of the same model, they are easily found online. Many devices, including popular, high-end devices like the Philips Hue smart bulb, suffer from this vulnerability. Symmetric keys are often used because of their light overhead, making them easier to implement in IoT devices. However, they also enable an attacker to sniff and decrypt setup traffic. As shown in Figure 5, the IoT device requests an updated encryption key from an external server. Since this traffic is encrypted using the default, publicly available symmetric key, the attacker can decrypt it. Furthermore, this traffic contains the new transport key sent by the server, which encrypts all remaining traffic. With the new key in clear text, the attacker can decrypt all remaining traffic. This method could enable a covert receiver to obtain data by sniffing nearby traffic, eliminating the need to send data outside the host network.

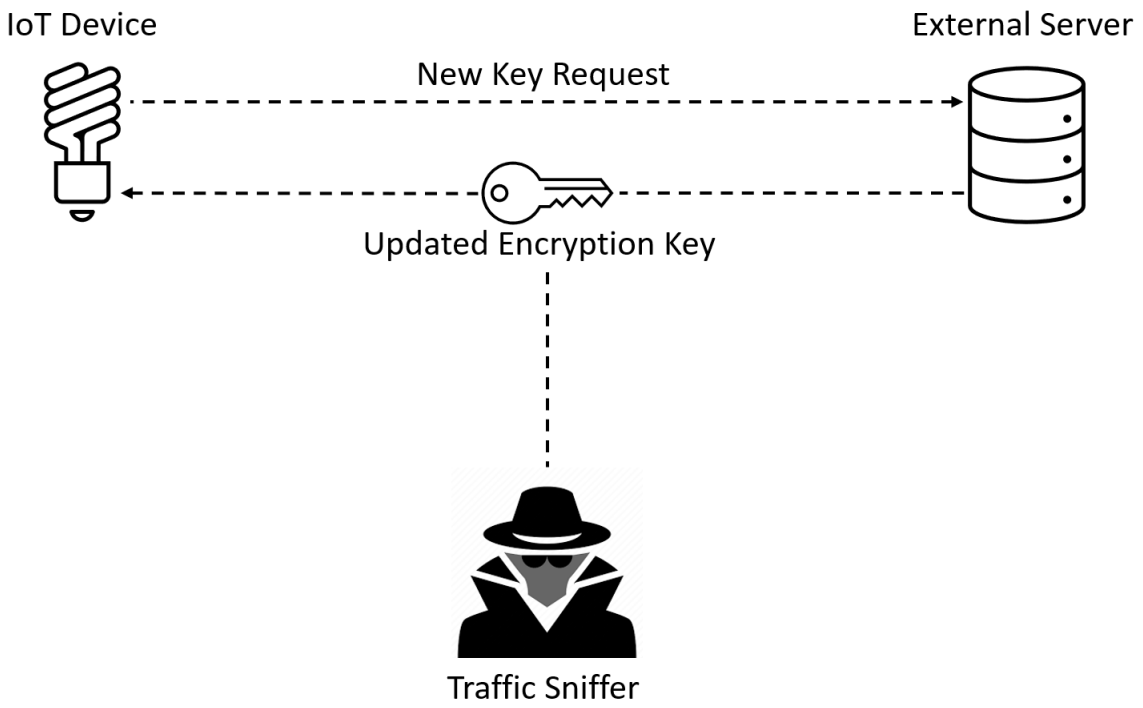

Figure 5: An example of encryption key interception

2.3 Covert Channels

Information leaks are a top concern to industry and government leaders [29]. A covert channel secretly sends information across a network to an outside receiver [30]. Therefore, the vulnerability of all computer systems to this type of exploit is of great concern to cyber security.

Covert channels are a method of clandestine communication that have existed in computer systems since they were first introduced by Lampson in 1973 [31]. They involve one or more covert senders transmitting information to covert receivers. For this communication to take place, there must be a resource that both the receiver and sender share and can access, directly or indirectly, and a predetermined pattern with which they can communicate [32]. Covert channels consist of two main categories, Covert Storage Channels (CSCs) and Covert Timing Channels (CTCs).

2.3.1 Covert Storage Channels

CSCs are where the covert sender will write data to a location, and the receiver will read it [33]. A typical CSC manipulates unused fields within packet headers (e.g., turning the flag on equates to "1" and turning it off equates to "0") [34]. Another straightforward method appends data to a message. Despite the ability to easily transfer large amounts of information through CSCs, most of them are relatively simple to detect.

2.3.2 Covert Timing Channels

CTCs modulate various system resources, such as intentional inter-packet delays or retransmissions, to transmit information between a sender and receiver [32]. CTCs can be further divided into passive or active. Passive CTCs modify the timing of existing, legitimate traffic to encode information. Active CTCs generate their own traffic that attempts to mimic legitimate traffic [35]. Compared to CSCs, CTCs typically have higher throughput but are harder to detect. Common CTCs are listed below:

- **Inter-Packet Delay (IPD):** Packets are sent at two different, distinguishable times. When the time delay between packets is closer to the first value, it represents an encoded zero. When the inter-packet delay is closer to the second value, it is decoded as a one [36].
- **Packet Reordering:** Packets are intentionally transmitted out of order. The CTC modifies packet sequence numbers and each possible sequence number permutation is used to encode information [36, 37].

2.3.3 Covert Channels with Port and IP Address Encoding

Hovhannisyan, et al. [38] propose a novel CTC using the destination IP addresses or ports to encode information. In this method, one port or IP address represents an encoded zero, and the other represents an encoded one. This approach does not introduce artificial delays, which are present in on/off timing channels and inter-packet delay-based CTCs. The result is a covert channel with significantly higher bandwidth that can evade most common detection techniques. This covert channel depends on packet arrival order to determine encoded information. If a packet arrives out of order, it could change the result of many other following packets. To circumvent this, the authors insert a slight inter-packet delay (approximately 1 ms) whenever there is a change in the transmitting route (i.e., a change from zero to one or visa-versa).

The research by Gimbi, et al. [39] encodes information within transport layer ports only using source ports. It presents a transmission scheme capable of transmitting up to 16 bits per message by encoding information in the port number. A redundancy check improved robustness while still having a throughput of eight bits of data per message. This solution can circumvent errors caused when a port is already in use by another process or when another process sends traffic to the device where the covert receiver is present, lowering the error rate from around 3% to 0.02%.

2.3.4 Covert Channels in the Internet of Things

Velinov, et al. [3] develop and analyze thirteen CSCs and CTCs within the Message Queuing Telemetry Transport (MQTT) protocol. This protocol transmits information between clients by publishing topics, which can be subscribed to via an intermediary broker. The authors classify their covert channels into Direct Covert Channel (DCC) and Indirect Covert Channel (ICC). As shown in Figure 6, clients

communicate directly with the broker in DCCs. ICCs are where a client communicates with another client using the broker as a medium, as depicted in Figure 7.

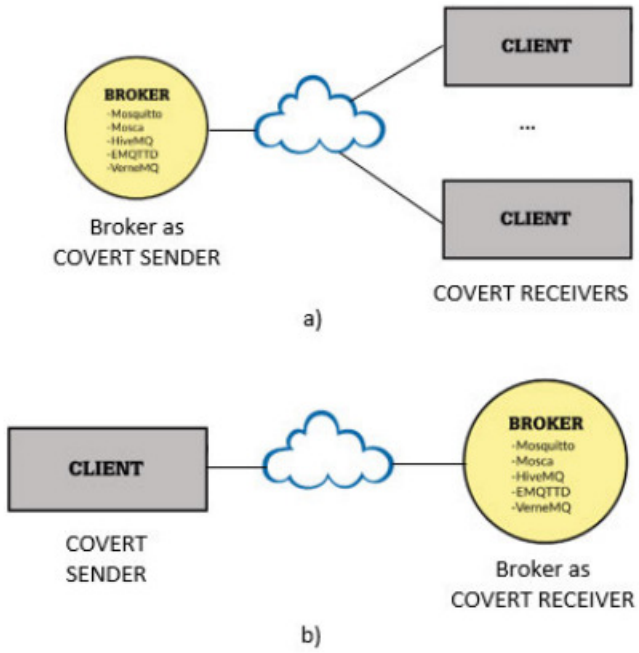

Figure 6: A DCC model with: a) The broker as the covert sender and b) The broker as the covert receiver [3]

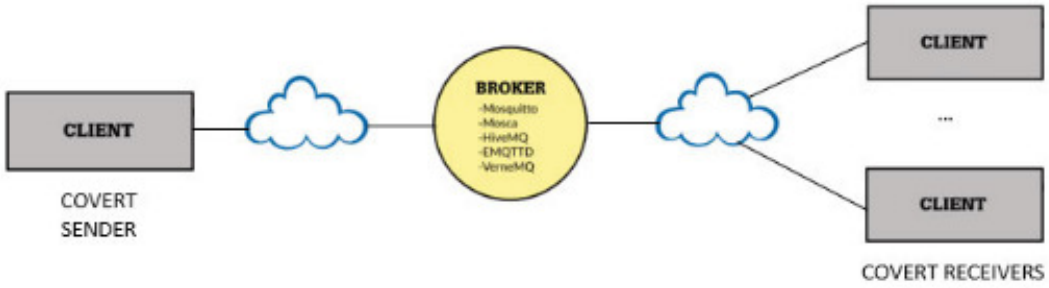

Figure 7: An ICC model with one client as the covert sender and other clients as covert receivers [3]

Two of their covert channels are similar to the method proposed in this work. The first publishes a topic to the broker, which the receiver subscribes to, transmitting information based on a predetermined pattern of topic names. The second publishes updates to topics the receiver subscribed to. Here, an encoded zero is represented by

the lack of a status update, and the presence of a status update represents an encoded one. Furthermore, these covert channels can mimic actual traffic using legitimate topic names to create a stealthy covert channel that a warden may not detect.

Tan, et al. [4] evaluate CTCs using IoT devices on (4G/5G) mobile networks. They contrast this traffic with IoT devices on Wi-Fi and analyze various CTCs on both networks. One case presented by the authors, shown in Figure 8, measures the jitter (i.e., variable delay) within a Voice over Internet Protocol (VoIP) call across both networks. The results showed that 85% of Wi-Fi traffic and 99% of mobile traffic was within a range of +/-10 ms. Mobile networks' high speed and reliability make adding a distinguishable delay problematic since they would not go unnoticed by an observer. When the delay between packets is not large enough, the covert receiver may misinterpret the bit and skew the message. Therefore, the capacity for delay-based CTCs within mobile networks has become significantly less probable than over Wi-Fi.

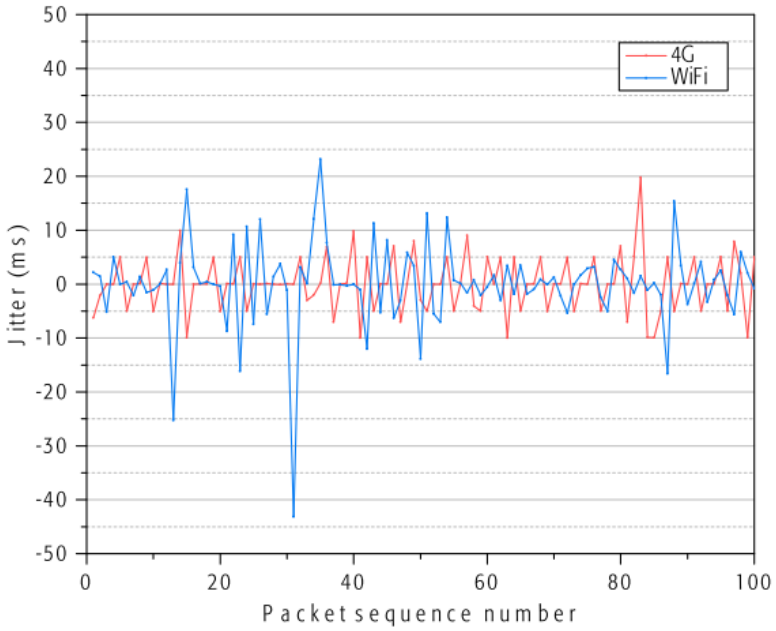

Figure 8: The jitter of VoIP traffic across a 4G mobile network and Wi-Fi [4]

Cabaj, et al. [40] discuss the application of Distributed Network Covert Channel (DNCC) within the IoT by designing numerous existing covert channels across varying devices, ports, and protocols. The goal behind combining all of these ordinarily independent covert channels into a single DNCC is to gain a high-level of covertness while maintaining meaningful throughput. If many covert channels all transmit a small amount of data, it enables each process to transmit information below the probable detection threshold, while still having meaningful throughput overall. Moreover, this approach is relevant to IoT devices since they transmit less traffic than standard networked devices. A large amount of traffic could negatively impact the covertness of an IoT covert channel.

There are also current works in IoT CTCs that are not as closely related, but still relevant. Mileva, et al. [41] research network covert channels in Constrained Application Protocol (CoAP). The authors propose encoding data within token and message ID numbers, file request methods, and message payloads. Alcaraz, et al. [42] address covert channels within the industrial IoT by exploiting Modbus/TCP. The research introduces a passive and active method of passing data. To passively exfiltrate data, the covert channel creates patterns within legitimate system messages such as measurements or alarms. When actively manipulating data, the channel creates fake Modbus/TCP queries and appends a portion of the covert message to it. In a continuing work, Mileva, et al. [43] create additional covert channels, focusing on the newest version of MQTT, v5.0. The authors analyze the covertness of their channels by using a compressiblity score. This metric divides the packet's size by its compressed size to determine a ratio. A baseline is determined for legitimate traffic and any traffic outside of the range of acceptance may have additional information encoded within it.

2.3.5 Related Works Comparison

Table 2 compares and contrasts all previously discussed covert channel works with this thesis. The table compares the type of covert channel researched, either CSC or CTC, and if it is within the IoT. It also denotes if the covert channel encodes data within network or transport layer protocols and if it is designed to be implemented simultaneously across multiple covert senders and receivers.

Gimbi, et al. and Hovhannisyan, et al. present encoding options within network or transport layer protocols, namely TCP/IP. Mileva, et al. and Velinov, et al. discuss CSCs and CTCs in CoAP and MQTT, two application-level IoT protocols. Tan, et al. implement IPD-based CTCs within 4G/5G IoT devices. Alcaraz, et al., develop covert channels in the industrial IoT using Modbus/TCP. Cabaj, et al. create a DNCC across several IoT protocols. This research implements network and transport encoding across multiple devices within the IoT.

Table 2: Summary of Related Research

Author	**CSC**	**CTC**	**Network/ Transport Encoding**	**Multiple Devices**	**IoT**
Gimbi, et al. (2012) [39]		X	X		
Hovhannisyan, et al. (2015) [38]		X	X	X	
Mileva, et al. (2018) [41]	X	X			X
Tan, et al. (2018) [4]		X			X
Alcaraz, et al. (2019) [42]	X	X	X		X
Velinov, et al. (2019) [3]	X	X			X
Cabaj, et al. (2020) [40]	X	X		X	X
Mileva, et al. (2021) [43]	X	X			X
Harris (2022)		X	X	X	X

2.4 Covert Timing Channel Detection Methods

Covert timing channel detection methods in this research analyze timing data of the inter-packet delays within network traffic. These tests analyze mean, variance,

distribution, and correlations in traffic data.

2.4.1 Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (KS) test is used to determine if one set of data matches the distribution of another. When implemented in covert channels detection, it compares covert traffic to legitimate traffic baseline [44]. The null hypothesis in this test is that the distributions are identical. The empirical Cumulative Distribution Functions (eCDFs) are plotted to evaluate this hypothesis, and the greatest vertical distance between them is calculated:

$$KSTEST = max|S_1(x) - S_2(x)| \tag{1}$$

where $S_1(x)$ and $S_2(x)$ are the eCDFs of the legitimate and covert traffic samples[45]. This value is the KS statistic. Additionally, the test produces a p-value measuring the likelihood that the distributions are different. As shown in Figure 9, the lower the p-value, the higher the certainty that the samples differ. A p-value lower than 0.10 is considered suggestive or convincing, while a higher p-value does not suggest a difference. The result is determined by comparing the KS test statistic and the p-value; if the test statistic is less than the p-value, the null hypothesis cannot be rejected [46].

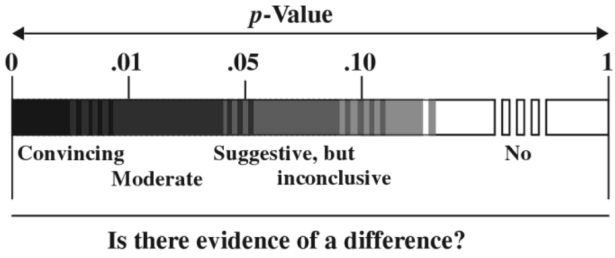

Figure 9: p-Value interpretation [5]

2.4.2 Shape Test

The Shape test can detect CTCs by measuring traffic distribution. It estimates the probability of a CTC that fluctuates encoded bits between two distinct values by ordering and binning all the IPDs. The probability that a covert channel exists $\mathrm{P_{CovChan}}$ is:

$$P_{\mathrm{CovChan}} = 1 - \frac{C_{\mu}}{C_{\mathrm{max}}} \tag{2}$$

where C_{μ} is the packet count at the mean IPD and C_{max} is the maximum packet count [47]. A CTC fluctuating between two IPD values will have two bins with a large packet count and a low volume at the mean between them. This causes the ratio in Equation 2 to approach zero and the probability of a CTC to approach one.

2.4.3 Regularity Test

The Regularity test quantifies correlations in the data. It bins IPDs based on population size (e.g., each bin has 100 values) and calculates the pairwise difference for each consecutive bin i and j [7]. The standard deviation of all the differences is taken. This value represents the regularity score, as shown in the following equation:

$$regularity = STDEV\left(\frac{|\sigma_{\mathrm{i}} - \sigma_{\mathrm{j}}|}{\sigma_{\mathrm{i}}}, i < j, \forall i, j\right) \tag{3}$$

where σ_{i} and σ_{j} are the standard deviations of consecutive bins. The regularity score is calculated for each CTC trial and then compared to the legitimate traffic baseline. If the covert traffic has a regularity score within 10% of the legitimate traffic's regularity score, it passes the test [45].

2.4.4 Similarity Test

The Similarity test places IPDs in ascending order and compares the relative differences between each consecutive IPD. It groups these results based on the size of the difference to determine the ϵ-*Similarity Scores* [48].

The ϵ-*Similarity Score* thresholds represent the relative percentage difference between each consecutive IPD and the value depicts the amount of traffic that has less than that difference. Example Similarity test results are shown in Table 3. In this example, 34.87% of the covert traffic has less than a 0.5% difference between consecutive IPDs. The covert and legitimate results in each group are then compared and if they fall within an accepted margin, experimentally defined by the author as 10%, that group passes. If a majority of the groups pass, the covert traffic cannot be considered different from the legitimate sample.

Table 3: Example Similarity test results [7]

Encoding Method	ϵ-***Similarity Score***						
	0.005	**0.008**	**0.01**	**0.02**	**0.03**	**0.1**	**> 0.1**
Covert Traffic	34.87	46.37	51.83	67.58	76.19	90.65	9.35
Legitimate Traffic	39.92	52.83	58.58	72.29	79.94	91.85	8.15

2.5 Background Summary

This chapter presents background information on the IoT and covert channels. It emphasizes security issues and standard protocols in the IoT necessary to develop a novel covert channel. Next, covert channels and CTC detection techniques are discussed. The chapter concludes with a survey on related research efforts in covert channels, focusing on IoT, and contrasts these works with this research.

III. Methodology

3.1 Overview

This chapter presents the methodology used to develop and evaluate a network-based Internet of Things (IoT) Covert Timing Channel (CTC). First the objectives of this research are discussed, followed by the approach used to achieve them. Next, the chapter defines the System Under Test (SUT). Then, the performance metrics, detectability and throughput, are presented, followed by the experiment factors and the system parameters. Finally, the design of the experiment is detailed, and the chapter is summarized.

3.2 Objective

This research aims to develop a CTC using preexisting network data to exfiltrate sensitive information stealthily from an adversary network or covertly send files across an insecure network. This covert channel will focus on IoT devices due to the sensitive, real-world data they often transmit as well as their proliferation. Furthermore, the CTC will have two modes: one that is stealthy and one focused on throughput. The CTC is evaluated by measuring throughput and detectability.

3.3 Approach

The approach taken accomplishes these objectives by using addresses or ports to encode information. The system implements seven encoding methods across two IoT communication protocols, Transmission Control Protocol/Internet Protocol (TCP/IP) and ZigBee. Additionally, these encoding methods are divided into two different modes: Stealth and Bandwidth. The Stealth mode mimics the pattern of legitimate

traffic, making it more difficult to detect. The Bandwidth mode sends traffic as fast as a device is capable and is meant for use on an unmonitored network.

The encoding process is shown in Figure 10. The covert sender begins by taking a secret file and breaking down the individual American Standard Code for Information Interchange (ASCII) bytes into binary. Then, each binary bit is encoded into one of the packet's network-based fields (i.e., address or port) and sent to the covert receiver(s), with each packet sent encoding one bit. This process must be pre-agreed upon between the covert sender(s) and covert receiver(s) before the secret file transmission takes place. This experiment uses precisely two sources to encode data (i.e., two different ports or two different addresses). Although more can be used, two is sufficient for proof of concept.

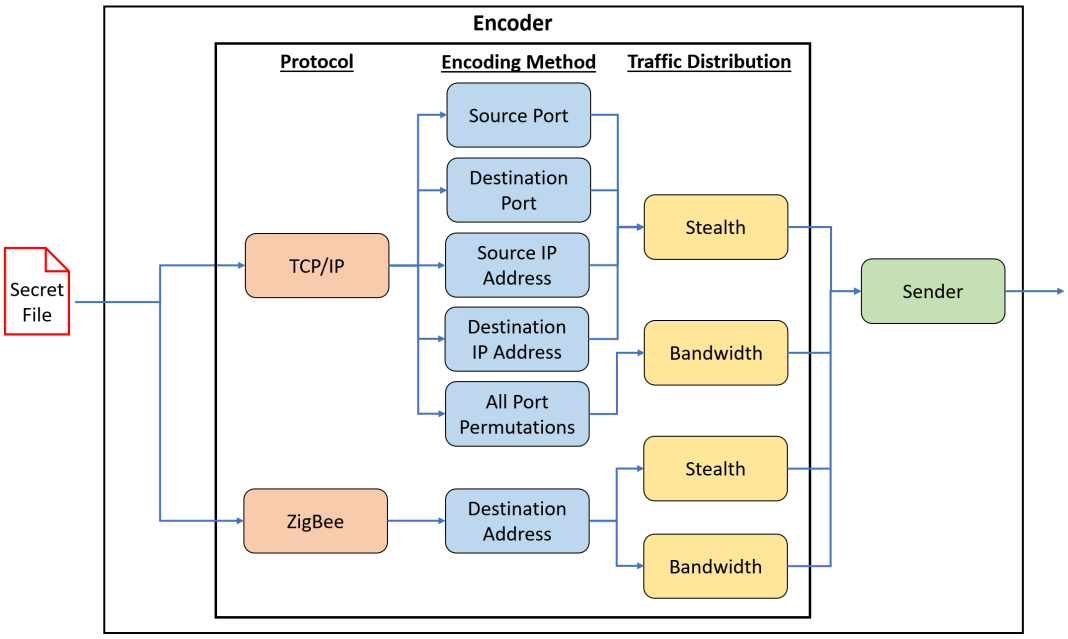

Figure 10: The framework for the encoder design

Once the encoded packets have reached the covert receiver, the decoding process begins. This process is shown in Figure 11. The receiver examines either the port or address header field, depending on the encoding method used. It also checks the

sequence number of each packet to ensure they are in the correct order. Once a full byte of consecutive characters is collected, the bits are combined, error-checked via a parity bit, and decoded based on their respective ASCII value. This process repeats until all encoded traffic from the covert sender(s) is transmitted.

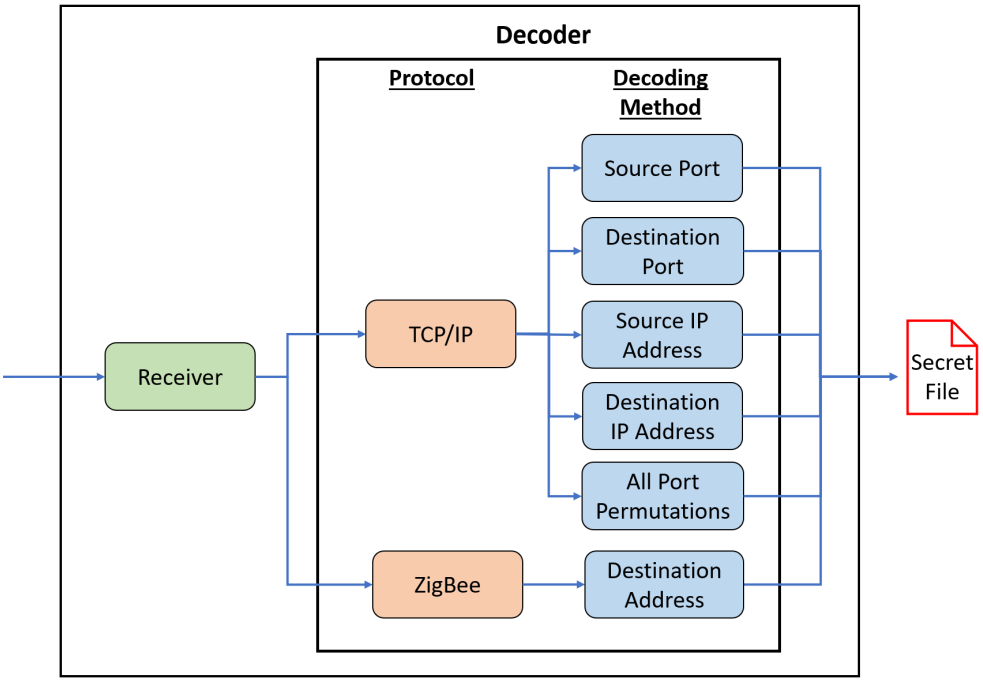

Figure 11: The framework for the decoder design

Figure 12 shows the encoder transmitting a single character to the decoder using the destination port encoding method. The encoder breaks down the ASCII character "A" in the secret file into its binary representation, "01000001." It transmits every zero to port "Y" and each one to port "Z." The decoder interprets each received bit on port "Y" as zero and on port "Z" as one. Once the decoder receives all eight packets, it decodes the bits "01000001" into the ASCII character "A."

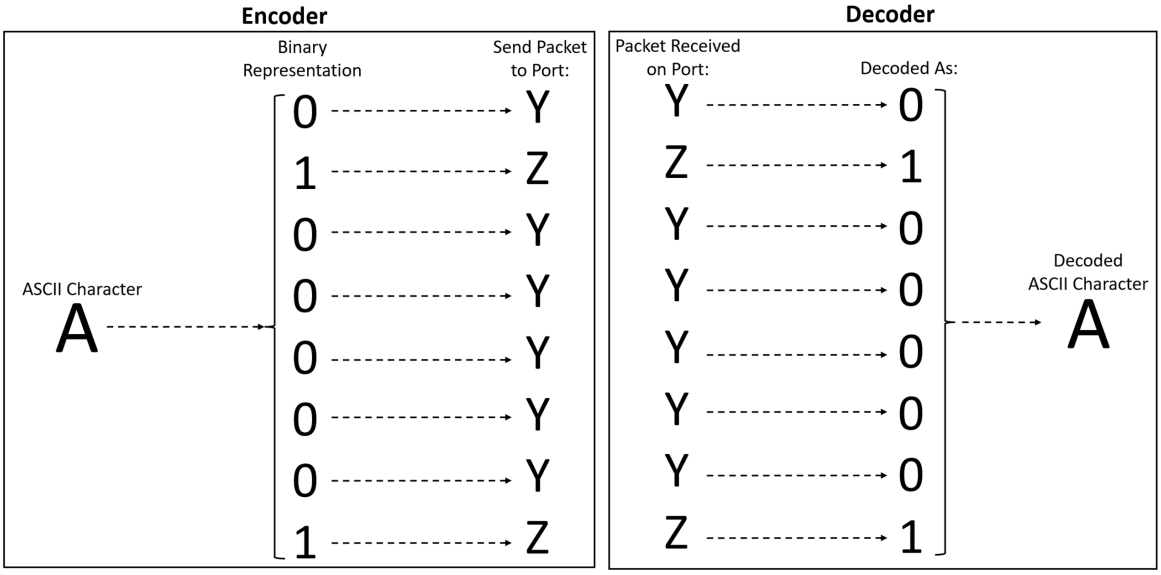

Figure 12: An example of encoding and decoding an ASCII character

3.4 Assumptions and Limitations

The following assumptions and limitations are made throughout this research:

- Root access is assumed. This is a pre-requisite for most types of covert channel research; however, that task is out of scope for this research.
- Only one type of device is used as a traffic baseline for each protocol. Therefore, the results are heavily influenced by the traffic baseline and may vary between devices.
- Outbound traffic from a covert sender will be monitored and analyzed per device. Accordingly, this is where all traffic baselines and captures are taken.

3.5 System Definition

The SUT is the IoT network-based covert channel, shown in Figure 13. The system comprises the Encoding Method, the Component Under Test (CUT), the exploited

IoT devices, the computer used to run the experiments, and the network where all of these components exist.

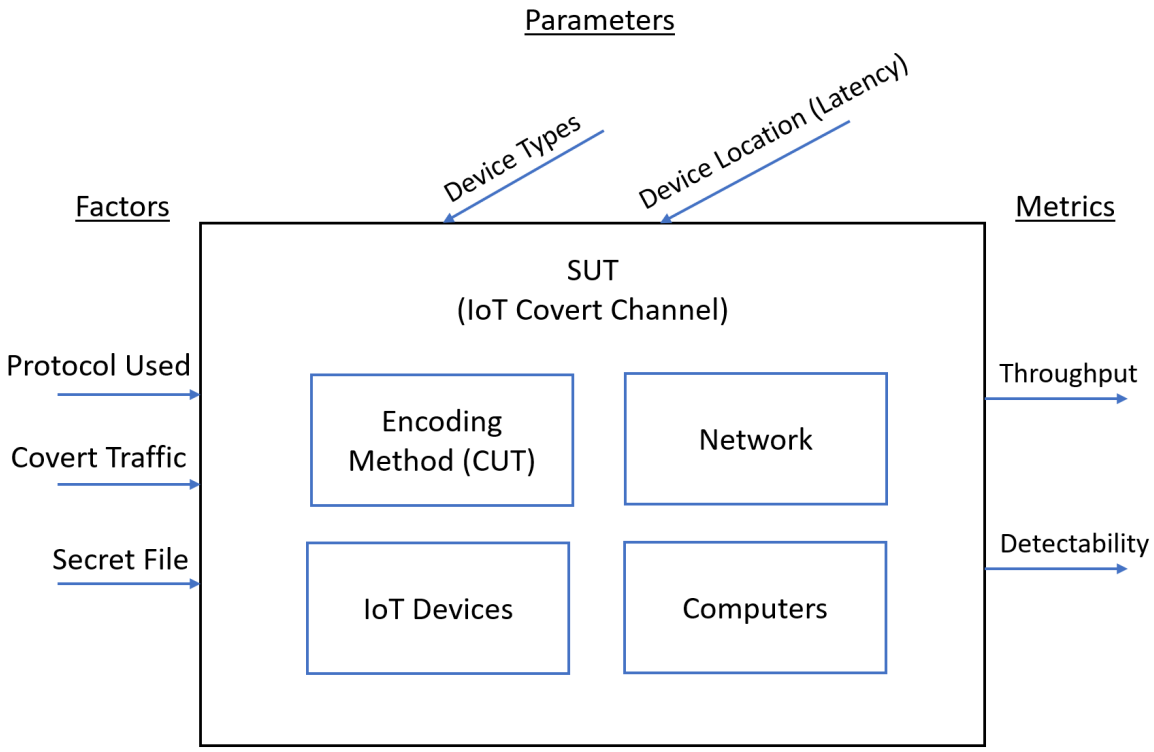

Figure 13: System Under Test: The network-based IoT CTC system

3.5.1 Metrics

The following metrics are used to measure and evaluate the results of the experiment.

- **Throughput:** Determined by measuring the number of bits the covert receiver(s) get from the covert sender(s) per second. Since these experiments use two sources (i.e., two different ports or two different Internet Protocol (IP) addresses) to encode data, each packet sent will equal one bit of data. The only exception is the TCP/IP Bandwidth method, which encodes 32 bits per packet.

- **Detectability:** Four statistical tests designed to detect covert timing channels estimate the system's detectability. The Shape test estimates the probability of a CTC's existence by calculating a ratio of the packet count at the mean IPD value over the packet count of the maximum Inter-Packet Delay (IPD) value. However, the Shape test does not differentiate well between CTC and cover traffic, possibly skewing the results. The Kolmogorov-Smirnov test is used to determine if a data sample (our covert channel traffic) fits a given distribution (a legitimate traffic baseline) [46]. The Regularity test measures the variance of IPDs by separating traffic into groups and calculating the relative difference between the standard deviations of each neighboring group. It then combines these values into a new set, and the standard deviation is retaken to calculate the traffic's regularity. A difference in regularity score of within 10% is considered legitimate [45]. The Similarity test computes the relative difference between neighboring IPDs to determine if a covert channel is present.

3.5.2 Factors

The factors below will be varied throughout the experiment.

- **Communication Protocol:** The encoding method depends on the protocol used, TCP/IP or ZigBee. For TCP/IP, the encoding methods include source ports, destination ports, source IP addresses, and destination IP addresses. For ZigBee, the destination address of the Coordinator's outgoing traffic encodes information.

- **Secret File:** The secret file sent via the covert channel is a random series of bits. Furthermore, the same file is used for each encoding method's respective trial.

- **Traffic Distribution:** Every encoding method is tested with each of the two different throughput distributions: Stealth and Bandwidth.

3.5.3 Parameters

The following parameters are kept constant throughout the experiment.

- **Device Type:**

  Amazon Web Services (AWS) Instances: Ubuntu Server 20.04 Long-Term Service (LTS) w/ t2.medium (2x Virtual Central Processing Units (vCPUs), 4 GB Random Access Memory (RAM)) instance

  ZigBee Coordinator: Raspberry Pi 4 Model B (8 GB), Raspberry Pi Operating System (OS) 5.10 with Texas Instruments CC2652R1 2.4-GHz multi-protocol wireless microcontroller

  Smart bulbs: Philips Hue white ambiance E26 with Bluetooth

  Sniffer Machine for ZigBee experiments: Dell Latitude 7400: Intel Core i7-8665U CPU @ 1.90GhZ 16 GB RAM with Texas Instruments CC2531 2.4-GHz multi-protocol wireless microcontroller

- **Device Location:** The TCP/IP experiments employ AWS servers in Virginia and California as the covert sender and receiver devices. The ZigBee devices must be monitored locally since they cannot be plugged into a remote network. Therefore, an isolated Local Area Network (LAN) serves as the network for the covert sender and an out-of-band sniffer machine acts as the cover receiver.

3.6 Experimental Design

The experiment measures the throughput and detectability of each configuration of the covert channel. Since there are seven encoding methods, seven trials are required

to obtain the necessary measurements. This research conducts three repetitions each to achieve high confidence, resulting in a total of 21 trials throughout the experiment.

The experiment uses covert sender and covert receiver devices at geographically separated locations. The covert sender devices will be local, physical devices (ZigBee) or virtualized AWS instances. All covert receiver devices will be AWS instances. The intent behind this design is to simulate a live IoT device as similarly as possible. AWS provides the hardware needed to replicate a realistic device and networking state as closely as possible without a separate physical network. Each AWS instance will run the closest hardware and OS to a typical IoT device, Ubuntu Server 20.04 LTS with 4 GB RAM. IoT devices are simulated instead of using the actual devices due to issues with gaining root access. Instead, this research sniffs traffic from real IoT devices and mimics that on the simulated devices to get as close as possible.

The covert sender and receiver run Python scripts on each respective machine. The sender script takes a file as input, decodes the ASCII values for each byte to binary, and creates the packet for the covert receiver using Python's Scapy plugin. Then the sender script transmits packets to the covert receiver at a rate determined by the mode selected, either Stealth or Bandwidth. Concurrently, the script on the covert receiver listens for and saves inbound packets from the sender to a .pcap file. The encoding scheme decodes each packet into a binary one or zero, combines them into bytes, error-checks them, and converts them into ASCII characters. This process repeats until the script stops receiving incoming packets from the covert sender. The encoding methods used to create the CTC packets vary depending on the protocol used, TCP/IP or ZigBee.

3.6.1 TCP/IP-based Covert Communication

The network-based IoT CTC implements five different methods to encode data in TCP/IP packets. Figure 14 depicts the experimental methodology. The experiment evaluates throughput and detectability for all TCP/IP configurations across 15 different trials. The covert sender(s) open two different sockets for each trial to transmit encoded information with the covert receiver(s).

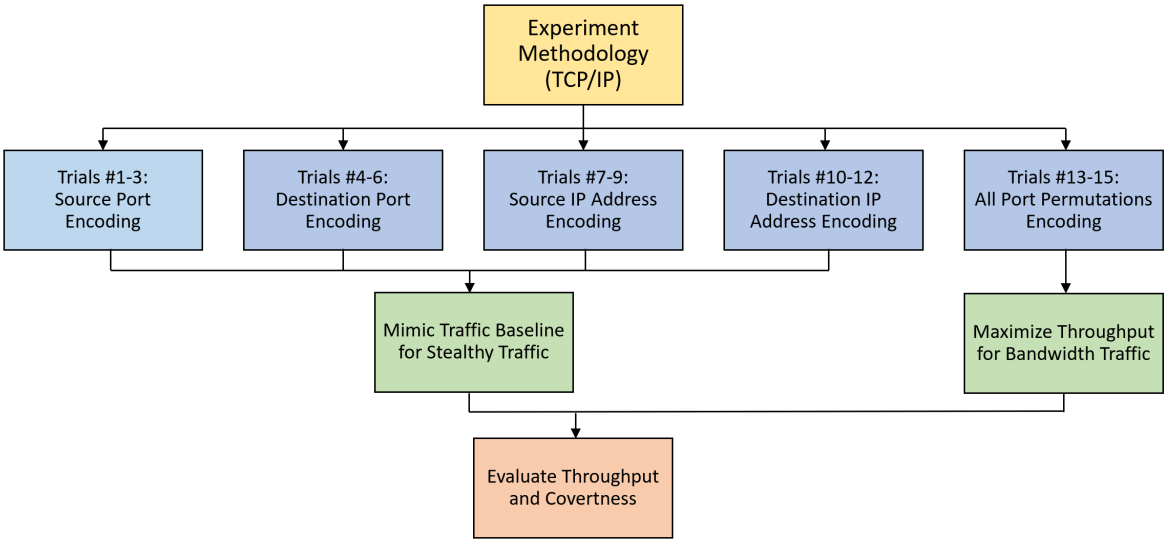

Figure 14: TCP/IP based CTC experimental methodology

The network diagram for the TCP/IP CTC is shown in Figure 15. It consists of two covert senders, "A" and "B," at one AWS server and two covert receivers, "C" and "D," at another. Each TCP/IP encoding method exfiltrates data from the senders at the first server to the receivers at the second.

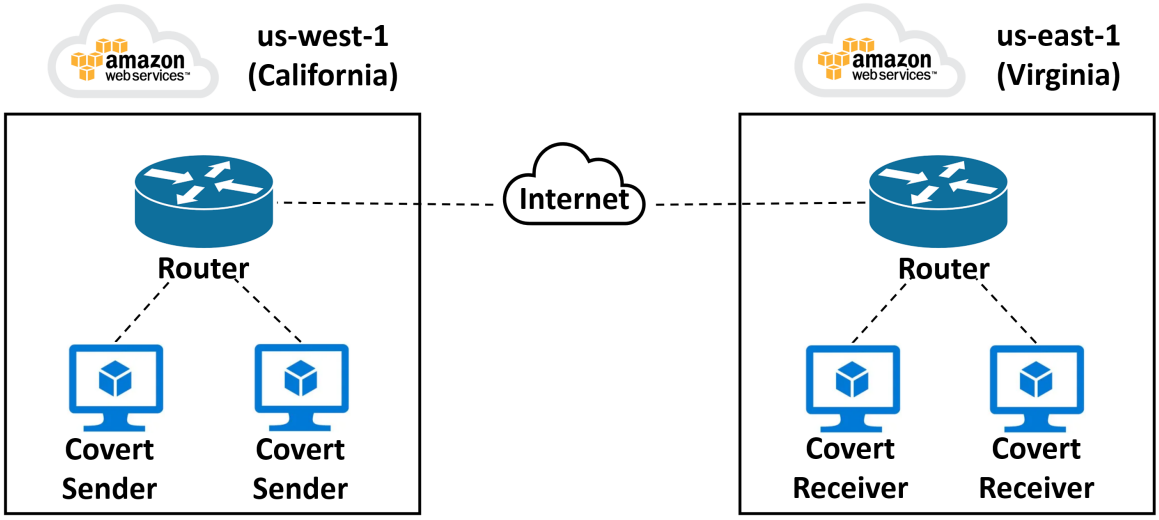

Figure 15: TCP/IP network diagram

3.6.1.1 Stealth Encoding

The first method of encoding information is within Transmission Control Protocol (TCP) source ports, shown in part 1 of Figure 16. This method involves a single covert sender, "A," and a single covert receiver, "C," transmitting information across two different TCP sockets. The receiver interprets any packet from source port "W" as an encoded zero. Similarly, it decodes any packet on source port "X" as one.

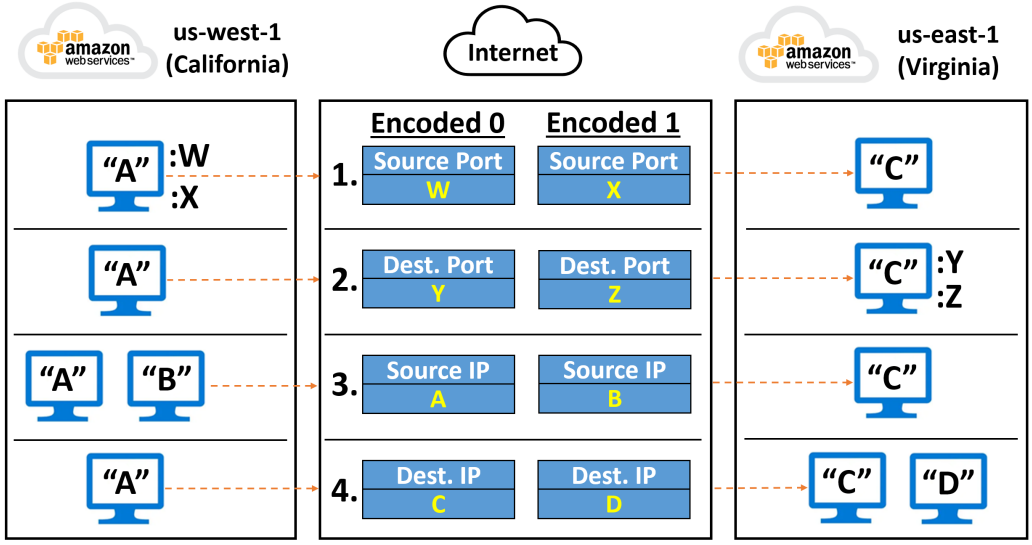

Figure 16: A packet example diagram for TCP/IP encoding methods

As shown in part 2 of Figure 16 encoding bits in TCP destination ports is also possible. This method similarly involves a single covert sender, "A," and a single covert receiver, "C," transmitting information across two different TCP sockets. The receiver interprets any packet from destination port "Y" as an encoded zero. It decodes any packet on destination port "Z" as one.

The covert channel can also encode information using source IP addresses, as shown in part 3 of Figure 16. This method involves two covert senders and a single covert receiver transmitting information across two sockets. The receiver interprets any packet from IP address "A" as an encoded zero. Similarly, it decodes any packet from IP address "B" as one.

Encoding bits within destination IP addresses is shown in part 4 of Figure 16. This method involves two covert receivers and a single covert receiver transmitting information across two sockets. Covert sender "A" encodes a zero each time it transmits a packet to covert receiver "C." Likewise, each packet sent to covert receiver "D" is encoded as one.

3.6.1.2 Bandwidth Encoding

The TCP/IP Bandwidth approach encodes data using all source and destination port permutations. This method operates similarly to the source and destination port encoding methods. However, instead of encoding a single bit between alternating ports, it encodes data across all possible source and destination port values. Since each port is 16 bits in size, it encodes 32 bits in each packet. The approach is shown in Figure 17. It results in a total of 2^{16} values for each port, ranging from 0 to 65,535.

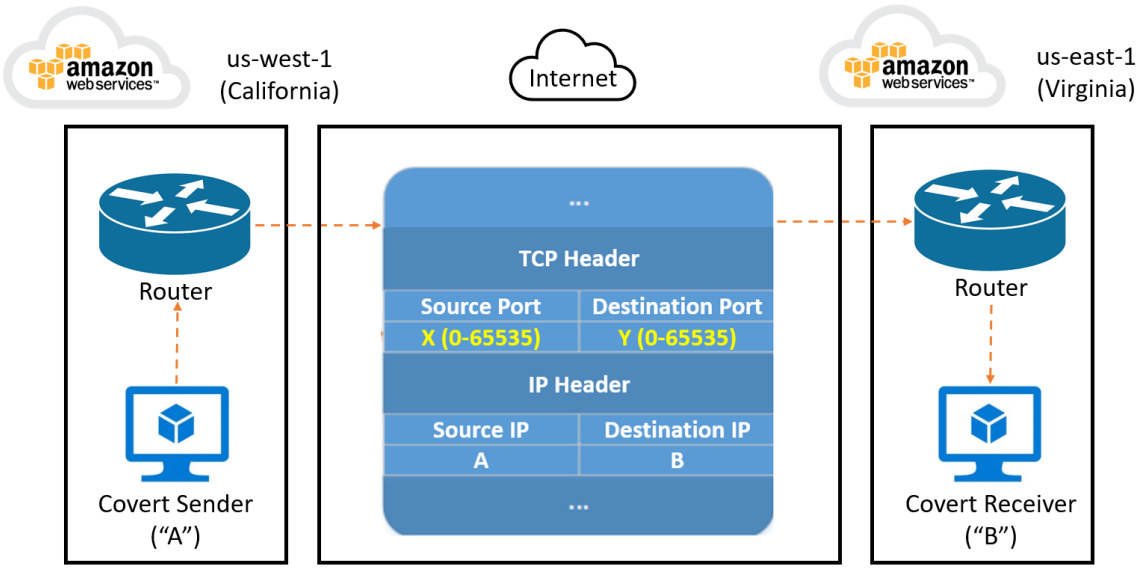

Figure 17: The network diagram for the all port permutations encoding method with example packets for each socket

3.6.2 ZigBee-based Covert Communication

The ZigBee encoding method implements both Stealth and Bandwidth modes. This methodology is shown in Figure 18. The researcher implements this experiment using two different trials, one for the Stealth mode and one for Bandwidth. Each trial is repeated three times for statistical significance, for a total of six trials.

Since the ZigBee stack does not support TCP or IP, using ports or IP addresses to encode information is not possible. However, ZigBee implements a 4-character hexadecimal address at the network level to identify each device. This address is used to encode covert bits, similar to an IP address. The majority of ZigBee traffic consists of requests from the single ZigBee Coordinator and responses from the individual ZigBee IoT devices. For creating a covert channel using network-based information in ZigBee, it is possible to use the address of the individual ZigBee device. Furthermore, to have the necessary root access to the ZigBee Coordinator, the ZigBee2MQTT library is implemented on a Raspberry Pi, enabling it to serve as the Coordinator.

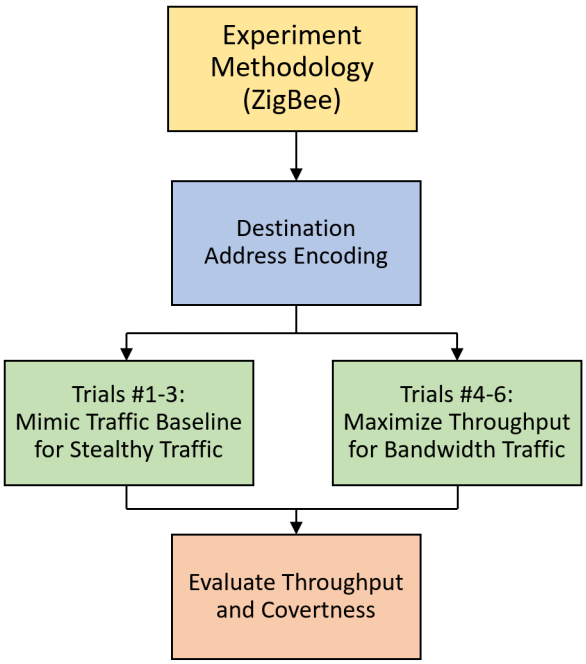

Figure 18: ZigBee based CTC experimental methodology

This experiment uses a laptop with a ZigBee adapter capable of eavesdropping on the traffic as the covert receiver. The ZigBee network diagram is shown in Figure 19. The Philips Hue encrypts the application portion of the ZigBee packets; however, it does not encrypt the network data. Therefore, any device capable of detecting ZigBee packets near the network can passively read the network addresses of the devices. This method also decreases detectability since our laptop's covert sender is out of band from the ZigBee network. This eliminates the need for the covert sender to transmit traffic outside the local network, making it less likely to be detected by an intrusion detection system. Once the covert receiver detects a packet sent from the Coordinator to either pre-agreed upon address, it will interpret it as a zero or a one.

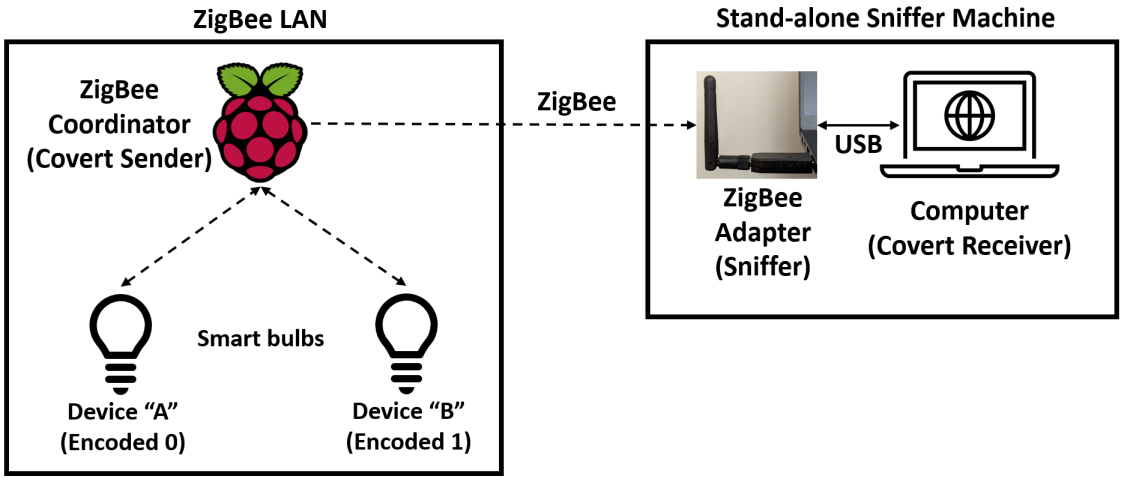

Figure 19: The network diagram for the ZigBee CTC trials

3.7 Summary

This chapter discusses the methodology behind developing and evaluating the network-based IoT CTC. It discusses the metrics, factors, and parameters used to measure and assess the system. The five different encoding methods and the three different throughput distributions used are also detailed. A total of seven trials are needed to evaluate the system's throughput, reliability, and detectability metrics. This series of trials is conducted three times, for a total of 21 trials, to create a high degree of statistical confidence in the results.

IV. Results and Analysis

4.1 Overview

This chapter presents and analyzes experimental results of the Covert Timing Channel (CTC) encoding methods presented in the methodology chapter. First, the summary of Inter-Packet Delays (IPDs) in legitimate traffic is discussed and compared with CTC traffic. Next, the throughput for each encoding method is presented and evaluated. Finally, the detectability metrics are presented.

4.2 Legitimate Traffic Analysis

This research captures traffic using an IoT device for each CTC protocol. A Ring video doorbell is used for Transmission Control Protocol/Internet Protocol (TCP/IP), and two Philips Hue smart bulbs and a Philips Hue Bridge are measured for ZigBee. The baseline traffic captures for both protocols were taken on a Local Area Network (LAN). The size of the traffic captures was determined to accommodate a 4 KB secret file transfer. When generating covert traffic, initial tests contained 1 KB, 2 KB, and 4 KB file sizes; however, the increased sample size of the 4 KB files provided a more accurate data when compared with the traffic baselines.

4.2.1 Legitimate TCP/IP Traffic

Table 4 presents the legitimate TCP/IP traffic baseline with the number of packets sent, the mean IPD, the standard deviation, and the 95% confidence interval. It measures the mean IPD and confidence interval in seconds. The data shows that 95% of the traffic has an IPD between 21 ms and 23 ms. The 2 ms variance in IPDs makes it challenging to implement typical inter-packet delay-based CTCs. With that method, the covert sender must differentiate between two distinct times to encode

data, which is difficult with network jitter. There are 321K packets in the traffic baseline because 89.1% of the traffic sent from the Ring is User Datagram Protocol (UDP) video traffic from when motion is detected. This traffic is unusable in the CTC due to its lack of sequence numbers. To maintain the detectability of the channel, the CTC also transmits this cover traffic.

Table 4: Summary of Legitimate TCP/IP Traffic

	Packets Sent	**Mean IPD**	**Standard Deviation**	**95% Confidence Interval**
Legitimate Traffic	321832	0.0220	0.5420	(0.0211, 0.0230)

4.2.2 Legitimate ZigBee Traffic

Statistical information for the legitimate ZigBee traffic baseline is presented in Table 5. It consists of repeated data requests from the ZigBee coordinator to the two smart bulbs on the network. Unlike the TCP/IP baseline, the ZigBee baseline contains no cover traffic. Each data request represents an encoded bit in the CTC, so fewer packets are needed. However, more than 32K packets are present to account for packet error or repeat packets. Additionally, the mean IPD in the baseline is 251 ms, significantly higher than the TCP/IP CTC using the Ring.

Table 5: Summary of Legitimate ZigBee Traffic

	Packets Sent	**Mean IPD**	**Standard Deviation**	**95% Confidence Interval**
Legitimate Traffic	43693	0.2514	0.3122	(0.2500, 0.2528)

4.3 Covert Traffic Analysis

Covert traffic was created by mimicking each protocol's legitimate traffic baseline. The results of transferring three different 4 KB secret files across 15 trials are shown

in the sections below.

4.3.1 Covert TCP/IP Traffic

The four Stealth encoding methods designed to evade an Intrusion Detection System (IDS) include source ports, destination ports, source Internet Protocol (IP) addresses, and destination IP addresses. As shown in Table 6, it takes 299,516 packets to transfer a 4 KB file mimicking the baseline. This is because most packets are UDP cover traffic from the doorbell's video transmission. UDP packets cannot be used in our CTC because they lack sequence numbers used for ordering. The Stealth encoding methods' number of packets sent, mean IPD, standard deviation, and 95% confidence interval are all nearly identical. The only variance between them is lower than the rounded results displayed in the table. Furthermore, the mean covert IPD of 23 ms is close to the mean legitimate IPD of 22 ms.

Table 6: Summary of Covert TCP/IP Traffic for sending 4 KB Secret File

Mode	**Packets Sent**	**Mean IPD**	**Standard Deviation**	**95% Confidence Interval**
Stealth Average	299516	0.023	0.560	(0.0210, 0.0250)
Bandwidth	1005	0.000393	0.000179	(0.000382, 0.000404)

The Bandwidth mode may not evade an IDS. However, the messages are still encoded and therefore unreadable by a third party. All possible source port and destination port combinations encode information, resulting in 32 encoded bits per packet. Additionally, the Bandwidth mode does not mimic a pre-defined traffic baseline but instead sends packets as fast as the device can. These factors result in a much smaller number of packets required to encode a 4 KB file at 1005. The Bandwidth mode also has a much faster mean send time of 393 μs than the Stealth approach of 23 ms.

4.3.2 Covert ZigBee Traffic

The Stealth and Bandwidth traffic modes for ZigBee traffic use destination addresses to encode data. The difference between them is their IPDs. The Stealth method mimics the IPDs of the legitimate ZigBee traffic. The Bandwidth mode sends traffic nearly as quickly as the device can support. When implementing this method using the Philips Hue, a 100 ms IPD was added to prevent the system from becoming overwhelmed. Several IPDs were tested, from 1 ms to 100 ms, but any IPD less than 100 ms eventually led to the system becoming backlogged and timing out before the file could be fully transmitted. These results are reflected in Table 7; the Stealth method has a mean IPD of 256.7 ms and the Bandwidth mode has an IPD of 102.5 ms.

Table 7: Summary of Covert ZigBee Traffic for sending 4KB Secret File

Encoding Method	**Packets Sent**	**Mean IPD**	**Standard Deviation**	**95% Confidence Interval**
Stealth	32159	0.2567	0.3272	(0.2531, 0.2603)
Bandwidth	31963	0.1025	0.0533	(0.1019, 0.1031)

4.4 Throughput Results

The first metric used to evaluate the CTC is the throughput of encoded bits. The throughput, measured in bits per second, is calculated by multiplying the number of error-free covert packets received by the number of encoded bits per packet. The results are discussed and evaluated for the TCP/IP and ZigBee CTCs.

4.4.1 TCP/IP Throughput Results

All the trials successfully transmitted the 4 KB secret file using TCP/IP without errors. Each of the four Stealth encoding methods had a throughput of 4.61 packets

per second, as shown in Table 8. Furthermore, each Stealth method encodes one bit per packet, either in one of the ports or IP addresses that make up the Transmission Control Protocol (TCP) socket. Therefore, the overall throughput is also 4.61 bits per second.

Table 8: Summary of throughput results for TCP/IP CTCs

Encoding Method	**Throughput**		
	Packets per second	**Bits per packet**	**Bits per second**
Stealth Average	4.61	1	**4.61**
Stealth 24-hour Average	4.51	1	**4.51**
Bandwidth	2,553.47	32	**81,710.88**

The experiment captured a traffic baseline over 48 hours to determine a more accurate average throughput. This period of traffic is averaged across 24 hours This metric is shown in Table 8 as the "Stealth 24-hour Average." It minimizes the bias present in short-term transmissions due to unpredictable real-world events occurring around the device impacting network traffic.

The experiment captured a more precise traffic baseline, shown in Table 8 as the "Stealth 24-hour Average." This metric minimizes the bias present in short-term transmissions due to unpredictable real-world events occurring around the device (e.g., more cars driving by the Ring during the day, causing it to capture more video). To compute the "Stealth 24-hour Average," the researcher averaged a 48-hour traffic capture over a 24 hours.

The Bandwidth mode transmits packets as fast as possible. This resulted in a throughput of approximately 2,553 packets per second across the three different 4 KB secret files. Since this method encodes 32 bits per packet, the overall throughput is 81.71K bits per second.

4.4.2 ZigBee Throughput Results

The ZigBee CTC is implemented via a compromised ZigBee coordinator oscillating data requests between two different devices on the network. All requests sent to one device represent an encoded zero, and any requests to the other represent an encoded one. However, the Stealth method follows the traffic baseline while the Bandwidth option sends as fast as possible.

The ZigBee trials were not error-free. Approximately 0.01% of packets contain some form of bit error. However, there is no more than one error across a single American Standard Code for Information Interchange (ASCII) character transmission because the parity bit corrected every character without further action. As displayed in Table 9, the Stealth method sends an average of 3.90 packets per second with one encoded bit per packet, also resulting in an overall throughput of 3.90 bits per second. The Bandwidth option follows the same method but has 9.76 bits per second. Due to hardware and protocol limitations, the ZigBee CTCs have noticeably less throughput than their TCP/IP counterparts.

Table 9: Summary of throughput results for ZigBee CTCs

Encoding Method	**Throughput**		
	Packets per second	**Bits per packet**	**Bits per second**
Stealth	3.90	1	**3.90**
Bandwidth	9.76	1	**9.76**

4.5 Detectability Results

The second evaluation metric measured for the CTC is detectability. Detectability estimates if the CTC is likely to be detected. This research implements four statistical tests to measure detectability: the Kolmogorov-Smirnov (KS) test, the Shape test, the Regularity test, and the Similarity test. These tests are implemented on outgoing

data collected from the covert sender since that would most accurately represent where an IDS is present. Finally, the comprehensive test results are shown in Tables 10 and 11, with each result also displayed in its respective section.

Table 10: TCP/IP Detectability Results

Encoding Method	**Secret 4KB File**	**KS Test Statistic**	**KS Test p-Value (exact)**	**Regularity Score w=100**	**Shape Test CTC Probability**
Stealth Average	File 1	0.034	1.0000	61.2451	0.9983
	File 2	0.034	1.0000	61.2473	0.9983
	File 3	0.034	1.0000	61.2457	0.9983
Bandwidth	File 1	0.672	4.99E-15	0.9549	0.9528
	File 2	0.479	4.99E-15	0.4466	0.8608
	File 3	0.558	4.99E-15	0.9165	0.2857
Legitimate Traffic	Baseline	N/A	N/A	61.2663	0.9987

Table 11: ZigBee Detectability Results

Encoding Method	**Secret 4KB File**	**KS Test Statistic**	**KS Test p-Value (exact)**	**Regularity Score w=250**	**Shape Test CTC Probability**
Stealth	File 1	0.078	1.0000	0.1791	0.9265
	File 2	0.084	1.0000	0.3343	0.9455
	File 3	0.071	1.0000	0.1761	0.9362
Bandwidth	File 1	0.394	5.27E-14	2.8172	0.0438
	File 2	0.414	6.93E-14	46.0641	0.4224
	File 3	0.435	6.74E-14	39.2816	0.4419
Legitimate Traffic	Baseline	N/A	N/A	0.1862	0.9776

4.5.1 Kolmogorov-Smirnov Test

The TCP/IP CTC results are shown in Table 12. Beginning with the KS-test, the null hypothesis cannot be rejected for the Stealth traffic since each trial's test statistic is less than its respective p-value. Therefore, the source port, destination port, source IP address, and destination IP address encoding methods have a distri-

bution that successfully mimics the baseline. The results for the Bandwidth option are shown for completeness. Since this method does not mimic the baseline, its test statistic is greater than the p-value, confirming that the null hypothesis can be rejected. Additionally, the legitimate and Stealth empirical Cumulative Distribution Functions (eCDFs) are shown in Figure 20. By visual inspection, the covert traffic is likely to match the legitimate traffic distribution.

Table 12: KS-test results for TCP/IP and ZigBee

Encoding Method	**Secret 4KB File**	**KS Test Statistic**	**KS-Test p-Value (exact)**
Stealth (TCP/IP)	File 1	0.034	1.0000
	File 2	0.034	1.0000
	File 3	0.034	1.0000
Bandwidth (TCP/IP)	File 1	0.672	4.99E-15
	File 2	0.479	4.99E-15
	File 3	0.558	4.99E-15
Stealth (ZigBee)	File 1	0.078	1.0000
	File 2	0.084	1.0000
	File 3	0.071	1.0000
Bandwidth (ZigBee)	File 1	0.394	5.27E-14
	File 2	0.414	6.93E-14
	File 3	0.435	6.74E-14

As shown in Table 12, the Stealth method, using ZigBee network addresses, passes the KS test. Similar to the TCP/IP Bandwidth method, the ZigBee Bandwidth option fails and can be detected using the KS test. Figure 21 shows the eCDF of the Stealth ZigBee traffic. While the difference between the distributions is at most 7.76%, the Stealth covert traffic and the legitimate baseline are still visually correlated.

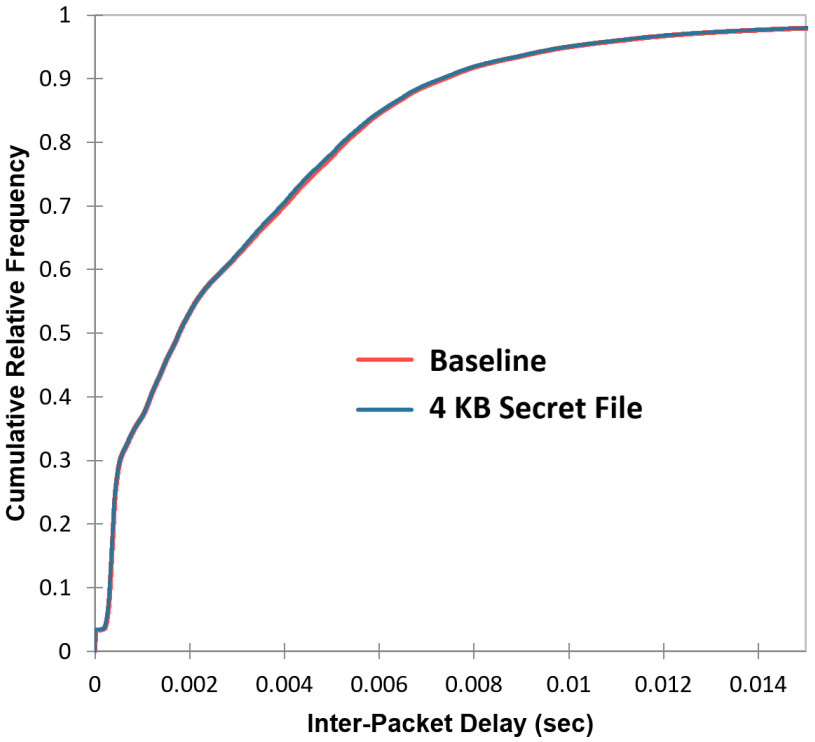

Figure 20: The Stealth TCP/IP CTC eCDF compared to the legitimate baseline

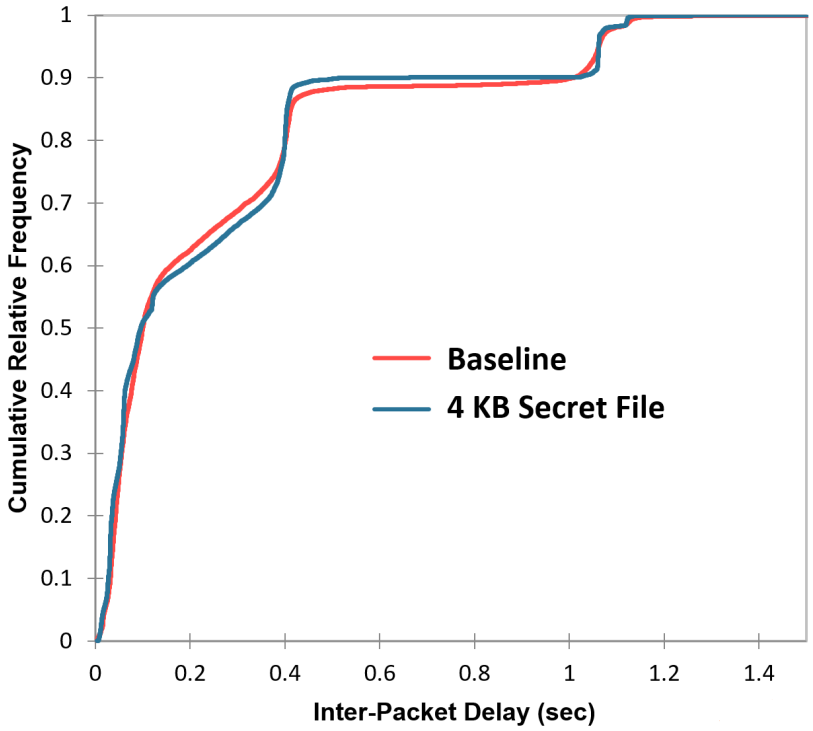

Figure 21: The Stealth ZigBee CTC eCDF compared to the legitimate baseline

4.5.2 Shape Test

The Shape test can detect CTCs that oscillate encoded bits between two distinct values. As shown in Figure 22, these two packet count peaks result in a low volume at the mean between them, causing the probability of a CTC to approach one. For the Shape test, the width of each bin is 1 ms. Higher precision is unneeded and would only dilute the results. Additionally, a lower precision factor would result in too few bins to accurately represent the results.

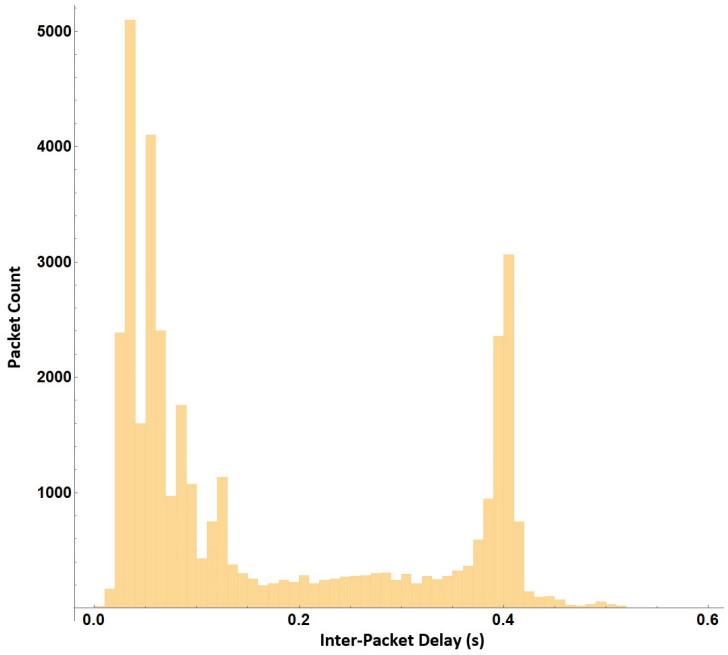

Figure 22: The IPD distribution for the ZigBee baseline

All Stealth traffic captures have above a 90% CTC probability and thus fail the Shape test. However, the legitimate traffic baseline also failed because there are too many data streams within the TCP/IP traffic for the Shape test to differentiate traffic accurately.

The ZigBee traffic naturally oscillates between two distinct values. Specifically, the Philips Hue Bridge sends three different data requests, approximately 50 ms apart, to each smart bulb and then waits, as shown by the IPD distribution in Figure

22. Since the Shape test detects data peaks and compares them to the mean, when the peak is at the mean, it resembles non-CTC traffic. This limitation makes the Shape test inadequate for measuring the detectability of the ZigBee Stealth data. Additionally, the Shape test cannot evaluate the ZigBee Bandwidth method due to the lack of variance caused by the 100 ms artificial delay between packets. This research conducts all the tests for completeness and reports their results in Table 13.

Table 13: Shape test results for TCP/IP and ZigBee

Encoding Method	**Secret 4KB File**	**CTC Probability (TCP/IP)**	**CTC Probability (ZigBee)**
Stealth Average	File 1	0.9983	0.9265
	File 2	0.9983	0.9455
	File 3	0.9983	0.9362
Bandwidth	File 1	0.9528	0.0438
	File 2	0.8608	0.4224
	File 3	0.2857	0.4419
Legitimate Traffic	Baseline	0.9987	0.9776

4.5.3 Regularity Test

The results for the TCP/IP and ZigBee regularity scores are shown in Table 14. All the Stealth TCP/IP methods pass the regularity test with a difference in regularity scores of less than 1% compared to the legitimate traffic baseline. The opposite is true for the Bandwidth methods; the regularity scores differ up to 99%, failing the test. Two of the three Stealth encoding methods for ZigBee passed, and the other failed due to a higher IPD variance with that particular trial. This caused it to slightly, but noticeably, differ from the baseline. As expected, there was a significant variance in the Bandwidth method's regularity scores, resulting in all trials being detectable.

Table 14: Regularity test results for TCP/IP and ZigBee

Encoding Method	Secret 4KB File	TCP/IP Regularity Score w=100	ZigBee Regularity Score w=250
Stealth Average	File 1	61.2451	0.1791
	File 2	61.2473	0.3343
	File 3	61.2457	0.1761
Bandwidth	File 1	0.9549	2.8172
	File 2	0.4466	46.0641
	File 3	0.9165	39.2816
Legitimate Traffic	Baseline	61.2663	0.1862

4.5.4 Similarity Test

The Similarity test compares the difference between IPDs by putting them in ascending order and comparing each consecutive IPD. It then calculates the relative difference between each successive packet and then groups them based on the size of the difference [48]. As depicted in Table 15, the ϵ-*Similarity Scores* for each group represent the proportion of IPDs that have less than that percentage difference. For example, a score of 98.75 in the 0.0005 group means that 98.75% of traffic has a 0.05% or less difference from the next consecutive packet. Finally, the Similarity test compares the covert traffic scores to those taken from the legitimate traffic baseline. The same 10% difference threshold for the Regularity test applies to the Similarity test. If the traffic fails most of the comparisons, three of the five, a CTC may be present.

The researcher observes that 96% of the TCP/IP traffic has a relative difference of 0% due to a large amount of cover traffic. As a result, most traffic skews towards a minor percentage difference. In Table 15, the Stealth ϵ-*Similarity Scores* in all five groups are within the 10% difference threshold and thus pass the test. The only score that comes close, at a 9.84%, is the >0.001 similarity score. Due to the small amount of traffic in that group, a slight increase in the number of IPDs present can cause a

meaningful increase. When evaluating the Bandwidth method, the traffic noticeably exceeds the 10% threshold in all five groups, failing the test.

Table 15: TCP/IP Similarity Results

Encoding Method	**Secret 4KB File**	***ϵ-Similarity Score***				
		0.00025	**0.0005**	**0.00075**	**0.001**	**> 0.001**
Stealth Average	File 1	97.73	98.77	99.16	99.37	0.63
	File 2	97.72	98.77	99.16	99.36	0.64
	File 3	97.71	98.76	99.16	99.37	0.63
Bandwidth	File 1	77.39	77.39	77.39	77.39	22.61
	File 2	74.30	74.30	74.30	74.30	25.70
	File 3	83.05	83.05	83.05	83.05	16.95
Legitimate Traffic	Baseline	97.86	98.85	99.22	99.42	0.58

The ZigBee Similarity test results are displayed in Table 16. For all three Stealth trials, the first four groups fall within the acceptable margin, but the >0.001 group exceeds it, resulting in four out of the five groups passing. Since a majority of the groups pass, the Stealth ZigBee method is not detected using the Similarity test. Unexpectedly, the Bandwidth option passes as well, fitting the similarity scores of the baseline even better than the Stealth traffic. However, upon further analysis, the 100 ms IPD implemented in the ZigBee Bandwidth method appears to have dwarfed the natural variance in the protocol. The Similarity test only measures and compares the IPD differences between each packet, so even though the Bandwidth option passes this test, its traffic distribution is very different from the legitimate baseline, which is why it fails the Regularity and KS tests.

Table 16: ZigBee Similarity Results

Encoding Method	Secret 4KB File	ϵ-*Similarity Score*				
		0.00025	**0.0005**	**0.00075**	**0.001**	**> 0.001**
Stealth	File 1	83.77	93.08	96.00	97.19	2.81
	File 2	83.96	93.07	95.88	97.01	2.99
	File 3	84.99	94.10	96.28	97.43	2.57
Bandwidth	File 1	89.59	94.27	96.31	97.32	2.68
	File 2	91.50	94.94	96.51	97.34	2.66
	File 3	92.70	95.41	96.71	97.35	2.67
Legitimate Traffic	Baseline	89.09	96.08	98.02	98.76	1.24

4.6 Comparison Assessment

This research creates a CTC capable of optionally having better detectability or throughput than other common CTCs. In Table 17, the four different CTCs from this research are compared with a common On/Off Timing Channel (OTC), the IP-Timing CTC, which encodes data in network information, and an Internet of Things (IoT) CTC. The OTC serves as a baseline CTC comparison, the IP-Timing CTC is most similar to the Bandwidth method, and the IoT CTC is comparable to the Stealth methods. To the researcher's knowledge, no comparable CTCs exist within ZigBee. Therefore, a relatable CTC within another IoT protocol, Message Queuing Telemetry Transport (MQTT), is used instead.

Table 17: Comparison of CTC evaluation metrics (* annotes presented CTCs)

CTC	**Throughput (bps)**	**Detection Resistant**	**IoT**
ZigBee (Stealth)*	3.90	Yes	Yes
MQTT 4-topic OTC [3]	4.00	Yes	Yes
TCP/IP (Stealth)*	4.61	Yes	Yes
ZigBee (Bandwidth)*	9.76	No	Yes
OTC [38]	180.49	No	No
IP-Timing [38]	2,790	No	No
TCP/IP (Bandwidth)*	81,711	No	Yes

The throughput and detection resistance of the seven CTCs in Table 17 are ordered

by bandwidth. The three detection-resistant CTCs have the low est throughput. The MQTT CTC's throughput (4.00 bps) is slightly higher than the ZigBee Stealth (3.90 bps) and lower than the TCP/IP Stealth (4.61 bps). However, the authors of the MQTT CTC only evaluated detection with a single test. Therefore, it may have comparable throughput, but it cannot be determined to have equal detectability.

The four non-detection-resistant CTCs have significantly higher throughput. The ZigBee Bandwidth CTC had the smallest throughput of 9.76 bps due to the 100 ms IPD present. The OTC has a throughput of 180.49 bps and is a baseline for the other CTCs. The IP-Timing CTC is most like those in this research and has a throughput of 2790 bps. The TCP/IP Bandwidth option has significantly higher throughput (81,711 bps) because it uses all possible port combinations, encoding 32 bits per packet instead of one.

4.7 Summary

This chapter presents and analyzes the data collected from five Stealth and two Bandwidth encoding methods across two different network protocols. Additionally, a statistical analysis of performance metrics in data throughput and detectability is performed for each CTC. The detectability analysis employs four different tests: the KS test, the Shape test, the Regularity test, and the Similarity test. The throughput for the Bandwidth TCP/IP method was 17,725 times greater than the Stealth encoding methods, but it also failed every detectability test, where the Stealth methods all passed. The only exception was the Shape test, which is inadequate for measuring noisy traffic. For ZigBee, the Stealth encoding method passed a majority of the tests, with one trial failing the Regularity test. The Bandwidth option failed all detection tests except the Similarity test. The results show that the Stealth encoding methods are best employed when detection methods are present and the Bandwidth options

are superior when one is not.

V. Conclusions

5.1 Research Conclusions

Internet of Things (IoT) devices sense and collect information based on real-world events around them, making the data they collect particularly sensitive. These devices often have limited resources, forcing them to use lightweight protocols and security solutions. Understanding the vulnerabilities of IoT protocols and devices, how to exploit them, and how to secure them is a high priority. One practical method of exfiltrating data from these devices is via covert channels.

This research develops a Covert Timing Channel (CTC) to exfiltrate information using IoT devices. It encodes data using preexisting network information in two different IoT protocols, Transmission Control Protocol/Internet Protocol (TCP/IP) and ZigBee. TCP/IP approach encodes information within source ports, destination ports, source IP addresses, and destination IP addresses. The ZigBee approach encodes data within network destination addresses. Furthermore, each protocol can send traffic in two different modes, one focused on Stealth and the other on Bandwidth. The Stealth mode mimics legitimate traffic patterns, making it difficult to detect, and the Bandwidth mode sends traffic as fast as possible.

The throughput and detectability of each channel are evaluated by sending 4 KB secret files. The number of encoded bits transmitted per second determines channel throughput. The Stealth distribution has a throughput of 4.61 bps for TCP/IP and 3.90 bps for ZigBee. The TCP/IP Bandwidth throughput is 81.71 Kbps, and ZigBee is 9.76 bps. The ZigBee Bandwidth method also required a 10 ms Inter-Packet Delay (IPD) to process traffic continually. The research includes this limitation because the ZigBee devices cannot handle traffic at a higher rate without eventually crashing, making them not ideal for a high-throughput CTC.

Detectability is calculated by applying statistical detection tests. This research employs four different tests to measure detectability: the Kolmogorov-Smirnov (KS) test, the Shape test, the Regularity test, and the Similarity test. Except for the Shape test, the Stealth traffic distributions successfully evade detection. However, the legitimate traffic baselines also fail the Shape test. They fail because the test is not adept at differentiating noisy traffic, making it insufficient for this research. Since the Bandwidth distributions focus on throughput over detectability, they are detectable by statistical analysis.

Finally, the researcher compared evaluation metrics from this research with other comparable CTCs. The detection resistant CTCs had similar throughput, ranging from 3.90 to 4.61 bps. When rating detectability, the Stealth CTCs evaded all four detection tests compared to the Message Queuing Telemetry Transport (MQTT) CTC's single test. Therefore, the MQTT CTC may be more detectable than this research's Stealth CTCs. The throughput-focused CTCs ranged from 9.76 to 81,711 bps, with the TCP/IP Bandwidth CTC having the most significant throughput.

5.2 Countermeasures and Limitations

Despite the Stealth CTCs being resistant to statistical-based detection, there are still various ways they could be detected. This is due to the differing nature of a covert traffic stream from a legitimate one. The subsequent methods may detect the CTC:

- **Behavior-based Intrusion Detection System (IDS):** An IDS with precise traffic normalization could detect the CTC. Specifically, examining socket data and the number of them open between devices could detect the channel. However, these are not commonly found on IoT networks.
- **Existence of the Traffic Stream:** Although the covert traffic may mimic

legitimate traffic, the presence of the channel may be irregular. For example, if a device does not normally transmit traffic outside the network and then does so to exfiltrate data.

- **Device Behavior:** The host application may not function properly if the CTC replaces the standard traffic. If the channel creates an additional traffic stream, the additional overhead may negatively impact the device and not go unnoticed.

5.3 Future Work

There are several avenues to improve and expand upon this research. The following have been identified:

- **Additional test devices:** Only a single type of device was used as a traffic baseline for each protocol. Although these devices are pervasive, creating additional traffic baselines with other devices would increase the CTC's effectiveness.

- **Different protocols:** Even though this research supports two of the most popular IoT protocols, TCP/IP and ZigBee, there are many more. Increasing the number of protocols the CTC supports would also increase the likelihood that the CTC can be successfully implemented on a network.

- **Distributed CTC:** The goal of this research was a proof of concept for network-based CTCs across multiple encoding methods. These individual encoding methods could be combined into a single, distributed CTC to increase throughput further. All the encoding methods combined will theoretically increase throughput several times over.

- **Data compression:** Compressing encoded messages prior to transmission and decompressing them on arrival would result in a more efficient throughput. For

this research's 4 KB test files, the compressed file was 11.5% smaller in 7z format and 12% smaller in zip format.

- **Custom encoding scheme:** This research uses American Standard Code for Information Interchange (ASCII) to encode characters; however, many characters in the ASCII table are rarely used. Implementing a custom encoding scheme to reduce the character set to only commonly used characters could reduce the overhead by one or two bits per character.

5.4 Research Significance

Since the beginning of time, wars have been decided by superior knowledge and technology. The IoT encompasses both of these advantages. The motivation for this research is to improve knowledge and control over the battlefield through the IoT. As the IoT grows, the more centralized knowledge and control become. The value of the data these devices constantly monitoring the world around them provide is immeasurable. The IoT's impact does not stop on the battlefield; it extends into the household. Compromising these devices is much easier on a home network, which lacks the strict security measures present on the Department of Defense (DoD) networks. Additionally, it could lead to the exfiltration of sensitive information such as audio, video, or location data. Creating a covert channel enables the exfiltration of this data without being caught.

These covert channels can also better protect DoD IoT assets and information. In an insecure environment, this CTC could be used to transfer information securely, without the need for encryption. Since existing network traffic encodes the secret transmission, an adversary may not know it occurred. Additionally, the knowledge of these exploits can help better detect similar exploits on DoD systems and prevent them from occurring.

Bibliography

1. Parvaneh Asghari, Amir Masoud Rahmani, and Hamid Haj Seyyed Javadi. Internet of Things applications: A systematic review. *Computer Networks*, 148:241–261, 2019.

2. Digi International Inc. *XBee/XBee-PRO S2C Zigbee RF Module*, Sep 2017.

3. Aleksandar Velinov, Aleksandra Mileva, Steffen Wendzel, and Wojciech Mazurczyk. Covert channels in the mqtt-based internet of things. *IEEE Access*, 7:161899–161915, 2019.

4. Yuan Tan, Xiaosong Zhang, Kashif Sharif, Chen Lian, Quanxin Zhang, and Yuanzhang Li. Covert Timing Channels for IoT over Mobile Networks. *IEEE Wireless Communications*, 25(December):12–18, 2018.

5. Fred Ramsey and Daniel Schafer. *The statistical sleuth: a course in methods of data analysis*. Cengage Learning, 2012.

6. Victor Baños-Gonzalez, M. Shahwaiz Afaqui, Elena Lopez-Aguilera, and Eduard Garcia-Villegas. IEEE 802.11ah: A technology to face the IoT challenge. *Sensors (Switzerland)*, 16(11), 2016.

7. Serdar Cabuk, Carla E. Brodley, and Clay Shields. IP covert timing channels: Design and detection. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 178–187, 2004.

8. The National Intelligence Council and SRI Consulting Business Intelligence. Disruptive Civil Technologies Six Technologies With Potential Impacts on US Interests Out to 2025. *National Intelligence Council*, (April):1–48, 2008.

9. Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.

10. Tanweer Alam. A Reliable Communication Framework and Its Use in Internet of Things (IoT). *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 3(5):450–456, 2018.

11. Michael Chiaramonte, Jennifer Veigel, and Jeffrey COllins. AIR FORCE CYBERWORX REPORT 17-002 Air Force Smart Bases. (002):1–16, 2017.

12. Bhabendu Kumar Mohanta, Debasish Jena, Utkalika Satapathy, and Srikanta Patnaik. Survey on IoT security: Challenges and solution using machine learning, artificial intelligence and blockchain technology. *Internet of Things*, 11:100227, 2020.

13. Fadele Ayotunde Alaba, Mazliza Othman, Ibrahim Abaker Targio Hashem, and Faiz Alotaibi. Internet of Things security: A survey. *Journal of Network and Computer Applications*, 88:10–28, 2017.

14. Alem Čolaković and Mesud Hadžialić. Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. *Computer Networks*, 144:17–39, 2018.

15. Rwan Mahmoud, Tasneem Yousuf, Fadi Aloul, and Imran Zualkernan. Internet of things (IoT) security: Current status, challenges and prospective measures. *2015 10th International Conference for Internet Technology and Secured Transactions, ICITST 2015*, pages 336–341, 2016.

16. Shancang Li, Li Da Xu, and Shanshan Zhao. The internet of things: a survey. *Information Systems Frontiers*, 17(2):243–259, 2015.

17. Lan Li. Security Architecture in the Internet of Things. *Securing the Internet of Things*, 1:374–377, 2012.

18. Debasis Bandyopadhyay and Jaydip Sen. Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58(1):49–69, 2011.

19. Alexander Kott, Ananthram Swami, and Bruce J. West. The Internet of Battle Things. *Computer*, 49(12):70–75, 2016.

20. Ms Dharmistha D Vishwakarma. IEEE 802.15.4 and ZigBee: A Conceptual Study. *International Journal of Advanced Research in Computer and Communication Engineering*, 1(7):477–480, 2012.

21. Abhishek Kumar and Sandeep Gupta. Study on zigbee technology. *International Journal of Engineering Sciences Research Technology*, 2:2733–2738, 10 2013.

22. Mahmoud Elkhodr, Seyed Shahrestani, and Hon Cheung. Emerging Wireless Technologies in the Internet of Things : A Comparative Study. *International Journal of Wireless Mobile Networks*, 8(5):67–82, 2016.

23. Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials*, 17(4):2347–2376, 2015.

24. Tara Salman and Raj Jain. A Survey of Protocols and Standards for Internet of Things. *Advanced Computing and Communications*, 1(1), 2017.

25. N. Kushalnagar, G. Montenegro, and C. Schumacher. 6LoWPANs: Overview, Assumptions, Problem Statement, and Goals. *Rfc 4919, Ietf*, 12(235):1–12, 2007.

26. Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

27. Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad Reza Sadeghi, and Selcuk Uluagac. Peek-a-boo: I see your smart home activities, even encrypted! *WiSec 2020 - Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 207–218, 2020.

28. Omer Shwartz, Yael Mathov, Michael Bohadana, Yuval Elovici, and Yossi Oren. Reverse Engineering IoT Devices: Effective Techniques and Methods. *IEEE Internet of Things Journal*, 5(6):4965–4976, 2018.

29. Annarita Giani, Vincent H. Berk, and George V. Cybenko. Data exfiltration and covert channels. *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V*, 6201:620103, 2006.

30. Jason Jaskolka, Ridha Khedri, and Qinglei Zhang. On the necessary conditions for covert channel existence: A state-of-the-art survey. In *Procedia Computer Science*, volume 10, pages 458–465. Elsevier B.V., 2012.

31. Butler W Lampson. Note on the Confinement Problem. *Commun. ACM*, 10:613–615, 1973.

32. Yogesh Heda and Rinku Shah. Covert channel design and detection techniques: a survey. *2015 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pages 1–6, 2016.

33. David Llamas, C Allison, and A Miller. Covert channels in internet protocols: A survey. In *Proceedings of the 6th Annual Postgraduate Symposium about the Con-*

vergence of Telecommunications, Networking and Broadcasting, PGNET, volume 2005.

34. Lihong Yao, Xiaochao Zi, Li Pan, and Jianhua Li. A study of on/off timing channel based on packet delay distribution. *Computers and Security*, 28(8):785–794, 2009.

35. Wayne C Henry and Barry E Mullins. VANISH: a variable advanced network IRC stealth handling system. *Int. J. Security and Networks*, 9(2):114–123, 2014.

36. Sebastian Zander, Grenville Armitage, and Philip Branch. A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys Tutorials*, 9:44–57, 2007.

37. Deepa Kundur and Kamran Ahsan. Practical internet steganography: Data hiding in ip, 2003.

38. Hermine Hovhannisyan, Kejie Lu, and Jianping Wang. A novel high-speed IP-timing covert channel: Design and evaluation. *IEEE International Conference on Communications*, 2015-Septe:7198–7203, 2015.

39. James Gimbi, Daryl Johnson, Peter Lutz, and Bo Yuan. A Covert Channel Over Transport Layer Source Ports. *The 2012 International Conference on Security and Management*, (July):2–6, 2012.

40. Krzysztof Cabaj, Piotr Z. Órawski, Piotr Nowakowski, Maciej Purski, and Wojciech Mazurczyk. Efficient distributed network covert channels for internet of things environments. *Journal of Cybersecurity*, 6(1):1–18, 2020.

41. Aleksandra Mileva, Aleksandar Velinov, and Done Stojanov. New Covert Channels in Internet of Things. *SECURWARE 2018 : The Twelfth International Con-*

ference on Emerging Security Information, Systems and Technologies, (c):30–36, 2018.

42. Cristina Alcaraz, Giuseppe Bernieri, Federica Pascucci, Javier Lopez, and Roberto Setola. Covert channels-based stealth attacks in industry 4.0. *IEEE Systems Journal*, 13:3980–3988, 2019.

43. Aleksandra Mileva, Aleksandar Velinov, Laura Hartmann, Steffen Wendzel, and Wojciech Mazurczyk. Comprehensive analysis of MQTT 5.0 susceptibility to network covert channels. *Computers Security*, 104:102207, 2021.

44. Pai Peng, Peng Ning, and Douglas S Reeves. On the secrecy of timing-based active watermarking trace-back techniques *. pages 1–15, 2006.

45. Wayne C Henry. *Covert Channels within IRC*. PhD thesis, Air Force Institute of Technology, 2011.

46. Steven Gianvecchio and Haining Wang. An entropy-based approach to detecting covert timing channels. *IEEE Transactions on Dependable and Secure Computing*, 8(6):785–797, 2007.

47. Vincent Berk, Annarita Giani, George Cybenko, and N Hanover. Detection of covert channel encoding in network packet delays. *Rapport technique TR536, de lUniversité de Dartmouth*, (August):19, 2005.

48. Serdar Cabuk, Carla E. Brodley, and Clay Shields. IP covert channel detection. *ACM Transactions on Information and System Security*, 12(4), 2009.

Acronyms

6LoWPAN IPv6 over Low power Wireless Personal Area Network. 9

ASCII American Standard Code for Information Interchange. 21, 22, 27, 39, 53

AWS Amazon Web Services. iv, 26, 27, 28

BLE Bluetooth Low Energy. 8

CoAP Constrained Application Protocol. 15, 16

CSA Connectivity Standards Alliance. 6

CSC Covert Storage Channel. 10, 11, 12, 16

CTC Covert Timing Channel. iv, viii, ix, x, 2, 3, 10, 11, 12, 14, 15, 16, 18, 19, 20, 24, 25, 27, 28, 33, 34, 35, 36, 37, 38, 39, 40, 42, 43, 44, 45, 47, 48, 50, 51, 52, 53

CUT Component Under Test. 23

DCC Direct Covert Channel. viii, 12, 13

DNCC Distributed Network Covert Channel. 15, 16

DoD Department of Defense. 1, 53

eCDF empirical Cumulative Distribution Function. ix, 17, 41, 42

ICC Indirect Covert Channel. viii, 12, 13

IDS Intrusion Detection System. 36, 51

IEEE Institute of Electrical and Electronics Engineers. 6, 8, 9

IoT Internet of Things. iv, viii, x, 1, 2, 3, 4, 5, 6, 8, 9, 14, 15, 16, 19, 20, 24, 27, 28, 31, 33, 47, 50, 51, 52, 53

IP Internet Protocol. vi, 2, 9, 12, 24, 25, 30, 31, 36, 38, 40, 48

IPD Inter-Packet Delay. ix, 11, 16, 18, 19, 25, 34, 35, 36, 37, 43, 44, 45, 46, 48, 50

KS Kolmogorov-Smirnov. iv, x, 17, 39, 40, 41, 46, 48, 51

LAN Local Area Network. iv, 26, 34

LTE-A Long-Term Evolution Advanced. 8

LTS Long-Term Service. 26, 27

MAC Media Access Control. 6

MQTT Message Queuing Telemetry Transport. 12, 15, 16, 47, 48, 51

OS Operating System. 26, 27

OTC On/Off Timing Channel. 47, 48

RAM Random Access Memory. 26, 27

SUT System Under Test. 20, 23

TCP Transmission Control Protocol. 29, 30, 31, 38

TCP/IP Transmission Control Protocol/Internet Protocol. iv, viii, ix, x, 2, 5, 20, 24, 25, 26, 27, 28, 29, 30, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 48, 50, 51, 52

UDP User Datagram Protocol. 35, 36

vCPU Virtual Central Processing Unit. 26

VoIP Voice over Internet Protocol. viii, 14

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
24–03–2022	Master's Thesis	Sept 2021 — Mar 2022

4. TITLE AND SUBTITLE

Exploiting the IoT Through
Network-based Covert Channels

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Harris, Kyle, Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-22-031

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RYAA
WPAFB OH 45433-7765
DSN 785-3636, COMM 937-255-3636

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Information leaks are a top concern to industry and government leaders. The IoT is a technology capable of sensing real-world events. A method for exfiltrating data from these devices is by covert channel.
This research designs a novel IoT CTC without the need for inter-packet delays to encode data. Instead, it encodes data within preexisting network information, namely ports or addresses. Additionally, the CTC can be implemented in two different modes: Stealth and Bandwidth. Performance is measured using throughput and detectability. The Stealth methods mimic legitimate traffic captures while the Bandwidth methods forgo this approach for maximum throughput. Detection results are presented using shape and regularity-based detection tests.
The Stealth results have a throughput of 4.61 bits per second (bps) for TCP/IP and 3.90 bps for ZigBee. They also evade shape and regularity-based detection tests. The Bandwidth methods average 81.7 Kbps for TCP/IP and 9.76 bps for ZigBee, but are evident in detection tests.

15. SUBJECT TERMS

covert channel, covert timing channel (CTC), Internet of Things (IoT), inter-packet delay (IPD), Transmission Control Protocol/Internet Protocol (TCP/IP), ZigBee

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Lt Col Wayne Henry, AFIT/ENG
U	U	U	UU	74	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x7243; wayne.henry@afit.edu