Detecting Streaming Wireless Cameras with Timing Analysis

Kevin Wu

A Master's Thesis
submitted in partial fulfillment of the
requirements of the degree of

Master of Science in Cyber Security Engineering

University of Washington

2018

Reading Committee:

Brent Lagesse, Chair

Marc Dupuis

Yang Peng

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Detecting Streaming Wireless Cameras with Timing Analysis

Kevin Wu

Chair of the Supervisory Committee:
Dr. Brent Lagesse
Computing & Software Systems

The Internet of Things (IoT) is growing rapidly thanks to the convenience it provides to users, as sensors collect, communicate, and collaborate with each other to provide better services. Wi-Fi cameras from a variety of manufacturers have been widely adopted to provide inexpensive monitoring services to general consumers. Although Wi-Fi cameras provide real-time monitoring, these devices often come with weak security mechanisms. This allow adversaries to exploit those IoT devices and have total control over with admin privileges. Moreover, those Wi-Fi cameras can be installed with bad intentions. Several incidents have been reported, where hidden Wi-Fi cameras are found in rental services such as Airbnb. To counter Wi-Fi camera spying and monitoring, we proposed a novel method to detect hidden Wi-Fi cameras, using timing analysis and a mobile phone as a detector. In order to provide constant and faster communication, IoT devices often required low-latency networks. Accordingly, the proposed methodology performs statistical analysis (Correlation Coefficient, Dynamic Time Warping, Kullback-Leibler divergence, and Jensen-Shannon divergence) to measure the similarity scores of network traffic streams and the recorded video from the mobile phone. Further, the similarity score is then used to identify hidden Wi-Fi cameras in the environment. The results of our experiments show that the proposed detection methodology can successfully discover hidden Wi-Fi cameras with an accuracy rate of 97.436%.

TABLE OF CONTENTS

Page

List of Figures iii

Chapter 1: Introduction 1
1.1 Recent attack targeting Wi-Fi camera 2
1.2 Attacker models 3
1.3 Current state-of-the-art research regarding IoT devices 4
1.4 Proposed methodology 5
1.5 Terminology 6

Chapter 2: Related Works 8
2.1 Network traffic analysis 9
2.2 Machine learning 10
2.3 Timing analysis 13

Chapter 3: Methodology 17
3.1 Proposed detecting methodology 17
3.2 Proposed detection models 31
3.3 Scenario of the experiments 33
3.4 Goal and metrics 34
3.5 Usage scenario 34

Chapter 4: Results 36
4.1 Data collections 37
4.2 Results 40
4.3 Evaluation metrics of the detection model 45
4.4 Results in evaluation metrics and the selected threshold-based classifiers . . . 47
4.5 Artificial neural network (ANN) model selections 50

4.6 The selected best classifiers 51

Chapter 5: Conclusion and Future Work 53

Bibliography 55

LIST OF FIGURES

Figure Number		Page
2.1	Illustration of different detecting mechanisms researchers have proposed.	8
3.1	The proposed detection framework.	20
3.2	Illustration of CC on two data streams.	25
3.3	Warping matrix of Sequence A and Sequence B.	26
3.4	Illustration of DTW on two temporal sequences.	27
3.5	Illustration of KLD differences on two probability distributions.	28
3.6	Illustration of JSD calculation.	29
3.7	Structure of a ANN.	30
3.8	Illustration of a artificial neuron.	31
3.9	Flowchart of the two detection models.	32
4.1	Data sample of the Wi-Fi-camera-based detection model.	39
4.2	Data sample of the mobile-phone-based detection model.	40
4.3	Plotted CC, KLD, and JSD for Wi-Fi-camera-based detection model.	48
4.4	Plotted DTW, KLD, and JSD for mobile-phone-based detection model.	48

Chapter 1

INTRODUCTION

The Internet of Things (IoT) refers to a nested network of sensors that connect and exchange data with each other. These sensors collect and share data between devices to provide convenient, almost instant services. According to Statista [36], installed IoT devices reached 20 million in 2017 and are estimated to reach almost 31 billion worldwide by 2020. IoT devices have been adopted not only in industrial environments but also home spaces. Examples of IoT devices in home spaces included smart locks, smart kitchen aids, smart TVs, smart light-bulbs, and Wi-Fi cameras. The ability to control these devices with voice-controlled home assistant devices, such as Amazon Echo and Google Home, contributes to their popularity. With their widespread use, however, managing these devices becomes an issue. Even worse, currently there are fewer no government regulations or industry standards regarding IoT devices. IoT devices are typically connected through Wi-Fi and are constantly transferring collected information. Most IoT devices operate within a Local Area Network (LAN), a group of computers and devices that share a common wired or wireless network communication. Small-office network is an example of LAN, where multiple computers and printers are connected to a single shared network. Another example of LAN is the home network, where mobile phones and computers share and are connected to a common network. When U.S. Department of Defense presented the first version of the Internet in the 1960s, with the creation of ARPANET, the Advanced Research Projects Agency Network, the concept of Internet security was not considered. Rather, usability was the main focus of those developing the Internet. Since then, security of Internet-based devices has not kept with the fast pace of innovation. Further, when designing the new products, security is often discussed and implemented only after those devices are manufactured. Unfortunately, IoT

devices also fall into the same dilemma of manufacturers favoring usability over security. Although IoT devices are heavily tied to network transportation, there is still little security policy or government regulation restricting the manufacturers. This leads to trading security for usability and the consumer's out-of-the-box satisfaction. Consequently, poor security on IoT devices has exposed owners to huge security risks.

Regarding privacy concerns on IoT devices, Maddan et al. [24] focused on privacy breaches over information linkage. During the time that IoT devices are collecting data to provide services, the data could be re-used or shared by different services and stakeholders. Unfortunately, customers do not have control over who can view or re-use their personal information in the IoT ecosystem. Further, Ziegeldorf et al. [3] pointed out that surveillance cameras are increasingly integrated and used in non-security contexts. These surveillance cameras can enable possible facial databases for non-governmental parties; even worse, Wi-Fi cameras can be installed by adversaries with malicious intent to spy on specific targets. The authors concluded that privacy is a constant challenge that requires the designer's and regulator's foresight [3]. Pasquier et al. [28] focused on privacy policy over IoT devices. Although there are suggesting policies and laws proposed by European regulators and US Federal Trade Commission, those concerns and suggestions have not yet been implemented as industry standards. The paper suggested to enforce auditing to compliance with privacy policy.

1.1 Recent attack targeting Wi-Fi camera

As mention above, while IoT devices bring convenience to the owners, they also create security risks. Weak security mechanisms allow adversaries to exploit IoT devices and even have total control over with admin privileges. In 2016, the Mirai malware took advantage of the weak password settings of IoT devices and hijacked 3.5 million Wi-Fi cameras and home routers [19]. The infected devices were located globally, including most of the countries in Europe, Asia, and North and South America [6]. The malware was not sophisticated but simply attempted to use the 60 most common default factory passwords and login credentials

associated with those IoT devices to gain admin privileges. This simple yet powerful brute-force attack allowed the adversaries to secretly gain admin permissions of the targeted Wi-Fi cameras and routers. The successful Mirai malware has built a botnet of millions of IoT devices, a large army that is ready to launch attacks at any time. The Mirai botnet, group of hijacked Wi-Fi cameras and routers, is also used to launch some of the largest distributed denial of service (DDoS) attacks. DDoS attacks are commonly used by adversaries to block online services from users by overwhelming them with significant amounts of network traffic from multiple sources. A DDoS attack on French hosting company OVH in 2016 reportedly peaked at 1 Terabit-per-second (Tbps), making it the biggest DDoS attack ever reported at the time [2]. The founder of OVH further reported that the attack was launched by IoT devices including hacked CCTV cameras and Wi-Fi cameras, believed to originate in the Mirai botnet. These incidents demonstrate the security vulnerabilities and potential risks to owners of IoT devices. In addition to belonging to the Mirai botnet, the Mirai malware also poses privacy risks for the owners of the affected Wi-Fi cameras. With the poorly implemented security on devices, adversaries are able to hijack Wi-Fi cameras and monitor the owner's behaviors without being detected. Even worse, some of the affected Wi-Fi cameras are not able to perform factory reset due to the poor design of the firmware. This leaves the owner no choice but to purchase a new Wi-Fi camera. Poorly designed Wi-Fi cameras not only bring more chaos to the existing cyber world but also pose serious privacy issue to the owners, including the possibility of being secretly monitored. The Mirai botnet represents a severe warning of how important security is within the IoT devices.

1.2 Attacker models

Wi-Fi cameras are easy targets for adversaries who attempt to exploit their vulnerabilities. This section further discusses some of the potential malicious attacker models on those Wi-Fi cameras. One attacker model extended the default password vulnerability exposed by the Mirai malware[19]. Adversaries can easily take control of these Wi-Fi cameras by launching a brute-force attack using combinations of default credentials. By gaining the admin privilege

of the victims' Wi-Fi cameras, adversaries can remotely monitor victim's behavior without notice. In second attacker model, a Wi-Fi camera is installed in the environment, such as a hotel room or meeting room. The adversaries can then monitor the victims' behavior secretly. This is a serious privacy concern, given that multiple incidents have already been reported. Hidden Wi-Fi cameras are reported to be found in Airbnb rental places and those incidents are happening more often [8]. Due to the lack of proper security and updates, Wi-Fi cameras are an easy target for adversaries. These two scenarios demonstrate how Wi-Fi cameras can be used to subvert their intended purposes and to stealthily monitor victims' activity.

1.3 Current state-of-the-art research regarding IoT devices

Regarding IoT devices within the home environment, research has shown positive results with respect to identifying those devices or studying their behaviors [4, 25, 34]. Proposed strategies include machine learning, traffic analysis, and timing analysis. Apthorpe et al. [4] discovered that network traffic of IoT devices has the potential to reveal user interactions. The study pointed out that Wi-Fi cameras tend to leave visible peaks in the network traffic, or timing characteristics, while they detect a fair amount of user interactions. Moreover, those traces are predictable and can be easily reproduced by the researchers. Siby et al. [34] implemented an IoTScanner that detects IoT devices within the home network. Recording the network traffic, the IoTScanner then stores the recorded data in a database and further performs analysis on the send-to-receive ratio. The send-to-receive ratio represents the ratio of sending and receiving network traffic per server. Analyzing the send-to-receive ratio allows the IoT Scanner to successfully detect IoT devices. Rather than analyzing the network traffic, Meidan et al. [25] proposed using machine learning models to identify IoT devices. With the full information of the network traffic, the well-trained machine-learning model is able to identify IoT devices within a LAN.

1.4 Proposed methodology

Considering the two attacker models introduced in section 1.2, this paper proposes a novel method to detect hidden Wi-Fi cameras in a pervasive environment where various electronic devices are communicating using different wireless and wired protocols. The proposed methodology utilizes the previously discovered characteristics of Wi-Fi cameras, i.e., the remained timing characteristics [4]. Using the discovery of timing characteristics, this paper proposes to detect hidden Wi-Fi cameras with timing analysis. Timing analysis is carried out by calculating the Correlation Coefficient (CC), Dynamic Time Warping (DTW), Kullback-Leibler divergence (KLD), or Jensen-Shannon divergence (JSD) between the two data streams. The two data streams are collected from the network traffic and the recorded video from a mobile phone. CC is a measure to calculate the relationship between two variables. The higher the CC, the stronger the relationship they have. DTW, in contrast, is used to measure the similarity of two temporal sequences, which may be vary over time. The calculation of the DTW gives the similarity of the two input sequences. While the proposed methodology relies on user interactions to generate timing characteristics in network traffic, an environment with movements is considered as our testing baseline. The setting of the experimental environment is a 900-square-foot room with a minimum of two individuals moving in the space. This method differs from current research [4, 25, 34], since it does not require full information from the network traffic. Rather than simply acknowledging the existence of the Wi-Fi cameras, this method is able to discover live streaming Wi-Fi cameras in real time.

The metrics of the proposed approach are based on the accuracy rate of the detection. The accuracy rate of the proposed methodology shall reach at least 80%. True positive rate and false positive rate of the detection shall also reach at least 80%. The overall F1 score of the proposed methodology shall reach at least 90%. The detection shall also be reliable in the setting, specifically, a 900-square-foot room with movements.

1.4.1 Contribution

The contribution of this paper is a novel methodology that is able to detect hidden Wi-Fi cameras with a mobile phone. The proposed methodology has not been studied in any form in the research area focused on detecting hidden Wi-Fi camera. This paper serves as the first research to proposed detecting hidden Wi-Fi camera with timing analysis comparing network traffic and recorded video, specifically, by calculating Correlation Coefficient, Dynamic Time Warping, Kullback-Leibler divergence, and Jensen-Shannon divergence. Moreover, this project had successfully tuned an Artificial Neural Network (ANN) specifically on identifying hidden Wi-Fi camera based on the network traffic. This innovative Wi-Fi camera detection methodology will benefit users who want to discover hidden cameras in an alien environment.

This paper is organized as follows: Chapter 2 describes previously conducted research and their contributions and limitations, chapter 3 introduces the proposed methodology and the proposed detection models. Chapter 4 discusses results of the proposed detection method. Finally, Chapter 5 concludes this paper with a discussion of limitations and future work.

1.5 Terminology

1.5.1 Network

Network infrastructure can be categorized into two models: 5-layer model (TCP model) and the 7-layer model (OSI model). This project uses the TCP model to describe the network traffic. Any network transportation between two entities involves constantly sending packages back and forth to transfer data. Those packages, also called packets, further go through 5 layers of encapsulation and decapsulation. The 5 layers in the TCP model are: 1) Physical layer, 2) Data link layer, 3) Network layer, 4) Transport layer, and 5) Application layer. Each layer serves specific purposes and is used to identify source and destination devices by identifying different information contained inside the packet. Physical layer refers to the actual hardware wire transferring information; Data link layer refers to the MAC address, which is a hard-coded address that is used to uniquely identify any device; Network layer re-

ferred to the IP address, which is a dynamic address given by the network router; Transport layer refers to the protocol usage of the transportation; and Application layer refers to the applications or services the client is currently using. Network traffic can be separated into multiple network streams. Each network stream represents single communication channel between two devices. Network stream refers to a data stream that contains all the network packets between two communicating entities.

Chapter 2

RELATED WORKS

Related research has been focused on identifying services, applications, websites, and connected devices with various detecting mechanisms. Since network traffic contains critical information regarding communicating entities and ongoing communications, most of the research is concentrated on detecting targets by utilizing the data embedded within network traffic. Other research is focused on utilizing machine learning to improve identification accuracy. Other studies have introduced novel approaches to perform timing analysis identifying targets. Figure 2.1 below groups previously conducted research based on their detecting mechanisms.

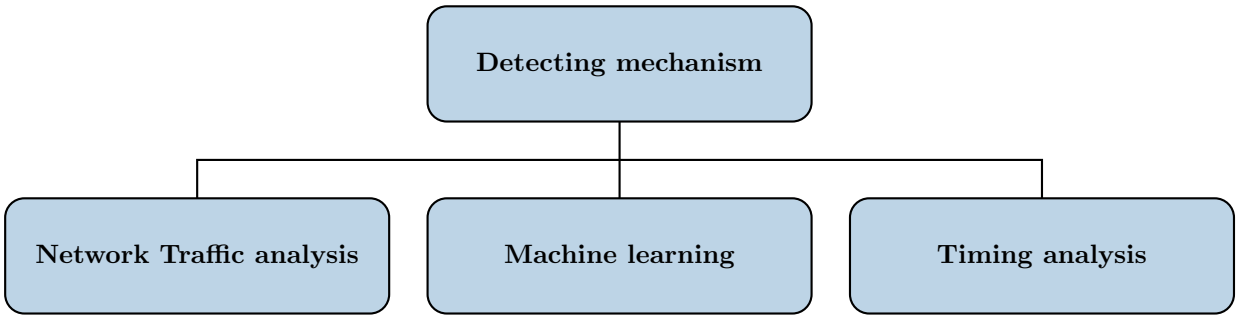

Figure 2.1: Illustration of different detecting mechanisms researchers have proposed.

Previously proposed detecting mechanisms can be categorized into 3 categories: 1) Network traffic analysis, 2) Machine learning and 3) Timing analysis. All of these detecting mechanisms analyze particular features from the collected data to identify certain targets such as websites, services, and connected devices. Various features, along with different approaches, have been proposed to detect target services and activities. Each category is discussed, with their advantages and limitations, in the following sections.

2.1 Network traffic analysis

Network traffic analysis has been widely adopted by corporations and governments to monitor communications and detect abnormal behaviors. Since the Internet is used by computers and Internet-based devices to transfer data, network traffic analysis is certainly an excellent approach to identify devices and activities. Several previous studies focused on analyzing network traffic to identify certain targets; they are further discussed in the following paragraphs.

Focusing on network traffic analysis, Geer et al. [15] did a thorough study discussing the nature of traffic analysis. The authors stated that traffic analysis is a powerful tool to identify targets because of the network traffic volume that is generated daily. The research listed several features of the network traffic, such as frequency, volume, and timing, that enable attackers to identify particular patterns. Moreover, encryption over network traffic does not prevent adversaries from studying those features. The findings of this research allowed adversaries to identify certain behaviors and services from the network traffic. Looking into encrypted network traffic, Coull et al. [10] researched network traffic analysis for Apple iMessage. The authors looked into the volume of the encrypted network traffic that is being transferred. The study found that adversaries can successfully learn the victim's actions, language used, and the length of the messages, with 96% accuracy. This study demonstrates that one can discover such information by simply analyzing the volume of the network traffic. It also raises the issue that current infrastructure of the Internet is vulnerable to attempts to discover partial information, even when security measures are implemented.

Regarding remotely observed network traffic, Siby et al. [34] studied one attribute, specifically, the send-to-receive ratio, of the network traffic. Focusing on an IoT-rich environment and privacy concerns, the research proposed and implemented the IoTScanner, which can discover existing wireless infrastructure. The IoTScanner was implemented on a Raspberry Pi with a traffic interceptor that is used to collect network traffic. The embedded traffic analyzer on the Raspberry Pi further extracted and analyzed information from the Link layer

frame header, such as the source and destination address, frame type, SSIDs, size of the frame, and the channel number. By analyzing the numbers of Frames, mFrames, cFrames, and dFrames; network traffic volume; and send-to-received ratio, the scanner can identify IoT devices passively. The IoTScanner further presented the corresponding relationship of the discovered existing wireless infrastructure on a hand-held device. The study demonstrates that one can scan and analyze the IoT environment passively without previous knowledge. The limitation of the IoTScanner is the detection time and the extra hardware and software equipment required to perform IoT detection. The detection time of the IoTScanner can vary between 60 seconds to 100 seconds.

To filter noises within remotely observed network traffic, Gong et al. [18] studied the feasibility of Dynamic Time Warping (DTW) on network traffic patterns. DTW is an algorithm that measures the similarity between two sets of time-series data. It has the attribute to map data samples which may differ in time or speed. The study showed that website fingerprinting is applicable, even with noisy network traffic, by applying DTW to traffic analysis. DTW allowed the research to fingerprint certain testing websites based on the DTW similarity score of the network traffic. However, some of the testing websites were virtually invulnerable to the fingerprinting method. Due to the highly variable traffic patterns of those websites, this proposed detection mechanism was not able to observe any true positive results. This is because the inconsistency of the traffic patterns downgraded the overall performance of their proposed method.

2.2 Machine learning

In order to identify particular services efficiently without sacrificing accuracy, researchers have proposed to detect targets based on supervised and unsupervised machine learning. Supervised machine learning takes advantage of large numbers of labeled datasets to gain knowledge about the data. It then predicts the new input based on the knowledge learned from the dataset. A well-trained supervised machine-learning model with a large labeled dataset is good at classifying new input as belonging to a preexisting label in a short amount

of time. On the other hand, unsupervised machine learning does not require any labels in the dataset. Unsupervised machine learning is able to cluster dataset into smaller groups based on particular attributes. The accuracy of unsupervised machine learning depends on how well the model can distinguish the data based on the selected attributes.

Moore et al. [26] focused on identifying applications based on network traffic and the header information with Nave Bayes machine learning. While the traditional Nave Bayes model yields 65% accuracy, the study proposed to utilize discriminator selection and a dimension reduction technique to refine the model. Applying those two techniques has improved the accuracy rate to 95%. Moreover, the model is applicable to network traffic which contains less information. This research has showed the possibility to classify services based on partial information of the network traffic.

Looking into identifying IoT devices based on the network traffic, Meidan et al. [25] proposed to use supervised machine learning to classify the network sessions of both IoT and other electronic devices. A multi-stage approach was introduced in which two machine-learning-based classifiers are applied to classify IoT devices from the network traffic. Their proposed methodology detected IoT devices with a high percentage of accuracy of 99.28%. However, their classifier required full information from the network traffic, including everything from MAC address to IP address. This approach may, however, failed in identifying IoT devices in the situation which network traffic is fully encrypted. The proposed approach required extra computation resources and more time to train the two machine-learning classifiers compared with other non-machine- learning-based approaches.

Regarding identifying targets based on timing differences, Giannoula et al. [16] proposed to apply DTW with unsupervised machine learning. The study aimed to group patient disease trajectories based on their temporal patterns. A total of 17,800 trajectories were plugged into the DTW clustering algorithm. The threshold-based approach was utilized to cluster the data samples based on the DTW distances. Their proposed clustering algorithm is able to successfully identify disease patterns and statistically significant disease associations. The authors stated that the proposed methodology, which applies temporal assessment of trajec-

tories, serves as an important step to developing a disease-prediction system in the future. The study has showed the feasibility of identifying targets by combining the threshold-based approach with a machine-learning-based classifier. In this paper, we want to elaborate their findings to build a detection classifier using a threshold-based approach, which will compare the DTW distances of the temporal sequences.

DTW is also utilized by Ding et al. [12] to identify human actions based on postures, using spectral clustering learning. Postures were defined and extracted from screw motion between relative rigid bodies in a 3D environment. These 3D rigid motions were further grouped by a hidden Markov model (HMM). The HMM model grouped those 3D rigid motions based on their DTW distance, which allowed the model to perform human action identifications. Based on the data samples collected from the UT-Kinect dataset, the HMM model achieved an overall recognition rate of 91.5%, while previous research achieved 88.5% and 90.92%, respectively. Their proposed approach has achieved promising results in comparison with other human-action recognition algorithms.

Li et al. [23] proposed to apply DTW to identify wireless devices based on RF signals. Specifically, they focus on unique identification of wireless devices. The research addressed the case where individuals carried more than one wireless device, e.g. multiple entrance cards. The time-series data are collected in a simulation of 300 wireless nodes moving randomly and transmitting signals in an unsynchronized pattern. After applying DTW to calculate the similarity between time series, the research further utilized spectral clustering to reduce computational complexity. The result of the research showed that it is able to uniquely identify wireless devices with a run time of approximately 100 seconds.

Machine learning is an effective tool for analyzing a large dataset with multiple features. Both supervised machine learning and unsupervised machine learning have shown the potential to identify targets, even with temporal patterns. While previous research focused on different types of datasets, their proposed machine-learning classifiers were able to achieve 90% or above as the accuracy rate. Research conducted by Meidan et al.[25] has demonstrated that it is possible to identify IoT devices by analyzing the rich information

contained in the network traffic using supervised machine learning. In addition, Li et al.[23] showed that DTW is able to identify targets based on timing temporal differences. In this research, we want to extend their findings and propose to identify hidden Wi-Fi cameras with supervised machine learning based on the DTW distance.

2.3 Timing analysis

While most computers and Internet-based devices transfer data through the Internet, adversaries are able to perform timing analysis on network traffic to detect targeted services. Even if the network traffic is encrypted, the time intervals within network packets are exploitable by adversaries [17, 33]. Thus, timing analysis is utilized by researchers to identify targets based on unique patterns of the time-series data.

Regarding time-series data, DTW has been applied by previous research to identify unique patterns. Kim et al.[20] studied identifying cycle-times of construction equipment. The study applied DTW on IMU signals as features for the activity classification. A smart phone with an inbuilt IMU sensor was mounted to the construction equipment to collect raw data from an accelerometer and a gyroscope. The authors further stated that DTW aids identifying the rotation of cabin. It solves the issue in previous research of distinguishing mixed activities. Compared with traditional machine-learning approaches, adopting DTW achieved an higher accuracy rate of 91.83%. Applying DTW with a threshold-based approach improved the accuracy rate of previous methods by 8.98%.

DTW has also been successfully used in the health and biology fields, showing excellent performance in identifying activities with time-series data. Chou et al. also studied identifying IMU signal-based activities [7]. This study focused on detecting risky activities of people under healthcare. While IMU signals often vary in time or speed, DTW is a promising candidate to measure their non-linear relationship. Data collection were conducted with a Bluetooth IMU device on health-under-cared people. Surprisingly, the researchers' DTW model outperformed other machine-learning models in both accuracy rate and model preparation time.

Aras et al. [5] have applied DTW to identify the limits of respiratory cycles based on the similarity of the patterns of lung sounds. Lung sounds were recorded in a hospital environment with ambient noise as the raw data. The model applied DTW onto the raw data to identify cycles of the respiration. Their model is able to identify seven different kinds of respiratory sounds based on DTW distances, with a mean absolute rate range from 3.2% to 5.6%. The proposed method fulfills the need of automatically determining limits of respiratory cycles without requiring manual confirmation. This study showed that DTW is able to classify multiple activities based on the difference in DTW distances.

In regard to detecting fall events using a microwave Doppler sensor, Shiba et al. [32] proposed to apply DTW to classify particular frequency patterns. Microware Doppler sensor is a non-wearable solution to fall event detection for elders, which provides ease of use without requiring constant recharging of the wearable devices. A total of 624 datasets (312 fall events and 312 non-fall events) were collected and studied. Considering time-series transitions on those datasets, DTW is utilized to classify fall events. The proposed model achieved a classification accuracy rate of 97.76%. This study has shown the feasibility of applying DTW to detect certain patterns in real time.

To detect certain activities based on time-series data, Singh et al. [35] proposed to detect driving behaviors based on data collected from a mobile phone. The collected data from the mobile phone including accelerometer, gyroscope, gravity, and GPS. These data are time-dependent, since there exist multiple factors, such as road and vehicle conditions, that might create variations. While the data are time-dependent, DTW served as a effective algorithm to detect non-linear relationships. Their proposed method outperformed existing machine-learning-based and threshold-based techniques, with a detection rate of 100% for sudden braking and 97% for left and right turns. Simple DTW calculation also required less time and resources; further, it also did not require any extensive training compared with machine-learning-based techniques.

Addressing the high variance of timing patterns in network traffic, Dimopoulos et al.[11] proposed to use DTW to identify anomalies in network traffic. The proposed algorithm

first groups similar instances based on the DTW distances, then performs classifications of contexts based on the scoring of a specific time window. This research used the dataset from the FCC's Measuring Broadband America project to evaluate the proposed algorithm. The proposed algorithm achieved higher accuracy rates compared with existing machine-learning-based approaches. In this paper, we want to extend their findings and apply DTW onto the collected network traffic to group testing samples.

Feghhi et al. [13] researched the effectiveness of timing-based attacks against encrypted network traffic and were able to infer the existence of web pages more than 87% of the time. Another study demonstrated that performing timing analysis reveals victim nodes within Tor [27]. The timing characteristics, which remained in the network traffic as they traveled through Tor, allowed the researchers to reveal targeted entities and servers. In another study, using only meta-data, such as a TCP/IP packet header and the send/receive rate, Apthorpe et al. [4] successfully determined the timings when a security camera detected movements. They were also able to ascertain whether the security camera was actively monitored. This literature has clearly demonstrated the feasibility of detecting IoT sensors, even if the network traffic is encrypted.

Apthorpe et al. [4] performed experiments on IoT smart home devices. They discovered that the network traffic of those devices often revealed potential information about user interactions. Based on the sending/receiving rates of the streams, they were able to map live traffic to user behaviors. This research indicates that the network streams of IoT devices have certain attributes that are controllable by the users. We expect to adapt their findings to build a novel IoT sensor detection method based on certain movement interactions. A timing analysis on a low-latency network has also been discussed [27, 33]. Both studies have pointed out that the timing characteristics of network traffic tend to be remained. We intend to extend their findings to perform statistical analysis on the timing characteristics of Wi-Fi cameras.

Previously conducted research has determined that IoT traffic reveals device owners' behaviors or can be used to discovered communicating entities. We propose to detect Wi-Fi

cameras based on the timing characteristics in the network traffic of a Wi-Fi camera.

Chapter 3

METHODOLOGY

This chapter introduces the proposed framework and the proposed detection models, derived from the framework, used to identify hidden Wi-Fi cameras within the environment. Details of implementation and computer language use are further discussed in each of the subsequent sections. Goals, evaluation metrics, and usage scenarios are discussed at the end of this chapter. Section 3.1 explains the proposed methodology and presents the proposed framework. Section 3.2 introduces the two proposed models. Section 3.3 discusses the goal and the metrics used to evaluate the designed models. Section 3.4 explains trade-offs of the design decision. Lastly, section 3.5 introduces the usage scenario and practical environments of the proposed methodology.

3.1 Proposed detecting methodology

This research is an extension of our previous research. Our previous study proposed to detect hidden Wi-Fi cameras actively by using the flashlight from a mobile phone to interact with the physical environment [22]. Those flashes forced the Wi-Fi camera to generate abnormal peaks in network traffic, which were then used to perform timing analysis. The result of the timing analysis was further used to determine whether a particular network stream is a Wi-Fi camera. The detection accuracy rate was able to achieve 90%, with a false positive rate of 5%. Nevertheless, the detection method works based on an assumption of the testing environment; specifically, the detection method required an environment in which little to no movement was happening. We assumed that movements of objects might affect the accuracy rate, since the flashes from the mobile phone might get blocked, and movements tend to create noise in the network traffic.

In this paper, we look into this problem and propose a novel methodology to detect a hidden Wi-Fi camera passively. Rather than requiring an environment with little to no movement, the assumption of the proposed methodology is to have an environment with fair amount of movements. As in the previous paper, the proposed methodology utilizes a feature from a mobile phone to successfully detect hidden Wi-Fi cameras. Mobile phones are a prolific technology in our modern age. Therefore, the mobile phone platform is ideal because it is easily accessible to the general population. Thus, utilizing features from a mobile phone to perform detection can benefit general users.

With the goal of detecting hidden Wi-Fi cameras, this paper proposes to detect hidden streaming Wi-Fi cameras using timing analysis. Timing analysis has been utilized by various previously published detection mechanisms to successfully discover targeted websites or devices [4, 27, 13]. Our previous study used flashes, actively interacting with the environment. In this paper, we propose detecting the Wi-Fi camera passively by recording the environment. The detection mechanism will analyze timing characteristics that exist in the recorded video and the network traffic of the Wi-Fi camera.

Due to the design of the video compression algorithms used by Wi-Fi cameras and mobile phones, those devices tend to generate more data if there are movements in the recording environment. The H.264/ MPEG-4 video compression algorithm was first introduced by [1]. One of the improvements of the H.264(MPEG-4 Part 10) is the ability to reduce the size of a video file, which requires less network bandwidth and storage space. The H.264(MPEG-4 Part 10) achieved this by removing unnecessary information, specifically, the unchanged pixels between frames, during the encoding process. Instead, the algorithm only encoded the changing pixels with respect to reference frames. Thus, more movements occurring in the environment forced the Wi-Fi camera and the mobile phone to generate more data in the form of network traffic and video frames.

For video files, timing characteristics are observed as the peaks between video frames in the recorded video. Since mobile phones use the MPEG-4 video compression algorithm to encode the recorded videos, these video files tend to contain more data when there is

a significant amount of movements in the environment. This behavior is a result of how MPEG-4 encodes the video frames. The MPEG-4 algorithm encodes all the changing pixels that differ from frame to frame. Thus, more movements generate larger data peaks in video files.

For Wi-Fi cameras, the timing characteristics are represented by the observed peaks in the network traffic. Wi-Fi cameras also tend to generate peaks in the network traffic while capturing movement [27]. This is the default behavior of the Wi-Fi cameras based on the video compression algorithm they use. The H.264 algorithm, a block-oriented motion-compensation-based video compression standard, is utilized by Wi-Fi cameras to transfer data efficiently. This particular standard only records motions between frames, in order to reduce storing overlapping information. Thus, a large amount of movement forces the Wi-Fi camera to generate and transfer large amounts of data, which creates peaks, the timing characteristic, in network traffic. Furthermore, the tendency of timing characteristics to remain within the network traffic has previously been discovered and studied[27, 33] . The tendency for the timing characteristics to remain creates the possibility to perform timing analysis.

Those timing characteristics that remain in recorded video and network traffic are essential to performing timing analysis, which includes statistical analysis on the data. Statistical analysis is able to analyze the variables and calculate the similarity between network traffic and the recorded video. The similarity of the recorded files is then used to identify possible hidden Wi-Fi cameras. Based on the proposed methodology, this paper further proposes a detection framework. It categorizes and presents the major steps of the methodology. The proposed detection framework is presented in Fig. 3.1.

The proposed framework has four major steps. The first step is to monitor the environment digitally by recording video and network traffic simultaneously. The recorded files contained timing characteristics that are essential to identify Wi-Fi camera. The second step is to extract a feature, specifically, bytes-per-second, from both the video file and the recorded network traffic file. This results in data streams that are further used to perform statistical

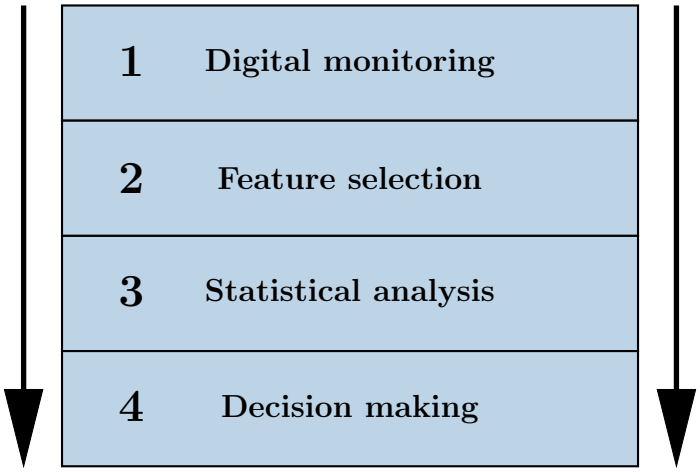

Figure 3.1: The proposed detection framework.

analysis. The third step is to perform statistical analysis, specifically, calculating the Correlation Coefficient (CC), Dynamic Time Warping (DTW) distance, Kullback-Leibler divergence (KLD), and Jensen-Shannon divergence (JSD) on the bytes-per-second data streams. CC is a numerical measure to calculate the relationship between two variables, and DTW is an algorithm for measuring the similarity between two temporal sequences. The last step is to identify possible hidden Wi-Fi cameras by comparing the similarity score with the threshold. Detailed descriptions of each steps and corresponding implementation are presented in the sections below.

3.1.1 Step 1: Digital Monitoring

Digital monitoring is the first step in gathering data from the network traffic and the mobile phone. Network traffic is monitored and sniffed while the mobile phone is recording the environment with the back camera. In this step, the simultaneous recording of the network traffic and the mobile phone is performed. The Digital monitoring can be separated into 1) Network Monitoring and 2) Video Recording.

Network Monitoring

To retrieve information from the Wi-Fi camera, the network is being monitored and recorded as one of the input data. In order to record the network traffic, a network sniffing tool is being used. Wireshark, an open source network sniffing tool supported in various platforms, is used in this paper to sniff the network traffic. The output of the recorded network traffic is further saved as a PCAP file. In the experiments, Wireshark is used on a Macbook Pro with macOS High Sierra 10.13.4 to perform network monitoring. The version of the Wireshark software installed on the laptop is 2.4.2 and the Network Interface Card installed on the laptop is AirPort Extreme (0x14E4, 0x170) with firmware version of Broadcom BCM43xx 1.0 (7.77.37.29.1a7).

With the assumption of having movements within the physical environment, Wi-Fi camera will generate peaks, visible timing characteristics, in the network traffic based on their default behavior. It is not feasible for Wi-Fi cameras to transfer all the recording through the internet, as this would consume too much memory and network bandwidth. Instead, Wi-Fi cameras use video compressing algorithms to determine what to transfer through the internet. H.264 is a widely adopted algorithm to perform video compression on Wi-Fi cameras. The H.264 video compressing algorithm detects any changing pixels and colors during the monitoring and only transfers these data, rather than the whole video frame. Thus, Wi-Fi cameras tend to generate peaks in network traffic when detecting movements or pixel changes.

Video Recording

To retrieve data from the environment that is monitored by the Wi-Fi camera, video recording is performed from the back camera of the mobile phone. The video recordings on the mobile phone also use a video compression algorithm to shrink the size of the video file. Mobile phones used the MPEG-4 algorithm to compress the video. Similar to H.264, the MPEG-4 algorithm only stores changing pixels and color in between video frames, to reduce

storing space by minimizing redundant information. This paper uses a Motorola-Z, with the OS version Android 8.0.0, to perform the experiments. The recorded video files have a resolution of 720p or 1080p and are all in the length of one minute. The videos are encoded as MP4 files with audio support.

Background sound recording was considered as one of the approaches to perform digital monitoring. In this approach, the mobile phone recorded the background sound of the environment as one of the input data. However, we intimately rejected this method, based on its poor performance compared with other approaches. Further, in the experiments, we found that the Wi-Fi camera required a high pitch or sudden loud background sound to generate visible peaks in the network traffic. Thus, this method might not be applicable to real-life situations, such as hotel or meeting rooms.

3.1.2 Step 2: Process for Features

After the recording is completed, features are extracted from the recorded files to form data streams. Two data streams are further extracted from the recorded network traffic and the video file. While the recorded video is encoded as an MP4 file and the recorded network traffic is saved as a PCAP file, it is necessary to extract the same feature from the recorded files to perform statistical analysis. Bytes-per-second, a shared feature in both MP4 and PCAP files, is extracted from the recordings.

The first data stream is extracted from the recorded video file. The recorded video from the mobile phone goes through FFmpeg to generate the bytes-per-second stream. FFmpeg is a free software that processes multimedia data. It first released in 2000, it has been widely used by software developers to extract information from audio or video files. More than 100 codecs are included in the software, which makes FFmpeg a effective tool to perform format transcoding, editing, or video scaling. This study used Libavformat, an audio and video transcoding program included in FFmpeg, to extract frame information from the MP4 files. The extracted frames contained information such as time and size, which are sufficient to generate a bytes-per-second stream.

The second data stream is extracted from the recorded network traffic, the PCAP file. The PCAP files goes through PyShark to generate a bytes-per-second stream. PyShark, a wrapper of Tshark, is a simple Python library that decodes the Wireshark PCAP file. Tshark is a terminal-oriented tool designed to capture and display network traffic. This tool also supports extracting time and size from captured network packets. Time and size are essential to generate a bytes-per-second stream.

The process of feature-extracting is implemented in Python, a high-level computer language that can be executed directly by the computer without compiling. Python was chosen for this project because of its easy readability and adaptability; it can be executed easily in multiple operating systems such as Windows, macOS, and Linux. This allows for rapid development and distribution across different platforms smoothly. Python has also been widely used by researchers in various fields for its ease of use and ability to adapt fast developing.

3.1.3 Step 3: Perform statistical analysis

After the byte-per-second streams are extracted, we further conduct statistical analysis to calculate the relationship between the two data streams. Before performing any statistical analysis, data normalization is utilized to shorten the performance time of the analysis. In this project, Correlation Coefficient (CC), Dynamic Time Warping (DTW), Jensen-Shannon divergence (JSD), and Kullback-Leibler divergence (KLD) are selected to measure the relationships between the two data streams. CC is a statistical measure to calculate the correlation between two variables, and DTW is another algorithm used to measure similarity between two temporal sequences. KLD calculates the differences between two normally distributed data samples, and JSD measures the similarity between two probability distributions. The following sections describe each algorithm in detail.

Data normalization

Data normalization is performed to standardize the range of the variables in byte-per-second streams. This pre-processing step eliminates the effect of particular outliers and prevents

certain objective algorithms from failing. The re-scaling process also speeds up the processing time of the applied algorithms. This study utilized feature scaling to perform data normalization. Feature scaling re-scales all values in the data stream into the range between 0 and 1. The formula of the feature scaling is presented in Equation 3.1.

$$z = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.1}$$

The equation represents the general formula of the feature scaling. x represents the original value of the variables, and z is the re-scaled value. To re-scale the variables into the range of [0,1], every original x minus the minimum x is divided by the total range of all the variables, maximum x minus minimum x. Minimum x represents the minimum number of all variables, and maximum x represents the maximum number of all variables. The result of the feature scaling then feeds into other proposed measuring algorithms to calculate the similarity.

Correlation Coefficient

Pearson's Correlation Coefficient is one of the measuring algorithms that is calculated for the two data streams. The CC was first presented by Karl Pearson in 1896 to derive the best value of the correlation coefficient[29]. The result of the CC is an measure of how correlated two input variables are. The range of the result is between -1 and 1. The value of 1 indicates a strong positive relationship and -1 indicates a negative relationship. In addition to the result value, P-value is also given by calculating the correlation coefficient. P-value is a metric used to show the probability of finding a current result if the correlation coefficient is zero. If the p-value is lower than 5%, the result value of the Correlation Coefficient is stated to be statistically significant. The formula of the CC is presented in Equation 3.2.

$$r = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sqrt{\sum(x - \bar{x})^2 \sum(y - \bar{y})^2}} \tag{3.2}$$

The equation represents the formula of the correlation coefficient. In this equation, x

represents the variables in the first bytes-per-second stream and y represents the variables in the second bytes-per-second stream. The equation first calculates the total sum of all the multiplications between every input x minus the average x and every input y minus average y. The output is then divided by the square root of the multiplication between total sum of the square of every x minus average x and total sum of every y minus average y. The resulting r value is the final result for CC. After the r value is calculated, the t value is further calculated to retrieve the p-value. The formula of the t value is given as Equation 3.3.

$$t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}} \tag{3.3}$$

The equation presents the formula of t-value, where n is the total number of paired variables. t value is calculated by multiplying the correlation coefficient by the square root of the total number of paired variables minus two. The output is then divided by the square root of one minus the correlation coefficient squared. Finally, p-value is calculated with $2r(T > t)$, where T follows a t distribution of $n - 2$ degree of freedom. An illustration of how CC calculates relationship between two data streams is presented in Figure 3.2.

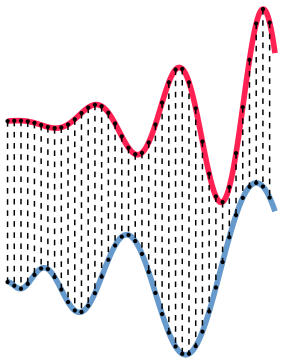

Figure 3.2: Illustration of CC on two data streams.

Dynamic Time Warping

Besides the Correlation Coefficient, DTW is another algorithm to calculate similarity between two temporal sequences. DTW is first introduced by Sakoe [30] for speech recognition. It aims to find the best alignment of two sequences with a non-linear time wrap. An estimated optimal matching, the DTW distance of two input sequences is then given by the algorithm. DTW distance between two streams indicate the strength of their similarly. The lower the DTW distance is, the higher the similarity between the two streams. Since DTW takes into consideration timing differences, it is more suitable than the Correlation Coefficient in the situation where the recordings do not start at the same time. Considering two time series sequences:

$$\text{Sequence } A = a_1, a_2, \cdots, a_n$$
$$\text{Sequence } B = b_1, b_2, \cdots, b_n$$

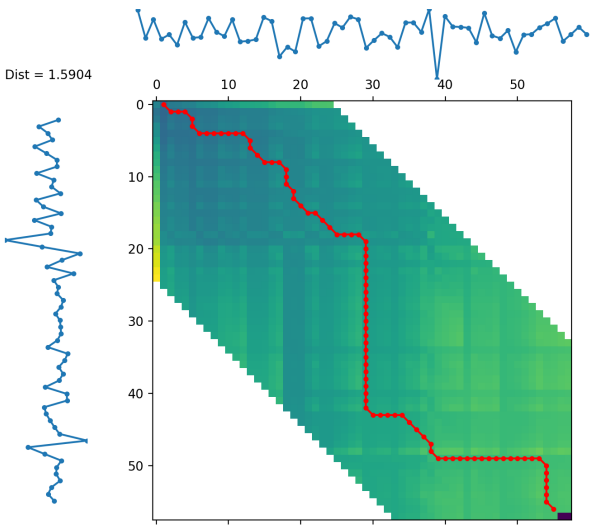

Figure 3.3: Warping matrix of Sequence A and Sequence B.

DTW began by putting the two sequences on the side of the grid in Fig. 3.3, one on the top and another on the left. Each cell is considered as a distance measure, which compares the corresponding elements of the two sequences. An optimal path, which is the best match

between the two sequences, is given in the figure by calculating the minimum total distance. The minimum total distance involves finding all the possible routes from all the cells in order to compute the DTW distance. Figure 3.4 illustrates how DTW computes between two temporal sequences.

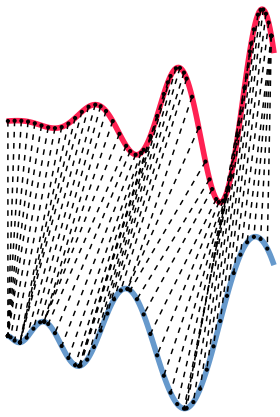

Figure 3.4: Illustration of DTW on two temporal sequences.

This paper utilized FastDTW [31] to perform DTW calculation in Python. FastDTW is able to compute DTW in the complexity of $O(N)$, rather than $O(N^2)$ as in the original implementation.

Kullback-Leibler divergence

The Kullback-Leibler divergence (KLD) was first introduced by Kullback and Leibler in 1951 [21]. It is an asymmetric measure used to compute the difference between two probability distributions. The KLD computes the information loss between the two probability distributions. The equation of the KLD is presented in Equation 3.4.

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{3.4}$$

The equation presents the formula of KLD between two distributions P and Q. For all the i in P and Q, KLD calculates the sum of every $P(i)$ time *log* function of $P(i)$ over $Q(i)$. It is a measure to calculate how likely it is that one can find sample i in between $P(i)$ and

$Q(i)$. A resulting value larger than 1 indicates P is the more likely model for i; on the other hand, a resulting value smaller than 1 indicates the opposite. An illustration of how KLD changes when two probability distributions have greater distances is presented in Figure 3.5.

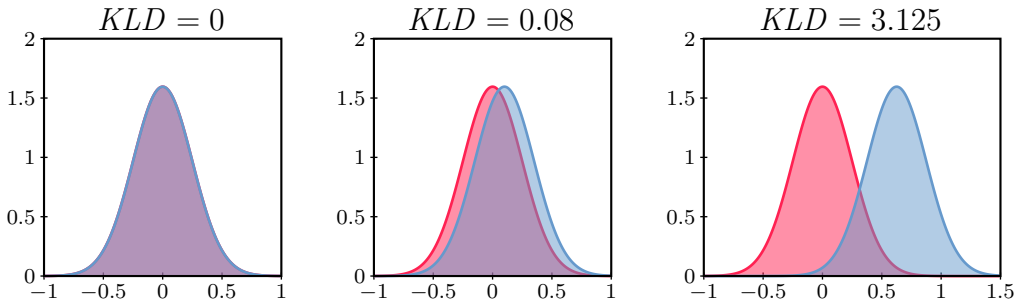

Figure 3.5: Illustration of KLD differences on two probability distributions.

Jensen-Shannon divergence

The Jensen-Shannon divergence (JSD) is another measure to calculate the relationship between two probability distributions. However, it differs from KLD in the case where JSD is symmetric and always has a finite value. The formula of JSD is presented in Equation 3.5 below.

$$\begin{aligned} &D_{\mathrm{JS}}(P \parallel Q) = \tfrac{1}{2}D(P \parallel M) + \tfrac{1}{2}D(Q \parallel M) \\ &\text{where } M = \tfrac{1}{2}(P + Q) \end{aligned} \tag{3.5}$$

Equation 3.5 presents the formula of *JSD*. In order to calculate *JSD*, we first extracted M, which is one half of P plus Q. After we retrieved M, *JSD* is calculated by taking one half of *KLD* over $(P \parallel M)$ plus one half of *KLD* over $(Q \parallel M)$. An illustration of how *JSD* is calculated from *KLD* is presented in Fig. 3.6 below.

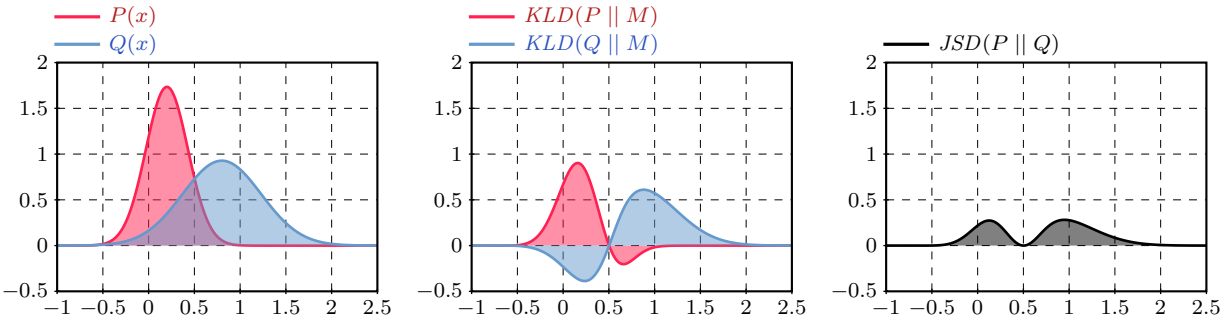

Figure 3.6: Illustration of JSD calculation.

3.1.4 Step 4: Decision Making

The results of the statistical analysis are used to decide whether the network stream is a possible Wi-Fi camera. Each result will be compared with a set threshold to determine whether the network stream is a possible Wi-Fi camera or not.

Threshold-based approach

The threshold selection was conducted based on the number of tests. Each collected result is further compared with the proposed threshold to determine the strength of the relationships. The threshold values are selected based on the corresponding F1 score. For each statistical algorithms, we computed the F1 scores for various threshold values and selected the one with the highest F1 score.

For CC, we conclude the network stream as non-Wi-Fi camera if the result is lower or equal to 0.21. The network stream is concluded as a possible hidden Wi-Fi camera if the result is higher than 0.21.

For DTW, we conclude the network stream as non-Wi-Fi camera if the DTW distance is higher or equal to 12.51, and the network stream is concluded as a possible hidden Wi-Fi camera if the DTW distance is below 12.51.

For KLD, we conclude the network stream as non-Wi-Fi camera if the KLD score is higher or equal to 0.021, and the network stream is concluded as a possible hidden Wi-Fi camera if the KLD score is below 0.021.

For JSD, we conclude the network stream as non-Wi-Fi camera if the JSD score is higher or equal to 0.005, and the network stream is concluded as a possible hidden Wi-Fi camera if the JSD score is below 0.005.

Machine-learning-based approach

This research also utilized a machine-learning-based approach to detect hidden Wi-Fi cameras. Supervised machine learning, specifically, the Artificial neuron network (ANN) model, was chosen. ANN is known for its ability to form a complex non-linear equation, which can predict new inputs based on knowledge learned from a large number of labeled datasets.

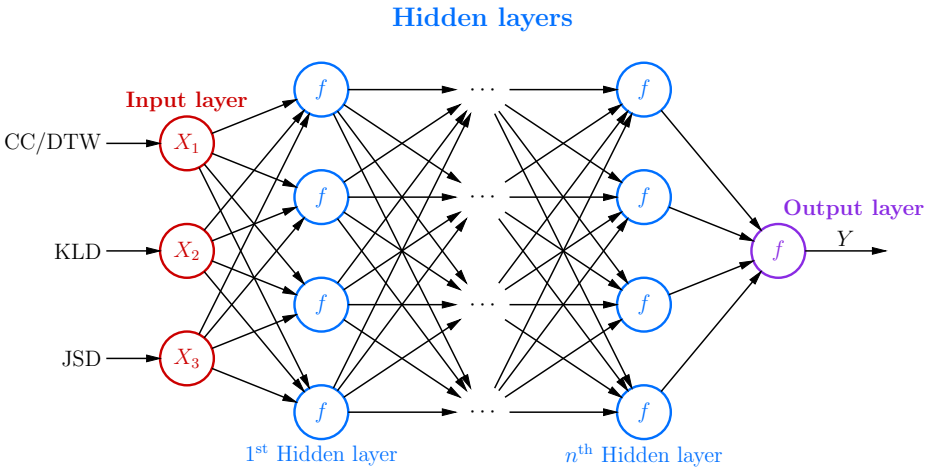

Figure 3.7: Structure of a ANN.

The overall structure of a ANN is presented in Figure 3.7. An ANN contains three layers: input layer, hidden layer, and output layer. The input layer simply takes in the raw inputs: CC, DTW, KLD, and JSD. After the raw data are passed, the information further goes to the next, hidden layer. The hidden layer can be multiple and is defined manually by the implementer. After the information reaches the first hidden layer, the neurons perform the activation function and output the results to the next layer. The progress of information passing repeats until the information reach the last layer, the output layer. The output layer again performs the activation function to output the final result. The ANN then classifies

the new inputs based on the final result. Figure 3.8 shows how information pass between each neuron.

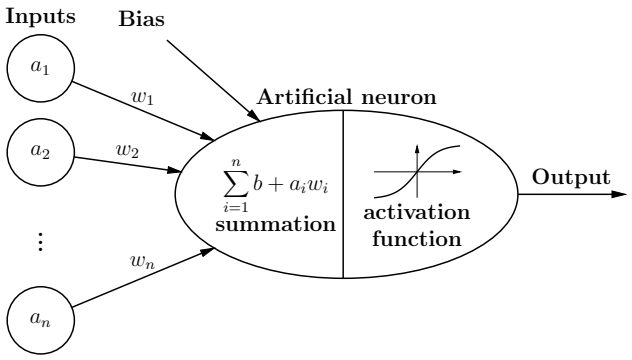

Figure 3.8: Illustration of a artificial neuron.

Before any input feeds into the next neuron, weights and bias are applied based on the solver setting. The solver setting is another manually defined function chosen by the implementer. The receiving neuron first calculates the sum of the inputs times their weights plus a bias. The neuron then passes the sum to the activation function. The activation function further calculates the output based on the given input.

3.2 Proposed detection models

Based on the proposed framework, we propose two detection models: 1) Wi-Fi-camera-based detection model and 2) mobile-phone-based detection model. The models elaborate the major steps of the framework and develop their own detecting mechanisms. The following sections describe the proposed models in detail.

3.2.1 Wi-Fi-camera-based detection model

In this detection model, a Wi-Fi camera is used to detect any hidden Wi-Fi camera within the same environment. The network traffic of the Wi-Fi camera and the environment are being recorded. The model further performs CC, KLD, and JSD to calculate strengths of their relationship. Wi-Fi camera is selected as the detector for its attribute of using the same

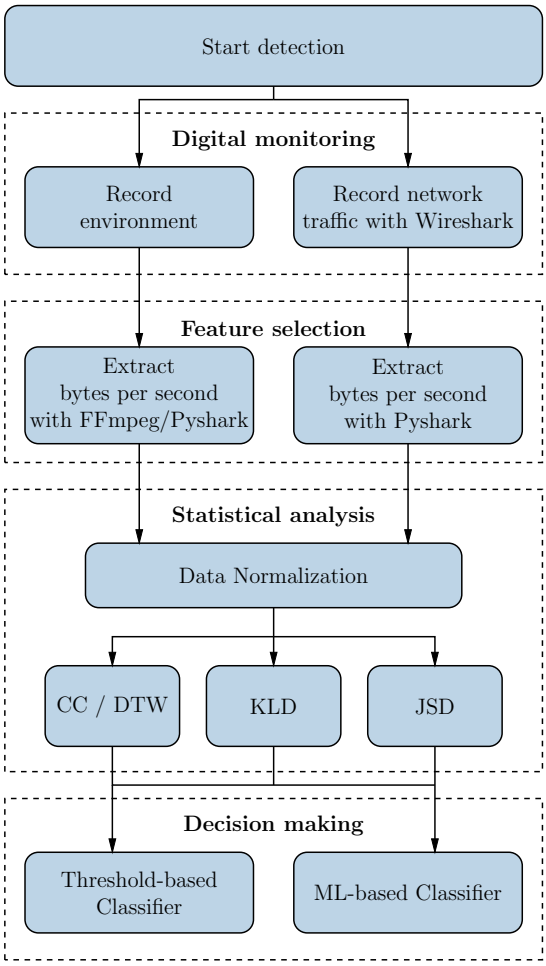

Figure 3.9: Flowchart of the two detection models.

video encoding algorithms, the H.264, to transfer the monitored recording. Figure 3.9 below is the flowchart of the proposed detection model.

Figure 3.9 presents the processes corresponding to the four major steps of the proposed framework in section 3.1. In digital monitoring, the network traffic of the detector Wi-Fi camera and the environment are recorded. These recorded PCAP files then go to the second step to extract the feature. Bytes-per-second is selected and extracted from the PCAP files by using PyShark. The generated bytes-per-second streams further proceed to perform statistical analysis. Before performing statistical analysis, both data streams first go through data normalization to shorten calculation time of the analysis. The scaled data streams then

feed into CC, KLD, and JSD to measure their relationship. The last step, decision making, compares the result with the set threshold. If the result of the CC is above the set threshold (0.21), there exists a possible Wi-Fi camera.

3.2.2 Mobile-phone-based detection model

In the second detection model, a mobile phone is used to detect any hidden Wi-Fi camera within the same environment. Recorded video from the mobile phone is being used as the detector in this model. The proposed detection model performs DTW, KLD, and JSD to measure similarity between the recorded video and the recorded network traffic. The flowchart of this proposed model is also shown in Fig. 3.9.

The flowchart separates the processes corresponding to the steps of the proposed framework in section 3.1. In digital monitoring, the network traffic is monitored while the mobile phone simultaneously records the physical environment. The recorded data are then sent to extract the feature, bytes-per-second, which is selected and extracted from the PCAP file and the MP4 files. The generated bytes-per-second streams are further used to perform statistical analysis. Before the model performs statistical analysis, both data streams first go through data normalization to shorten the calculation time. The scaled data streams then feed into DTW, KLD, and JSD to measure their similarity. The detection is based on the comparison of the resulting similarity with the set threshold. If the result of the KLD is below the set threshold (0.021), there exists a possible Wi-Fi camera.

3.3 Scenario of the experiments

For the experiments of this research, we used a 900-square-foot room with two individuals moving in the space. Movement in the environment is required, since the proposed method is based on the default behavior of H.264 and MPEG-4 to generate more data when the encoding algorithms detect motion. In the scenario of the experiments, the target hidden Wi-Fi camera is placed in a corner, from which it monitor the whole room. The proposed detectors are faced in the same general direction as the monitoring area of the hidden Wi-

Fi camera. With the detectors and the hidden Wi-Fi camera monitoring the same general direction of an area, the proposed method is able to identify the hidden Wi-Fi camera by calculating the relationships between the recorded network traffic and the recorded video.

3.4 Goal and metrics

The goal of the proposed detection models is to successfully discover hidden Wi-Fi cameras in the environment. The baseline setting of the experimental environment requires multiple movements creating fair amount of motions. In addition to successfully identifying hidden cameras in the environment, the proposed models also need to avoid classifying other non-hidden camera devices as malicious hidden Wi-Fi cameras, referred to as a false positive. The situation of identifying possible Wi-Fi cameras as other types of devices, which is referred to as a false negative, also needs to be avoided. While previous research is able to achieve a 94% accuracy rate [30], the baseline success of the thesis is defined as successfully detecting hidden Wi-Fi cameras with an accuracy rate of 94% and a false negative rate below 10%. Moderate success is defined as a 97% accuracy rate, with 10% false negative rate and 15% false positive rates, respectively. A 99% or above accuracy rate, with 3% or below false negative rate and 3% or below false positive rates, respectively, would constitute outstanding success.

3.5 Usage scenario

The proposed models are able to detect possible hidden Wi-Fi cameras within the environment. Any situation where the users do not trust the environment would be a good fit for the use of the Wi-Fi camera detection models. One example might be conference room where the attendees suspect that hidden Wi-Fi cameras are monitoring and recording their conversation. In this scenario, an individual could use their mobile phone or their personal Wi-Fi camera to detect possible hidden Wi-Fi cameras inside the private room. Another example would be hotel rooms in foreign countries. The rooms of the hotel or Airbnb are alien environments for general users, especially when it is their first time entering a foreign country. The user might want to check for the presence of a hidden Wi-Fi camera placed in-

side their room, secretly monitoring their behavior. In this situation, the proposed detection model allows general users to discover hidden Wi-Fi cameras with their mobile phone.

Chapter 4

RESULTS

Based on the proposed methodology, detecting hidden Wi-Fi camera with timing analysis, experiments are performed in controlled environments, in order to test the proposed models' usability and accountability. The baseline of the environment setup is a 900-square-foot room with two individuals moving in space. Experiments of the Wi-Fi-camera-based detection model are analyzed by calculating Correlation Coefficient (CC), Kullback-Leibler divergence (KLD), and Jenson-Shannon divergence (JSD) between the network traffic of the hidden Wi-Fi camera and the owned Wi-Fi camera, which are monitoring the same environment. Experiments of the proposed mobile-phone-based detection model are performed by calculating KLD, JSD, and Dynamic Time Warping (DTW) between the network traffic of the hidden Wi-Fi camera and the recorded video within the same environment. The following sections discuss the results of the two proposed detection models; subsections further discuss the practical applicability and evaluate the results in detail. Section 4.1 presents the data collections of the experiments. Section 4.2 discusses the results from the statistical algorithms. In section 4.3, we introduce the confusion matrix and the evaluation metrics. Section 4.4 discusses the plotted metrics of the detection models and the selected threshold-based classifiers. Section 4.5 introduces the artificial neural network (ANN) model selections and the selected machine-learning-based classifiers. Finally, section 4.6 concludes by selecting the best classifiers from threshold-based and machine-learning-based classifiers, and further discusses their advantages and drawbacks.

4.1 Data collections

This section presents the parameter settings of the experiments and the collected data for both of the detection models. In the data-collecting process, two individuals were moving in the space to generate movements in the environment. The following sections present the parameter setting and the collected data for the experiments.

4.1.1 Parameter setting

For this paper, we used an Android-based Nexus 6P and a D-Link Wi-Fi camera (DCS-936L) to perform data collection. The parameter setting of the experiments is presented in Table 4.1.

Table 4.1: Parameter settings of the experiment.

Parameters settings	**Parameters considered**
Wi-Fi camera used for data collection	D-Link Wi-Fi camera (DCS-936L) - HD 720p
Video compression algorithm of Wi-Fi camera	H.264
Mobile phone used for data collection	Google Nexus 6P
OS platform of mobile phone	Android 8.0.0
Video resolution of recorded video	720p and 1080p
Video compression algorithm of recorded video	MPEG4
Room size	900 square feet
Illumination level of the room	Bright
Testing angles	0 degree, 90 degrees and 180 degrees
Window of recording	60 seconds

As seen in Table 4.1, the testing environment of the experiments is a 900-square-foot room with illumination. The window size of the recordings (network traffic recording and video recording) is 60 seconds. Different angles between the hidden Wi-Fi camera and the detectors are also being considered. Testing angles include 0 degree, 90 degrees, and 180

degrees. The video compression algorithm of the Wi-Fi camera is H.264 with 720p resolution, and the video compression algorithm of the mobile phone is MPEG4 with 720p and 1080p as resolution, respectively.

4.1.2 Collected data

In this research, we have collected in total 724 data samples from the Wi-Fi camera, mobile phone, and other non-spying camera traffics. Table 4.2 below presents the whole dataset of this research.

Table 4.2: Data collections of the experiment.

Data collection				
Wi-Fi camera		**Recorded video**		**Other non-spying camera traffics**
No motion		No motion		Skype
	0 degree		0 degree	YouTube
Motion	90 degrees	720p	90 degrees	YouTube TV
	180 degrees		180 degrees	Amazon TV
Campus			0 degree	Switch gaming
		1080p	90 degrees	Normal browsing
			180 degrees	Video downloading
Total				
182		282		260

As Table 4.2 shows, we have collected data samples both with and without motion. We wanted to compare the performance of the proposed detection models and prove the hypothesis, i.e, confirm the ability of our model to detect hidden Wi-Fi camera in an environment with movements. We have also collected data in both 720p resolution and 1080p resolution from the recorded video, in order to compare and contrast their performance. For the Wi-Fi-camera-based detection model, a total of 182 data samples were collected. For the mobile-phone-based detection model, 282 data samples were collected in total. For

non-spying camera traffics, we collected in total 260 data samples of network traffics from Skype, YouTube, YouTube TV, Amazon TV, Switch gaming, Normal browsing, and Video downloading. The non-spying camera traffic was used in this paper to evaluate the detection model's capability in identifying different kinds network traffic patterns. The data sample of the Wi-Fi-camera-based detection model is presented in Figure 4.1 below.

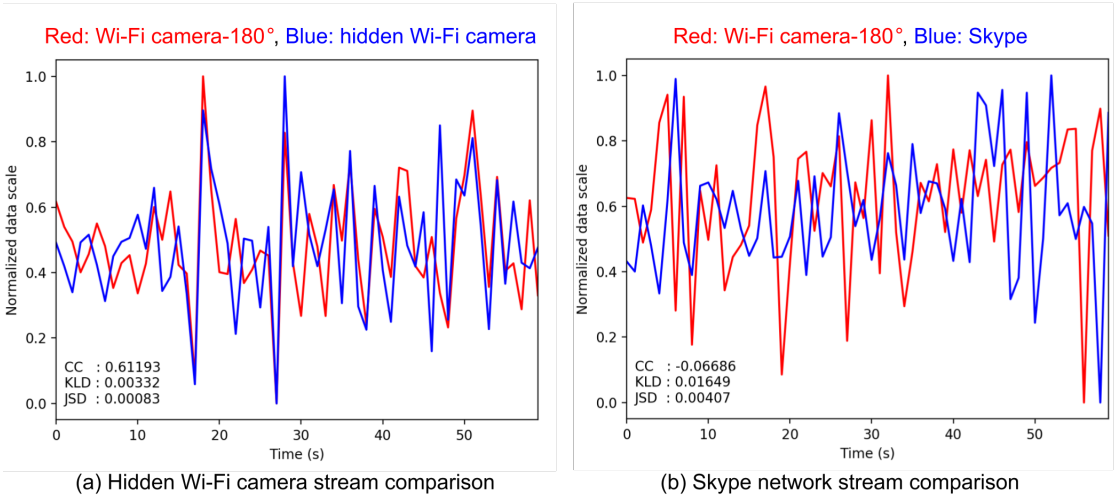

Figure 4.1: Data sample of the Wi-Fi-camera-based detection model.

Fig. 4.1(a) represents the network traffic streams of the hidden Wi-Fi camera and the owned Wi-Fi camera, respectively. Fig. 4.1(b) represents the network traffic streams of Skype and the owned Wi-Fi camera. As seen from Fig. 4.1(a), the two network streams are more closely correlated than in the Fig. 4.1(b). It also had lower KLD and JSD compared to the right figure.

Figure 4.2 presents the data sample of the mobile-phone-based detection model. Fig. 4.2(a) represents the data streams of the recorded video and the network traffic stream of the hidden Wi-Fi camera. Fig. 4.2(b) represents the data stream of the recorded video and the network traffic stream of Skype. As seen in Fig. 4.2(a), the data streams have a lower DTW compared to Fig. 4.2(b). They also had lower KLD and JSD compared to non-spying camera network stream.

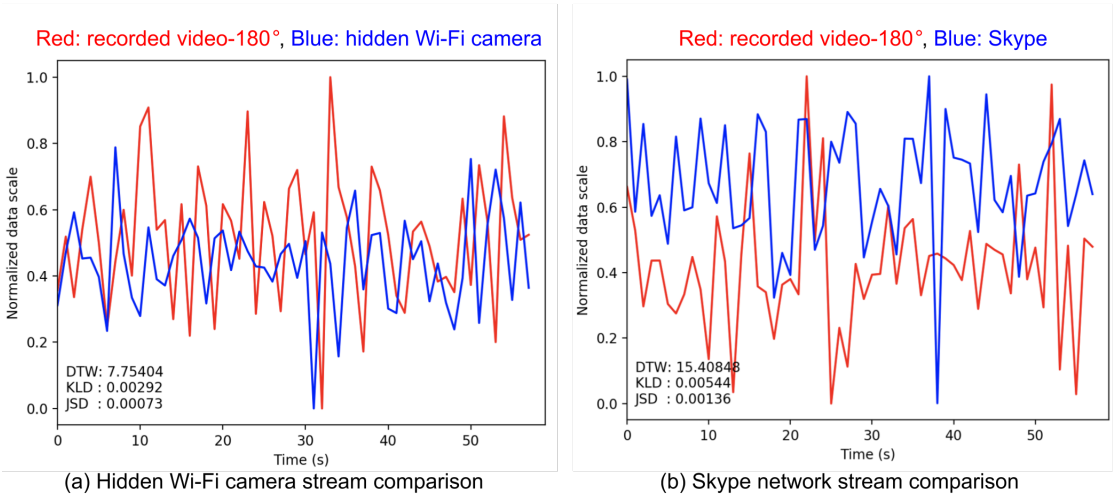

Figure 4.2: Data sample of the mobile-phone-based detection model.

4.2 Results

This section further presents the results of the experiments. Results calculated for CC, DTW, KLD, and JSD are presented and discussed in the following sections. Section 4.2.1 presents the results of the Wi-Fi-camera-based detection model, and section 4.2.2 presents the result of the mobile-phone-based detection model.

4.2.1 Wi-Fi-camera-based detection model

For the Wi-Fi-camera-based detection model, CC, KLD, and JSD are calculated to measure the relationship between the network traffic streams of the hidden Wi-Fi camera and the owned Wi-Fi camera. The results for CC is presented in Table 4.3, and the results for KLD and JSD are presented in Table 4.4.

Table 4.3 presents the results for CC between different types of network traffic and the data stream of the owned Wi-Fi camera. As we can see, CC helped us in differentiating hidden Wi-Fi camera traffic from non-spying camera traffic. The overall average of the CC ranged from 0.3 to 0.6 with a p value below 0.05. For the non-spying camera traffic, the average CC is significantly lower and even scored a negative value. The results showed

Table 4.3: Result of the CC for Wi-Fi camera-based detection model.

Testing samples		Average	Standard deviation
Wi-Fi camera			
No motion		$0.303706044 (p < 0.05)$	0.171841154
Motion	0 degree	$0.602895942 (p < 0.05)$	0.172544897
	90 degrees	$0.386569086 (p < 0.05)$	0.317507533
	180 degrees	$0.402724691 (p < 0.05)$	0.210951641
Campus		$0.304020465 (p < 0.05)$	0.272608308
Other non-spying camera traffics			
Amazon TV		−0.010107507	0.097566339
Switch gaming		0.064991552	0.087779456
Normal browsing		−0.025660436	0.152399668
Skype		0.002051681	0.143668036
Video downloading		−0.002227956	0.097922771
YouTube		−0.007038639	0.110673732
YouTube TV		0.027252663	0.088012449

that CC served as a efficient algorithm to classify hidden Wi-Fi camera stream and other non-spying camera streams.

The middle columns of Table 4.4 present the results of KLD for different types of network traffic and the data stream of the owned Wi-Fi camera. As seen from the table, the average KLD of hidden Wi-Fi camera streams ranged from 0.005 to 0.028. The average KLD of campus testing samples gave us the worst KLD among all the hidden Wi-Fi camera streams. This shows that the proposed methodology did not perform well in an open-space environment. In addition, the average KLD of other non-spying camera traffic ranged from 0.037 to 8.394. This indicates that KLD did not have the ability to significantly differentiate those network streams. Based on the results, KLD served as a weaker measure than CC in identifying the hidden Wi-Fi camera.

The right columns of Table 4.4 show the results of JSD between different types of network

Table 4.4: Result of the KLD and JSD for Wi-Fi camera-based detection model.

Testing samples		KLD		JSD	
		Average	Standard deviation	Average	Standard deviation
Wi-Fi camera					
No motion		0.011453722	0.005228987	0.002838556	0.001257969
Motion	0 degree	0.005479047	0.003783815	0.001374186	0.000975698
	90 degrees	0.009357307	0.014222304	0.002262839	0.002495481
	180 degrees	0.007548263	0.006156813	0.001852968	0.001395961
Campus		0.028721493	0.028825679	0.007344628	0.007309972
Other non-spying camera traffics					
Amazon TV		0.069038229	0.026416606	0.017119859	0.006477092
Switch gaming		0.037086219	0.015433377	0.008618703	0.003415605
Normal browsing		8.394858985	4.355179579	0.356973887	0.118067946
Skype		0.038609769	0.099741421	0.009027846	0.019909646
Video downloading		0.063745689	0.055284152	0.015102394	0.012185499
YouTube		1.471183735	1.268920535	0.291693541	0.146882616
YouTube TV		0.047435962	0.015764356	0.011856255	0.003827649

traffics and the data stream of owned Wi-Fi camera. The results showed that the average JKD of hidden Wi-Fi camera streams ranged from 0.001 to 0.007, whereas the average JSD of other non-spying camera traffics ranged from 0.008 to 0.356. Although the JSD proved to be the worst measure to differentiate hidden Wi-Fi camera streams, it had the ability to identify campus testing samples from non-spying camera traffic. In other words, JSD had the potential to detect hidden Wi-Fi camera streams in a open-space environment.

4.2.2 mobile-phone-based detection model

For the mobile-phone-based detection model, DTW, KLD, and JSD are calculated to measure the relationship between the network traffic stream of the hidden Wi-Fi camera and the data

stream of the recorded video. Results of DTW are presented in Table 4.5, and the results of KLD and JSD are presented in Table 4.6.

Table 4.5: Result of the DTW for mobile phone-based detection model.

Testing samples		Average	Standard deviation
Recorded video			
No motion		11.33004756	3.757097416
720p	0 degree	9.28481822	2.894107595
	90 degrees	11.67728415	3.127578334
	180 degrees	8.41229919	1.806480631
1080p	0 degree	8.68058130	2.176324263
	90 degrees	9.61102612	1.679525195
	180 degrees	10.84164283	6.356508534
Other non-spying camera traffics			
Amazon TV		13.23717512	4.404442176
Switch gaming		10.57320069	3.504299053
Normal browsing		22.98383699	5.867796576
Skype		12.01834661	6.626638803
Video downloading		11.08212199	3.572363009
YouTube		23.18158897	6.491112642
YouTube TV		13.56435898	4.158019029

Table 4.5 presents the results of DTW between different types of network streams and the data stream of the recorded video. As we can see, DTW did not perform well in differentiating the hidden Wi-Fi camera stream from other non-spying camera streams. The average DTW for hidden Wi-Fi camera streams ranged from 8.4 to 11.6, while the average DTW for other non-spying camera streams ranged from 10.5 to 23.18. This shows that there were multiple overlapping DTW distances between the two groups. Nevertheless, we also see that different resolution settings of the video file did not affect the resulting DTW. Both 720p and 1080p scored an average DTW from 8.4 to 11.6. We also see that the 90-degrees angle of 720p data

samples gave us the worst DTW score of all hidden Wi-Fi camera samples.

Table 4.6: Result of the KLD and JSD for mobile phone-based detection model.

Testing samples		KLD		JSD	
		Average	Standard deviation	Average	Standard deviation
Recorded video					
No motion		0.015826686	0.026754365	0.003311632	0.004660873
720p	0 degree	0.004309264	0.004465262	0.001070912	0.001106269
	90 degrees	0.689425629	1.408425095	0.027033558	0.026568581
	180 degrees	0.008186615	0.011143278	0.001913996	0.002261404
1080p	0 degree	0.006986962	0.006638193	0.001720459	0.001616679
	90 degrees	0.005086417	0.003487072	0.001255988	0.000838767
	180 degrees	0.007823499	0.014107678	0.001749409	0.001875401
Other non-spying camera traffics					
Amazon TV		0.062772516	0.024791261	0.015640886	0.006103079
Switch gaming		0.033513376	0.015027166	0.007738307	0.003305228
Normal browsing		8.264745959	4.291879708	0.354532631	0.117062109
Skype		0.032482209	0.100760789	0.007499879	0.020246439
Video downloading		0.058066744	0.055273492	0.013653066	0.012183275
YouTube		1.463401654	1.263712261	0.290810136	0.147179819
YouTube TV		0.042908844	0.015430795	0.010730559	0.003713637

The middle columns of Table 4.6 present the results of KLD between different types of network streams and the data stream of the recorded video. As wee see, the average KLD of the hidden Wi-Fi camera stream ranged from 0.005 to 0.689, while the average KLD of non-spying camera stream ranged from 0.03 to 1.46. Surprisingly, KLD is able to differentiate the no-motion testing samples from other non-spying camera streams. This demonstrates that KLD has the potential to detect hidden Wi-Fi camera even in an environment with no motion. We also see that 90-degree angle gave us the worst scoring. The 90-degree samples

can accidentally be identified as non-spying camera streams.

The right columns of Table 4.6 compare the results of JSD for different types of network streams and the data stream of the recorded video. We can see that the average JSD for the hidden Wi-Fi camera stream ranged from 0.001 to 0.026, while average JSD for other non-spying camera traffics ranged from 0.007 to 0.354. While the no-motion testing samples scored an average JSD of 0.003, JSD also served as an effective measure to identify hidden Wi-Fi camera in an environment with no motion. Exceptionally, the average JSD of the 90-degree testing samples did not increase significantly compared to DTW and KLD.

4.3 Evaluation metrics of the detection model

In this project, we used the confusion matrix to evaluate the performance of the detection models. The confusion matrix is a specific layout that aids in visualization of the algorithms. The confusion matrix categorized the outputs of an algorithm into four groups: true positive, false positive, true negative, and false negative. The confusion matrix is shown in Table 4.7.

Table 4.7: Confusion matrix.

		True condition	
Predicted condition	Total population	Hidden Wi-Fi camera	Not hidden Wi-Fi camera
	Predicted as hidden Wi-Fi camera	**True positive**	**False positive**
	Predicted as not hidden Wi-Fi camera	**False negative**	**True negative**

The rows represent the predicted conditions, and the columns represent the true conditions. If a hidden Wi-Fi camera stream is classified as a hidden Wi-Fi camera, it is stated as true positive (TP). On the other hand, if a non-hidden Wi-Fi camera stream is classified as

a hidden Wi-Fi camera, we stated it as false positive (FP). If a non-hidden Wi-Fi camera stream is classified as a non-hidden Wi-Fi camera, it is stated as true negative (TN). In contrast, if a hidden Wi-Fi camera stream is classified as a non-hidden Wi-Fi camera, we stated it as false negative (FN).

We further derived other useful evaluation metrics from the confusion matrix, which are presented in Table 4.8, which presents the key criteria extracted from the confusion matrix.

Table 4.8: Derivations from confusion matrix.

Accuracy	$\frac{TP + TN}{TP + FP + TN + FN}$
Error	$1 - \text{Accuracy}$
Precision	$\frac{TP}{TP + FP}$
Recall	$\frac{TP}{TP + FN}$
F1 score	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

Accuracy rate is calculated by dividing the sum of TP and TN by the total number of datasets. It gives us the detection rate of identifying both hidden Wi-Fi camera and non-hidden Wi-Fi camera targets.

Error rate is extracted by having one minus the accuracy rate. It represents the rate of an error classification.

Precision rate is calculated by having TP divided the sum of TP and FP. It represents the rate of identifying the hidden Wi-Fi camera over all the positive identifications; the higher the precision rate is, the lower the false positives are.

Recall rate is calculated by dividing *TP* by the sum of *TP* and *FN*. The recall rate indicates how well the model classifies all the hidden Wi-Fi camera samples. The higher the recall rate is, the lower the false negative are.

F1 score is calculated by doubling the precision rate times recall rate and then dividing by the sum of precision rate and recall rate. While the F1 score took into consideration of both the precision rate and the recall rate, it represents the overall accuracy rate of the detection model.

4.4 Results in evaluation metrics and the selected threshold-based classifiers

This section further plotted the discussed results in section 4.2 into graphs in order to visualize the performance of each statistical algorithms. Section 4.4.1 presents the results of the Wi-Fi-camera-based detection model, and section 4.4.2 presents the results of the mobile-phone-based detection model. The selected threshold-based classifiers are then introduced in section 4.4.3.

4.4.1 Wi-Fi-camera-based detection model

The plotted results of the three statistical algorithms (CC, KLD and JSD) are presented in Fig. 4.3. Red lines represent the true positive rate, blue lines represent the false positive rate, light-blue lines represent the accuracy rate, and yellow lines represent the F1 score. As we can see, the *TP* rate and *FP* rate of CC declined quickly as the threshold decreased. The CC has the steepest *FP* rate compared with KLD and JSD. While the *FP* of CC declines, the accuracy rate and F1 score increases until the *TP* rate is too low to maintain the accuracy rate. On the other hand, the results for KLD and the JSD follow a similar plotted graph. Specifically, they both maintain the *TP* rate until the *FP* rate is below 20%.

4.4.2 mobile-phone-based detection model

The plotted results of the three statistical algorithms (DTW, KLD and JSD) are presented in Fig. 4.4. We can see that both the *TP* rate and *FP* rate of DTW decline quickly as

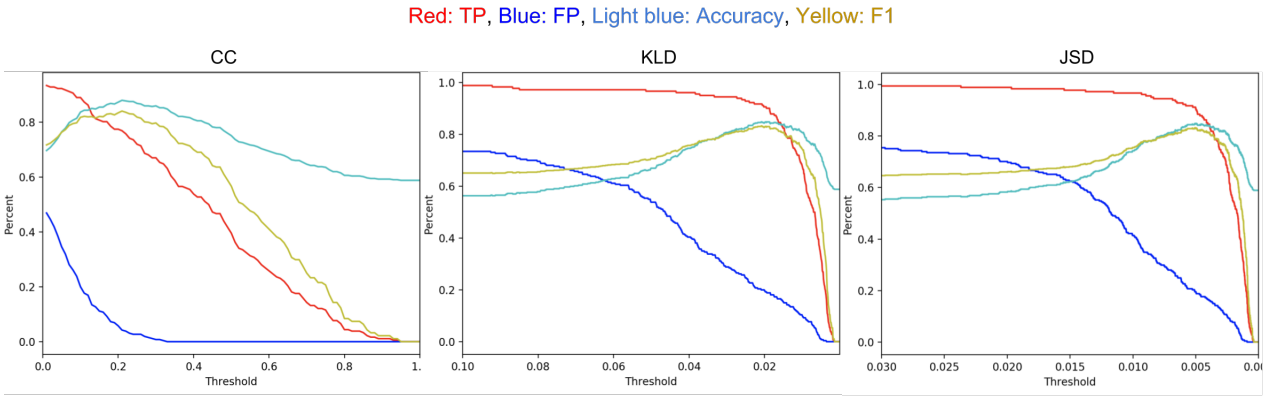

Figure 4.3: Plotted CC, KLD, and JSD for Wi-Fi-camera-based detection model.

the threshold approaches 5. The *TP* rate maintains above 90% until the *FP* rate reaches 30%. It is clear that DTW gives us the lowest accuracy rate and F1 score compared with KLD and JSD. The graph of KLD and JSD also appear nearly identical in this detection model, as they both have the F1 score maintained around 80% until the *TP* rate starts to decline. We concluded that DTW is the worst measure to identify hidden Wi-Fi cameras in this detection model.

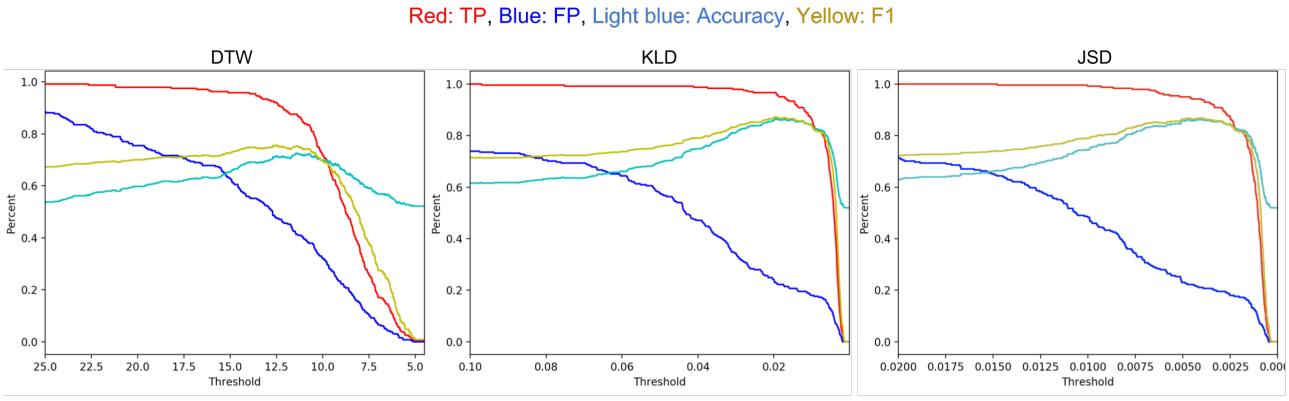

Figure 4.4: Plotted DTW, KLD, and JSD for mobile-phone-based detection model.

4.4.3 Selected threshold-based classifiers

The highest F1 scores from each of the plotted graphs are selected as the threshold-based classifiers. The threshold-based classifiers classify the new input data by comparing the calculated CC, DTW, KLD, or JSD with the set threshold. If the resulting measures are higher than the threshold, the new input data is classified as a hidden Wi-Fi camera. The threshold-based classifiers selected from the statistical algorithms are presented in Table 4.9.

Table 4.9: Threshold-based classifiers.

Metrics	F1 score	Accuracy	Error	Precision	Recall(TP)
Wi-Fi camera-based detection model					
CC	84.697	90.620	9.379	92.338	79.689
KLD	78.638	83.623	16.377	69.516	90.389
JSD	78.849	83.523	16.476	69.275	91.147
Mobile phone-based detection model					
DTW	74.139	72.157	27.843	63.462	89.216
KLD	88.613	86.431	13.568	84.157	93.435
JSD	86.206	86.427	13.572	79.096	94.790

For the Wi-Fi-camera-based detection model, CC served as the best classifier with the highest F1 score. It achieved a 90.620% accuracy rate and a 92.338% precision rate. However, it only had a 79.689% recall rate. This indicates that the CC-based classifier has a higher chance to accidentally classify hidden Wi-Fi camera traffic as a non-hidden Wi-Fi camera stream. On the other hand, KLD and JSD both performed poorly compared to CC. They only achieved 78.697% and 78.849% as F1 score, respectively. KLD and JSD also had a 16% error rate, which is 7% higher than CC.

For the mobile-phone-based detection model, KLD served as the best threshold-based classifier with 88.613% as F1 score. It also achieved a precision rate of 84.157% and a recall rate of 93.435%. This shows that KLD had a lower FP and FNrate compared with DTW and

JSD. Overall, the mobile-phone-based detection model has a better threshold-based classifier than the Wi-Fi-camera-based detection model.

4.5 Artificial neural network (ANN) model selections

In this paper, ANN is selected as our machine-learning-based classifier. In order to retrieve the best ANN model, grid search is performed with 10-fold cross validation. In the 10-fold cross validation, we first separate the dataset into 10 folds. For each fold, the ANN model is trained with the rest of the data and further performs prediction on the selected fold. After 10 runs, we evaluate the model's performance based on the average accuracy rate. This prevents over-fitting the machine learning model. Grid search performed exhaustive searching through the specific subset of hyper-parameters to retrieve the best model.

Table 4.10: Grid search on 10-fold cross validation.

ANN models				**Avg. Accuracy**	**Avg. Error**
Solver	**Hidden layers**	**Neurons per layer**	**Activation function**		
Wi-Fi camera-based detection model					
Lbfgs	3	6	Identity	91.136	8.86
Lbfgs	3	7	Relu	91.364	8.64
Lbfgs	3	5	Logistic	92.273	7.73
Lbfgs	3	13	Logistic	92.727	7.27
Mobile phone-based detection model					
Lbfgs	3	8	Identity	85.370	14.63
Lbfgs	3	6	Logistic	85.556	14.44
Lbfgs	3	7	Tanh	85.741	14.26
Lbfgs	3	12	Logistic	85.926	14.07
Lbfgs	3	13	Logistic	86.111	13.89
Lbfgs	3	14	Logistic	86.667	13.33

The result of the grid search on ANN is presented in Table 4.10. For this study, a total of

768 combinations of hyper-parameters were tested. As we can see, the best ANN model for Wi-Fi-camera-based detection model had an average accuracy rate of 92.727%. The selected ANN model had Lbfgs as the solver and Logistic as the activation function. It also had three hidden layers each containing 13 neurons. For the mobile-phone-based detection model, the best ANN gave us an average 86.667% accuracy rate. The model also had the same parameter as the best ANN model for Wi-Fi-camera-based detection model; only difference is that it had 14 neurons in each hidden layer instead of 13.

4.6 The selected best classifiers

Based on the results from section 4.4.3 and section 4.5, we further selected the best threshold-based and machine-learning-based classifiers for the two detection models. The selected best classifiers are presented in Table 4.11 below.

Table 4.11: The selected best classifiers.

Classifiers	F1 score	Accuracy	Error	Precision	Recall (TP)
Wi-Fi camera-based detection model					
Threshold-based: CC	84.697	90.620	9.380	90.338	79.689
ML-based: ANN	96.551	97.436	2.564	93.333	100.000
Mobile phone-based detection model					
Threshold-based: KLD	88.613	86.431	13.569	85.157	93.435
ML-based: ANN	94.736	95.750	4.250	90.000	100.000

Table 4.11 presents the best classifiers for the two detection models. As we can see, the ANN models outperformed threshold-based classifiers both in terms of the F1 score and accuracy rate, achieving above 94%. Moreover, both of the ANN models had an outstanding 100% recall rate. This indicates that the models eliminated all false negatives; for all the possible hidden Wi-Fi cameras, the ANN models had the ability to identify all of them without misclassifying any as a non-hidden Wi-Fi camera stream. Although the machine-learning-based classifiers surpassed threshold-based classifiers, machine learning models re-

quired longer computation time. ANN models also required large numbers of datasets to achieve higher accuracy rate. Despite their lower accuracy rate and F1 score, threshold-based classifiers can detect hidden Wi-Fi cameras in real time.

Chapter 5

CONCLUSION AND FUTURE WORK

This paper has proposed a novel method for detecting hidden Wi-Fi cameras with timing analysis. The two proposed detection models were able to successfully identify hidden Wi-Fi cameras by analyzing timing characteristics of the network traffic and the recorded video. By calculating Correlation Coefficient (CC), Dynamic Time Warping (DTW), Jensen-Shannon divergence (JSD), and Kullback-Leibler divergence (KLD) between the recorded network traffic data stream and recorded video data stream, the threshold-based classifiers can detect hidden Wi-Fi cameras with an accuracy rate of 88%.

Furthermore, this paper also analyzed the performance difference between machine-learning-based classifiers and threshold-based classifiers. This research selected the artificial neural network (ANN) as the machine-learning model. Feeding the resulting scores from CC, DTW, KLD, and JSD into an artificial neural network (ANN) model improves the accuracy rate to 97.436%. The machine-learning-based classifier also has a 5% false positive rate and a 0% false negative rate. Having a zero false negative rate indicates that the classifier is effective in detecting both hidden Wi-Fi cameras and classifying non-hidden-Wi-Fi-camera devices at the same time. The machine-learning-based classifier has met the proposed goal of moderate success, which we stated as an accuracy rate of at least 97%. Nevertheless, the machine-learning-based classifiers do require extra time to perform classification, compared to threshold-based classifiers.

Based on the experiments, however, we learned that the proposed mobile-phone-based detection model does not perform well if the mobile phone and the camera are placed at an angle of 90 degrees to one another. This negatively affects the accuracy rate of the threshold-based classifiers. Different approaches are required to overcome to limitation of 90-degree-

angle hidden Wi-Fi cameras. Based on the results, we also discovered that the proposed Wi-Fi-camera-based detection model does not perform well in an open-space environment, which was outside the scope of this research. Thus, more study in the future is required to evaluate the applicability of the methods in such an environment.

5.0.1 Future Work

In the future, we propose to extend the experiment into an IoT-rich environment to test the applicability of the proposed detection models. While the experiments were conducted in a controlled environment, we want to test the detection time of the proposed models in a pervasive environment. In order to collect more network traffic, different sniffing tools and their performance comparison are considered in the future. Since most of the Wi-Fi cameras utilized H.264 as their video compression algorithms, Wi-Fi cameras from other manufacturers will be included in the future work to compare their detection rates.

Regarding the limitation on 90-degree-angled devices, we want to propose a new method to overcome the constraint. The new method will allow the detector to be mobile while detecting the hidden Wi-Fi cameras. We hypothesize that, if the detector changes its angle periodically, it will be able to detect hidden Wi-Fi cameras in any direction. The moving detector should also be able to discover the corresponding direction of the hidden Wi-Fi camera, based on the resulting similarity scores.

We also consider extending this research to further implement a mobile application to detect hidden Wi-Fi cameras. Utilizing the detection models, a mobile application that can identify hidden Wi-Fi cameras will benefit mobile phone owners who want to detect possible hidden Wi-Fi cameras in the environment. Last but not least, sound is proposed as a possible sensing modality and is therefore included in our vision of future work. In conclusion, with the 97% accuracy rate on detecting hidden Wi-Fi cameras with a mobile phone, this research is an important step with respect to developing a mobile-based hidden-Wi-Fi-camera-detection application.

BIBLIOGRAPHY

[1] H.264:advanced video coding for generic audiovisual services.

[2] Key findings 2017 internet security threat report volume 22 | infographic | symantec.

[3] Privacy in the internet of things: Threats and challenges. 7.

[4] Noah Apthorpe, Dillon Reisman, and Nick Feamster. A smart home is no castle: Privacy vulnerabilities of encrypted IoT traffic.

[5] Selim Aras, Mehmet ztrk, and Ali Gangal. Automatic detection of the respiratory cycle from recorded, single-channel sounds from lungs. 26(1):11–22.

[6] Shattuck Bobby. F5 labs hunt for IoT vol 3.

[7] Y. H. Chou, H. C. Cheng, C. H. Cheng, K. H. Su, and C. Y. Yang. Dynamic time warping for IMU based activity detection. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 003107–003112.

[8] coffey. How to spot a hidden camera in your airbnb.

[9] Mauro Conti, Luigi V. Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. Can't you hear me knocking: Identification of user actions on android apps via traffic analysis. pages 297–304. ACM Press.

[10] Scott E. Coull and Kevin P. Dyer. Traffic analysis of encrypted messaging services: Apple iMessage and beyond. 44(5):5–11.

[11] Giorgos Dimopoulos, Pere Barlet-Ros, Constantine Dovrolis, and Ilias Leontiadis. Detecting network performance anomalies with contextual anomaly detection. pages 1–6. IEEE.

[12] W. Ding, K. Liu, H. Chen, and F. Tang. Human action recognition using similarity degree between postures and spectral learning. 12(1):110–117.

[13] S. Feghhi and D. J. Leith. Time and place: robustness of a traffic analysis attack against web traffic. In *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6.

[14] S. Feghhi and D. J. Leith. A web traffic analysis attack using only timing information. 11(8):1747–1759.

[15] K. Geers. Core illumination: Traffic analysis in cyberspace. In *2017 9th International Conference on Cyber Conflict (CyCon)*, pages 1–18.

[16] Alexia Giannoula, Alba Gutierrez-Sacristn, lex Bravo, Ferran Sanz, and Laura I. Furlong. Identifying temporal patterns in patient disease trajectories using dynamic time warping: A population-based study. 8.

[17] Xun Gong, Negar Kiyavash, and Nikita Borisov. Fingerprinting websites using remote traffic analysis. page 684. ACM Press.

[18] Xun Gong, Negar Kiyavash, Nabl Schear, and Nikita Borisov. Website detection using remote traffic analysis.

[19] Ben Herzberg, Dima Bekerman, and Igal Zeifman. Breaking down mirai: An IoT DDoS botnet analysis.

[20] Hyunsoo Kim, Changbum R. Ahn, David Engelhaupt, and SangHyun Lee. Application of dynamic time warping to the recognition of mixed equipment activities in cycle time measurement. 87:225–234.

[21] S. Kullback and R. A. Leibler. On information and sufficiency. 22(1):79–86.

[22] Brent Lagesse, Kevin Wu, Jaynie Shorb, and Zealous Zhu. Detecting spies in iot systems using cyber-physical correlation.

[23] Q. Li, H. Fan, W. Sun, J. Li, L. Chen, and Z. Liu. Fingerprints in the air: Unique identification of wireless devices using RF RSS fingerprints. 17(11):3568–3579.

[24] Nishtha Madaan, Mohd Abdul Ahad, and Sunil M. Sastry. Data integration in IoT ecosystem: Information linkage as a privacy threat. 34(1):125–133.

[25] Yair Meidan, Michael Bohadana, Asaf Shabtai, Juan David Guarnizo, Martn Ochoa, Nils Ole Tippenhauer, and Yuval Elovici. ProfilIoT: a machine learning approach for IoT device identification based on network traffic analysis. pages 506–509. ACM Press.

[26] Andrew W. Moore and Denis Zuev. Internet traffic classification using bayesian analysis techniques. 33(1):50.

[27] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of tor. In *2005 IEEE Symposium on Security and Privacy (S P'05)*, pages 183–195.

[28] Thomas Pasquier, Jatinder Singh, Julia Powles, David Eyers, Margo Seltzer, and Jean Bacon. Data provenance to audit compliance with privacy policy in the internet of things. 22(2):333–344.

[29] Karl Pearson. VII. mathematical contributions to the theory of evolution.III. regression, heredity, and panmixia. 187:253–318.

[30] Hiroaki Sakoe. Dynamic programming algorithm optimization for spoken word recognition. 26:43–49.

[31] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. 11(5):561–580.

[32] K. Shiba, T. Kaburagi, and Y. Kurihara. Monitoring system to detect fall/non-fall event utilizing frequency feature from a microwave doppler sensor: validation of relationship between the number of template datasets and classification performance. 23(1):152–159.

[33] Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: attacks and defenses. pages 18–33. Springer-Verlag.

[34] Sandra Siby, Rajib Ranjan Maiti, and Nils Tippenhauer. IoTScanner: Detecting and classifying privacy threats in IoT neighborhoods.

[35] Gurdit Singh, Divya Bansal, and Sanjeev Sofat. A smartphone based technique to monitor driving behavior using DTW and crowdsensing. 40:56–70.

[36] Statista. IoT: number of connected devices worldwide 2012-2025.

[37] Machiko Toyoda, Yasushi Sakurai, and Yoshiharu Ishikawa. Pattern discovery in data streams under the time warping distance. 22(3):295–318.

[38] L. Xin and W. Neng. Design improvement for tor against low-cost traffic attack and low-resource routing attack. In *2009 WRI International Conference on Communications and Mobile Computing*, volume 3, pages 549–554.