NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

SIDE CHANNEL ATTACKS ON ENCRYPTED VIDEO CONFERENCING TRAFFIC

by

Glenn C. Daves

June 2021

Thesis Advisor: Vinnie Monaco
Second Reader: Theodore D. Huffmire

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
SIDE CHANNEL ATTACKS ON ENCRYPTED VIDEO CONFERENCING TRAFFIC

5. FUNDING NUMBERS

6. AUTHOR(S) Glenn C. Daves

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

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

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Popular video conferencing platforms, such as Skype, Zoom, and Facebook Messenger, use Variable Bit Rate (VBR) codecs to perform audio compression, which allows the system to dynamically change the audio encoding rate based upon the complexity of a sound. While this significantly reduces network bandwidth usage, past work has shown that using VBR codecs results in the length of packets, even when encrypted, being highly correlated to the phonemes (basic building blocks of speech) they encode. Previous papers have demonstrated how this vulnerability, which also exists in Voice over Internet Protocol (VoIP) technology, allows an adversary to determine the language in use and the existence of specific phrases within a conversation. From this data, an adversary can even produce a transcript of the entire call. In this paper, we explore the use of this exploit on video conferencing traffic in four domains: identification, authentication, profiling, and correlation to audio amplitudes. Significant limiting factors, including an extremely small dataset, preclude achieving statistically significant performance in the first three areas; we do discover and present a high correlation between lengths of encrypted packets and the amplitudes of the audio they are transmitting.

14. SUBJECT TERMS VoIP, encrypted, network, video conferencing, video chat, Zoom, Skype, packet, neural-network, Facebook Messenger, authentication, identification, profiling, amplitude, TIMIT, phoneme, phonics, variable bit rate codec	**15. NUMBER OF PAGES** 77
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

SIDE CHANNEL ATTACKS ON ENCRYPTED VIDEO CONFERENCING TRAFFIC

Glenn C. Daves
Ensign, United States Navy
BS, United States Naval Academy, 2020

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2021

Approved by: Vinnie Monaco
Advisor

Theodore D. Huffmire
Second Reader

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Popular video conferencing platforms, such as Skype, Zoom, and Facebook Messenger, use Variable Bit Rate (VBR) codecs to perform audio compression, which allows the system to dynamically change the audio encoding rate based upon the complexity of a sound. While this significantly reduces network bandwidth usage, past work has shown that using VBR codecs results in the length of packets, even when encrypted, being highly correlated to the phonemes (basic building blocks of speech) they encode. Previous papers have demonstrated how this vulnerability, which also exists in Voice over Internet Protocol (VoIP) technology, allows an adversary to determine the language in use and the existence of specific phrases within a conversation. From this data, an adversary can even produce a transcript of the entire call. In this paper, we explore the use of this exploit on video conferencing traffic in four domains: identification, authentication, profiling, and correlation to audio amplitudes. Significant limiting factors, including an extremely small dataset, preclude achieving statistically significant performance in the first three areas; we do discover and present a high correlation between lengths of encrypted packets and the amplitudes of the audio they are transmitting.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

CELP Code-Excited Linear Prediction

DF Don't Fragment

DOD Department of Defense

EER Equal Error Rate

END Ensemble of Nested Dichotomies

FB Facebook

FP False Positive

GMM Gaussian Mixture Model

HMM Hidden Markov Model

IPA International Phonetic Alphabet

KNN K-Nearest Neighbor Classifier

LSTM Long Short-Term Memory

MTU Maximum Transmission Unit

NN Neural Network

NPS Naval Postgraduate School

paprefs PulseAudio Preferences

pavucontrol PulseAudio Volume Control

pcap Packet Capture

RNN Recurrent Neural Network

ROC	Receiver Operating Characteristic
RTP	Real-time Transport Protocol
SRTP	Secure Real-time Transport Protocol
TCP	Transmission Control Protocol
TIMIT	DARPA - Texas Instruments/Massachusetts Institute of Technology Acoustic-Phonetic Continuous Speech Corpus
TP	True Positive
UDP	User Datagram Protocol
USN	U.S. Navy
VoIP	Voice over Internet Protocol
VBR	Variable Bit Rate

Acknowledgments

There are many people I would like to thank that have contributed to this project in various ways. First, to Admiral Frank L. “Skip” Bowman and the C.J. Mack Family Foundation, who established the Bowman Scholar program at the United States Naval Academy, which allowed me the opportunity to pursue my master’s degree at Naval Postgraduate School (NPS) immediately after graduation. I must also recognize Dr. Christopher W. Brown from the United States Naval Academy, who guided me in becoming a Bowman Scholar and greatly encouraged my interest in higher education and computer science research.

Here at NPS, my thesis advisor Dr. Vinnie Monaco has been an amazing source of information and guidance, encouraging and challenging me to consistently go further and deeper into this project. I would not have made it through this process without his wisdom and encouragement.

Finally, I must thank my family for their love and support, first among them my amazing wife, Emily, who has endured the long nights, frustrations, and challenges with incredible grace and love. She is a never-ending source of encouragement and support and a wonderfully accomplished woman in her own regard. I love you, Emily.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Voice over Internet Protocol (VoIP) is a technology that utilizes the Internet as a medium for telephonic voice communications. When compared to traditional phone providers and mediums of transmission, VoIP offers several advantages. VoIP is cheaper, more portable, and easier to scale than a wired phone service, which has contributed to its widespread adoption across sectors [1]. To transition from spoken voice to bits travelling across the Internet, VoIP technology utilizes audio codecs, which capture speech at its most basic building block, a phoneme, and map them to a set of pre-encoded sounds. Different from generic audio codecs, speech codecs utilized by popular VoIP technologies take advantage of the limited range of sounds the human voice can produce and the periods of silence that naturally occur in speech (pausing for effect, not talking while someone else is, etc.) to efficiently utilize network bandwidth. This manifests itself in Variable Bit Rate (VBR) mode, a very popular option that allows the codec to set its encoding rate based on the complexity of the sound to encode, and shift this rate dynamically during a conversation [2].

The drawback to the bandwidth saving use of VBR mode is that, by definition, the encoding rate changes based on sounds, and therefore each basic unit of speech will be encoded at a different rate, resulting in differently sized payloads (assuming a constant time interval per packet). In practice, this results in a strong correlation between the length of a packet and the phoneme(s) it is encoding, which creates a serious leak of information. However, as one would expect, VoIP packets are encrypted before being sent across the Internet to protect the privacy and integrity of the call. The encryption protocol that VoIP uses, Secure Real-time Transport Protocol (SRTP) does not pad the length of the packets, so while the data is encrypted, the original length remains, and therefore so does the relationship between packet length and phonemes. Several papers have demonstrated how this leak of information can be used to identify the language spoken over a call, identify the existence of specific phrases within a call, and even develop a transcript of the entire conversation [2], [3].

Video conferencing technology, widely used today through platforms such as Skype, Zoom, and Facebook (FB) Messenger, were developed after the emergence and success of VoIP

technology, and importantly built on existing VoIP technology. The speech codecs used in video conferencing software also utilize VBR mode, which means the fundamental information leak produced by the correlation between audio packet lengths and the phonemes they encode still remains. In this project, we examine we examine how this exploit might be leveraged in the video conferencing domain rather than VoIP.

Specifically, we attempt to perform identification, authentication, and profiling using encrypted video conferencing traffic, and examine the correlation between packet lengths and audio amplitude.

1.1 Contributions

While building upon existing work, this project also makes several key, new contributions, listed below.

1. Apply the phoneme/packet length exploit to video conferencing software instead of VoIP platforms.
2. Examine whether this attack is viable given variable packet time sizing when using SILK.
3. Attempt to perform gender and dialect profiling from encrypted video conferencing network traffic.
4. Examine the correlation between packet size and audio amplitude.

1.2 Motivation

The Covid-19 pandemic brought on sudden changes to every aspect of our lives, and long periods of stay at home orders and social distancing guidelines pushed many businesses to operate remotely. Accordingly, many organizations, including the Department of Defense (DOD), turned to video conferencing platforms like Zoom to hold meetings and coordinate work. Zoom alone saw its use rise from roughly 10 million participants a day in December of 2019 to over 300 million users per day in April of 2020. Similarly, its yearly revenue increased from $330.5 million in FY2019 to $622.7 million for FY2020 [4].

The rapid expansion in video conferencing use raised questions about how secure the technology actually was. Previous work nearly ten years prior had identified serious vulner-

abilities in VoIP audio codecs operating in VBR mode, and we wondered whether similar vulnerabilities existed in video chat platforms such as Zoom or Skype. A short amount of research revealed that no work had been done to address the previously identified VoIP information leaks, and, most importantly, all of the major video chat providers (Skype, Zoom, FB Messenger, etc.) were using VBR codecs for audio. This made the security of encrypted video chat traffic a timely and important issue to investigate.

The implications of such an exploit, considering the sudden and massive rise in VoIP and video conferencing usage, are not too hard to imagine. From an adversarial perspective, knowing who is meeting with whom can reveal a lot of sensitive information. For example, if a traffic capture reveals that I am meeting with an oncologist, an attacker can deduce, with some degree of certainty, that I may have cancer. Conversely, this exploit could also be leveraged defensively. A network administrator could constantly monitor VoIP and video conferencing traffic, checking for unknown or unauthorized users. In practice, this could help prevent users who have been banned from certain platforms for violating terms of use from simply creating a new account and continuing their antics.

Now, with multiple vaccines in production and distribution, there seems to be an end in sight to the pandemic, but perhaps not work from home. Many workers enjoy the flexibility of remote work, and significant numbers plan to continue working remotely, full or part time [5]. If this work from home paradigm will become the new normal, it is imperative that businesses and governments alike truly understand the security implications of conducting their meetings over the Internet via video conferencing software.

1.3 Thesis Organization

The rest of this paper is organized in the following manner. Chapter 2 explores related works and establishes a baseline understanding of VoIP and video conferencing technology which is necessary to understand the rest of the paper. Chapter 3 details our process of data collection and training paradigms used to accomplish each goal. Chapter 4 provides insight into what the encrypted traffic data looks like. Chapter 5 presents experimental results for each area of interest. Finally, Chapter 6 offers analysis, mitigation, and limitations of this approach to exploiting video conferencing traffic.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background

2.1 Voice Over IP

VoIP technology utilizes the Internet as a medium for telephonic voice communications. When compared to traditional phone providers and mediums of transmission, VoIP offers several advantages. First is cost; IP telephony is less expensive than a traditional, wired phone service. Second, VoIP is portable; the service is available anywhere that has access to the Internet. Third is scale; VoIP allows telephony providers to piggyback off of already existing Internet connections rather than taking the time and resources to establish traditional wired connections [1]. For these reasons VoIP has exploded in popularity and deployment over the past two decades, becoming integral to telecommunication systems throughout the world.

The process of encoding speech into packets and transmitting them across the Internet is the foundation of VoIP technology. The following sections will break apart this complex process into more manageable pieces, highlighting the steps most important to understanding this project's motivation.

2.1.1 Phonemes

Speech can be broken down into a series of smaller pieces; some concepts, like *vowels* and *consonants*, will be familiar to most readers, even without a phonetics background. Others, like *phone* and *phoneme*, are less familiar. Regardless, this section will briefly break down speech to its smallest elements and build back up to arrive at a workable definition of the key term *phoneme*.

A *phone* is a specific speech sound, usually represented by a single letter in the English language. However, drawing too close a connection between letters and sounds can be misleading, as often the same letter is pronounced in vastly different ways based upon the other letters around it. Consider the sound of the letter *c* in the words *car* and *cell*. For this reason, the International Phonetic Alphabet (IPA) was created to accurately represent

phones.[1] Phones are organized into two main classes: *consonants* and *vowels*. Consonants are generally produced in human speech by restricting or blocking airflow in some way. Vowels, on the other hand, are characterized by less obstruction and being louder and longer lasting than consonants [6].

A *phoneme* is an abstraction that represents similar phones with one symbol. For example, the letter f in the words *fat*, *often*, and *cliff* are all represented by different phones but share a phoneme [3]. From this perspective, phonemes help simplify language classification and are more intuitive to the lay reader, to whom all the *f's* sound very similar. This more simplistic view of speech is a key reason why many of the most popular speech codecs are phoneme based.

2.1.2 Speech Codecs

Audio for VoIP is generally encoded with speech specific audio codecs, such as the SILK codec used by Skype and Zoom. Speech codes are more efficient in terms of network usage than generic audio codecs because they advantage of 1) the periodic rests inherent in most speech patterns and 2) the limited range of sounds that can be produced by a human mouth.

Most modern speech codecs are based on a process known as Code-Excited Linear Prediction (CELP). For each sound, a CELP encoder searches through a codebook to find the sound that most closely matches [2]. Most modern codecs, including Skype's SILK, encode sound in VBR mode. This means the encoder chooses the bit rate for each packet in order to reduce bandwidth usage. Because some sounds are more complex than others, and often regular speech patterns include long pauses (e.g. while someone else is speaking), VBR mode allows substantial savings in packet length, and therefore network usage. In practice, phonemes known as *fricatives* (*"f"* or *"s"* sounds) will be encoded at lower bit rates than vowels [7].

VoIP packets are encrypted and transmitted using SRTP. Importantly, SRTP does not alter the size of the payload with padding, which means the original packet lengths are transmitted across the Internet [3]. Because packet payloads differ based on the phoneme(s) they are transmitting, the packet length of this encrypted traffic leaks a significant amount

[1]The IPA is not the only phonetic alphabet, but is the most widely used standard in phonetics. ARPAbet is another common phonetic alphabet.

of information — since phonemes are the basic units of speech, if an eavesdropper knows what phonemes are being used, they can work backwards and reconstruct the actual speech.

2.2 Video Conferencing

Just as the telephone was developed decades before the Internet, video conferencing (or video chatting) was similarly envisioned in a pre-Internet world. AT&T introduced the Picturephone in 1964 at the World's Fair in New York City, but it, and many of its successors, were incredible failures. The high cost of these first machines and the aversion of many people to being seen while on a call prohibited mass adoption of the technology [8]. The rise of the Internet provided a more cost-effective medium for video conferencing to take place, and as consumers got over their fear of being on camera, video chat technology was widely and rapidly adopted in the early 21st century.

VoIP technology laid the groundwork for video conferencing platforms, like Skype, to come onto the scene. Fundamentally, video conferencing on the Internet is VoIP with the addition of real-time video. To accomplish this, video conferencing also utilizes the same basic paradigm as VoIP:

1. Use a video codec (H264, Xvid, MPEG-1, etc.) to compress video into packets.
2. Establish a User Datagram Protocol (UDP) connection.
3. Transport packets using Real-time Transport Protocol (RTP).

The audio and video is therefore recorded and transmitted concurrently but separately. It is then assembled and presented to user by the client machine. The extent to which we must understand modern video conferencing software is this: it utilizes VoIP technology for audio, and adds an additional layer of video on top. Therefore, the same relationship between packet lengths and phonemes, discussed earlier, should exist in video conferencing traffic once the audio packets have been separated from the video packets.

2.2.1 Video Conferencing Speech Codecs

This paper will utilize Skype, Zoom, and FB Messenger video conferencing platforms to collect data. A brief survey of their underlying speech codecs is provided below.

SILK

The SILK audio codec is used by both Skype and Zoom to encode and transmit audio during video calls. It is designed and maintained by Skype. SILK is a VBR codec that can set its encoding rate between 6 and 40 kbps [9]. Interestingly, SILK will vary the number of frames per packet based on network conditions: each packet could contain 20, 40, 60, 80, or 100 ms worth of encoded audio [3]. The effect of this on the phoneme/packet length exploit has not been explored.

Opus

The Opus audio codec is used by Zoom and FB Messenger. It is now the default codec for real-time audio communications because WebRTC (an HTML5 specification for real-time media communications) has required its use [10]. Opus is a VBR codec with ranges from 6 to 510 kbps.

2.3 Related Works

2.3.1 Spot me if you can

In the 2008 paper *Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations*, authors Wright, Ballard, Coull, Monrose, and Masson propose identifying spoken phrases in encrypted VoIP traffic by leveraging the relationship between phonemes and packet length when using a VBR codec since SRTP does not pad the packet lengths [7].

To accomplish this, the authors use the DARPA - Texas Instruments/Massachusetts Institute of Technology Acoustic-Phonetic Continuous Speech Corpus (TIMIT), a database that contains 10 phonetically rich sentences spoken by 630 English speaking individuals. These speakers are distributed between male and female and 8 dialects, for a total of 6300 spoken sentences [11]. After encoding the speech in TIMIT with the Speex codec in VBR mode, the authors trained a Hidden Markov Model (HMM) to search for phrases held within a conversation. Their results show 50% accuracy overall, with over 90% accuracy for certain, more common phrases.

2.3.2 Speaker recognition from encrypted VoIP communications

The 2010 paper *Speaker recognition from encrypted VoIP communications* applies the observed packet-length/phoneme correlation to the realms of speaker identification and verification [12]. Given a set of speakers with previously recorded speech segments, identification is defined as identifying the suspect to whom a disputed speech segment can be attributed. Verification, on the other hand, is the probability that two speech segments, one known and one unknown, are from the same person.

After packet length extraction from VoIP traffic, the authors apply various approaches to each domain. For speaker recognition, the authors explore using HMM, Gaussian Mixture Model (GMM), and Ensemble of Nested Dichotomies (END) to model the speaker's identity based on packet length sequences. For verification, the authors approach it as a two-class classification problem and a regression problem.

Final results yielded identification accuracy of 70-75% for a group of 10 speakers, and Equal Error Rate (EER) of 17% in the case of verification.

2.3.3 Hookt on fon-iks

The 2011 paper *Phonotactic reconstruction of encrypted VoIP conversations: Hookt on fon-iks*, by authors White, Matthews, Snow, and Monrose, builds on the success and motivations of *Spot me if you can*. They attempt to construct a hypothetical transcript of the entire encrypted VoIP call [3]. This work also seeks to exploit the relationship between SRTP packet lengths and phonemes.

The authors break this complex objective into smaller segments, and build a complete pipeline to go from encrypted VoIP packets to words of a transcript. The first step is to extract packet lengths from encrypted VoIP traffic. The authors use TIMIT sentences to create the conversations transmitted with VoIP. The second step is to segment the sequence of packet lengths into phonemes using maximum entropy modeling. The third is to classify the segments into concrete phonemes with a combination of maximum entropy modeling and profile HMM. The fourth step is to turn the phonemes into word segments using a phonetic constraint model. Finally, the authors use a phonetic edit distance metric to turn the word segments (comprised of phonemes) into English words that together form the transcript of a conversation.

Through this pipeline, which relies heavily on the fields of phonetics and phonotactics, the authors were able to successfully transcribe several segments of an encrypted VoIP conversation. This paper represents the height of what has so far been achieved with regards to the packet length/phoneme exploit demonstrated in Section 2.1.2.

2.3.4 TypeNet

In the 2020 paper *TypeNet: Scaling up keystroke biometrics*, authors Acien, Morales, Vera-Rodriguez, Fierrez, and Monaco investigate using keystroke dynamics to authenticate users typing free-text [13]. To accomplish this, the team built a Recurrent Neural Network (RNN) with two Long Short-Term Memory (LSTM) layers of 128 units each. The authors then used Siamese training, with two inputs (keystroke sequences) and two outputs (embedding vectors). During training, the model learned information about the keystroke sequences so that the distance between embedding vectors would be small for keystrokes from the same user and large for keystrokes of different users.

With this framework, the team achieved an EER from 9.53% to 3.33%, depending on the amount of user data employed. We see similarities between the keystroke sequences used in this paper and the sequences of packet lengths produced by VoIP traffic. For this reason, we believe that this model of training for authentication can be applied to our domain of interest.

2.3.5 Side-Channel Leaks in Web Applications

The 2010 paper *Side-channel leaks in web applications: A reality today, a challenge tomorrow* details so called "side channel leaks" from software-as-a-service applications [14]. Web applications are split into two parts: the browser/client side and server side. As information traverses from client to server, there is a potential for leaks of information to be revealed through various means: packet sizes, sequence of states, etc. Therefore, despite the encryption that has become commonplace throughout the Internet (via HTTPS or other standards), valuable information is leaked constantly.

The authors examine the extent of this danger, and find that an eavesdropper can deduce the medical status of a user based on her interactions with an online health site, a user's salary

and investment choices based on his use of an investment website, and many similar leaks of personal information.

Importantly, the authors note that the exploits they discovered are not specific to the targeted applications; rather, the root causes go to the very structure of web applications today: stateful communication, low entropy input, and significant traffic distinctions, all of which are characteristics of modern video conferencing applications as well.

2.3.6 Skype & Type

The 2019 paper *Skype & Type: Keyboard eavesdropping in voice-over-IP* explores keyboard acoustic eavesdropping over VoIP, primarily with the Skype platform [15]. Keyboard acoustic eavesdropping involves mapping the sound of typing on a keyboard to text, and this paper applies it to a new domain by extracting erroneous keyboard sounds from a VoIP call. By assuming the realistic scenario in which a target is multi-tasking by typing on his computer while also on a VoIP call, the authors investigate the extent to which an adversary can perform acoustic keyboard eavesdropping in this environment, in particular by discovering sensitive information the target is typing, such as personal details or a password.

To accomplish this attack, the authors break the problem into four phases: Data Collection, Feature Extraction, Model Training, and Attack. Feature Extraction involves isolating keyboard sounds from the rest of the audio, segmenting them into single keystrokes, and then mapping the audio to a single feature per stroke. Then, for Model Training, the authors use popular machine learning algorithms (Logistic Regression, Linear Discriminant Analysis, Support Vector Machine, Random Forest, and K-Nearest Neighbor) to perform multi-class classification, where each character on the keyboard is a class. From this classification they can string together characters to create words and potential passwords.

The authors demonstrate significant success with this exploit, 91.7% accuracy in guessing a random key when the attacker has some knowledge on the target's past typing style and keyboard model.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Methodology

3.1 Overview

Our methodology for this project consisted of two parts. First is data collection. We set up video calls between two machines on Skype, Zoom, and FB Messenger and used the audio files from TIMIT to simulate a conversation; we then captured the encrypted traffic as it travelled between the two hosts. The second step was to isolate the packet lengths for the audio traffic and put them into chronological sequences for each sentence, associating each sequence with its matching sentence label from TIMIT, which gives information on dialect, gender, and a unique ID number to differentiate speakers in those same subcategories. We investigated four uses of this captured data:

1. Identification — Can we match a sequence to a previously recorded individual?
2. Authentication — Do two sequences belong to the same person?
3. Profiling — Can we identify other features, specifically gender and dialect, from the sequence patterns?
4. Amplitude Correlation — Do the packets lengths correlate to the amplitude of the original audio?

Each problem requires a different approach in terms of machine learning design and training, as well as testing and evaluation methods. These will be expanded upon in Section 3.3.

3.2 Data Collection

3.2.1 Enabling Simultaneous Output

To come as close as possible to replicating live human voices on the video call and limit background noise, it is necessary to enable simultaneous output so audio will play from the regular speaker and be directed into the microphone as well. This provides better sound quality (closer to live speech) than playing the sound from one speaker while holding a microphone nearby. All data collection was performed using an Ubuntu 20.04 system, and

the necessary steps to enable simultaneous output (provided below) are specific to that operating system.

To enable this feature, two dependencies are needed: PulseAudio Preferences (paprefs) and PulseAudio Volume Control (pavucontrol). These can be installed with the command:

```
$ sudo apt-get install paprefs pavucontrol
```

Skype and FB Messenger

To set this feature for Skype and FB Messenger, we started by initiating a call. Next, we launched the paprefs program, navigated to the Simultaneous Output tab and checked the box. Finally, we launched pavucontrol, navigated to the Recording Tab, and changed the device to "Monitor..." [16]. This allowed audio to simultaneously play out of the speakers and directly into the microphone, to be treated as input audio for the speech codecs.

Zoom

Zoom does not allow for simultaneous output in the Ubuntu 20.04 environment. Instead of following the steps for Skype and FB, we had to create virtual audio sources and sinks with the following commands:

```
$ pactl load-module module-null-sink sink_name=
    zoom_input sink_properties=device.description=
    zoom_input
$ pactl load-module module-remap-source master=
    zoom_input.monitor source_name=zoom_mic
    source_properties=device.description="zoom_mic"
```

Next, we initiated a Zoom call and launched pavucontrol. In the Output tab of pavucontrol, we selected “zoom_input” for the application playing audio. Finally, in Zoom we selected “zoom_mic” as the microphone. After this, the audio was treated as input to the microphone and only played over the Zoom call [17].

3.2.2 Video Call Setup

To collect our data for this project, we set up a video call between two machines, Host A and Host B, for Skype, Zoom, and FB Messenger. Host A, running Ubuntu 20.04, was the worker for this experiment, while host B, running Windows 10, served as the other end of the call, and did not play an active role in the data collection other than to maintain the call for the duration of the collection phase. Once the call was setup between hosts A and B, we enabled simultaneous output using the methods described in the previous section. Then we ran a Bash script that accomplished three tasks:

1. Launch a Python program to play each of the 6300 TIMIT sentences, one at a time. Because simultaneous output is enabled, this will be treated by Host A as microphone input audio to the video call.
2. Record the timestamp before and after each sentence is played and the corresponding TIMIT label (Accomplished in the same Python program as Task 1).
3. Capture all outgoing traffic from Host A using *tcpdump*.

After running this script, we were left with a Packet Capture (pcap) of the video chat traffic, and a file containing the start and stop times for each of the 6300 TIMIT sentences.

3.2.3 Parsing the Packet Capture

The first step to parsing the resulting pcap was to remove any packets that fell outside the range of the first sentence being played and the last sentence finishing.[2]

We then created 6300 arrays, one for each sentence in the TIMIT database. Next, we used the Python library *dpkt* to parse through the packets in the pcap file and extract the timestamp and length of the Transport Layer data field for every packet that matched the following criteria: IPv4, destination IP address equal to that of Host B, and UDP protocol [18].[3]

We identified the sentence the packet belonged to by searching through the list of (start_time,

[2]Because the call had to be manually stopped after six hours of playing, *tcpdump* captured some packets that did not fall within the appropriate time zone. We used Wireshark and two time filters (start and stop) to filter out any unnecessary packets, but this can also be accomplished with command line tools such as *editcap*.

[3]We only captured outgoing packets with *tcpdump* so there was no reason to check for source IP address.

stop_time, label) 3-tuples, and finding the correct tuple such that:

$$start\_time < timestamp < stop\_time \tag{3.1}$$

After finding the appropriate label, we placed the (timestamp, length) tuple into its corresponding array, and then sorted chronologically. This left us with a sequence of packet lengths and timestamps for each of the 6300 sentences in the TIMIT repository. Chapter 4 offers an in-depth analysis of the data we extracted from the pcap files.

3.3 Training and Testing

3.3.1 Feature Extraction

This section will overview the general process of feature extraction used in Identification, Authentication, and Profiling.

In their original state, the timestamps in the pcap files are strictly increasing, even among different samples, due to the fact that we collected the data by playing each sample sequentially over time. This results in the last packet of Sentence 1 having a timestamp roughly six hours earlier than the first packet of Sentence 6300, which is not ideal. To move away from these non-stationary timestamps, we calculated the difference between each timestamp and the previous, to produce bounded values that can be compared across samples. We performed the same operation with the packet lengths as an additional feature that might be useful. Because each sequence has a different length, we padded all of the shorter sequences with a value of -1, to prevent any data loss. This left us with a feature space of size (6300 x *max_sample_length* x 2).

We created labels based on the specific goal of the training. For identification and authentication, the labels identify a unique individual, resulting in 630 labels. For gender profiling, there are two labels, male and female, and for dialect training there are 8 labels corresponding to the eight dialects present in the TIMIT dataset. We then partitioned the features and labels into training and testing subsets: X, Y, X', and Y', where X and X' designate features, and Y and Y' designate labels.

Because the feature space was so small, we built a Neural Network (NN) to extract a larger

feature space for each sample. This feature extraction NN will be referred to as *NN1* for the rest of this chapter. NN1's architecture is described in Table 3.1.

Table 3.1. NN1 Architecture

Layer Number	Layer Type	# Nodes	Activation Function
1	Masking	n.a.	n.a
2	Bidirectional LSTM	64	None
3	Dense	64	None

We applied L2 normalization to every layer in NN1, and then compiled it with the Adam optimizer and triplet semi-hard loss function. We then trained NN1 with X and Y, using 200 epochs and a batch size of 512.

The triplet loss function is the key to NN1's function as a feature extractor. Through its architecture, it transforms an input array of shape (*length* x 2) to a vector of shape (64). With this 64 dimensional vector, we can calculate a distance from the origin, or from another data point of equal dimension. The triplet loss function identifies three samples: an anchor a, a positive p, and a negative n. Generally, a and p will be two samples from the same class, while n is drawn from a different class. The loss L is then calculated as:

$$L(a, p, n) = max(0, D(a, p) - D(a, n) + margin) \tag{3.2}$$

where $D(x, y)$ represents the distance between points x and y [19]. What this loss function does is reward NN1 when the distances between samples from the same class are small, and distances from samples of different classes are large. Using this training paradigm, we used NN1 to extract features that we hope can differentiate samples based on their class.

The fully trained NN1 changed our input of shape (*num_samples* x *max_sample_length* x 2) to (*length* x 64), which enabled us to use more traditional machine learning approaches in the "true" training phase.

3.3.2 Identification

In identification, we attempt to determine which speaker, from a pool of potential suspects, a sentence can be attributed to based on previously captured samples from each potential speaker.

To accomplish this, using X, Y, X', Y', and NN1, we first calculated an expanded feature space, X'', by using NN1's predict function on X'. Then, X'' and Y' are further partitioned into x, y, x', and y' for training and testing, respectively.

Since NN1 learned how to create a more expressive feature space from the original input, we were able to transition to more basic machine learning algorithms in the second phase of this process. Therefore, we built a K-Nearest Neighbor Classifier (KNN) with K=1 neighbor, and trained it with x and y. Finally, we evaluated its performance with x' and y'. The results will be presented in Chapter 5.

3.3.3 Authentication

In authentication, we attempt to determine whether two sentences are spoken by the same person, or translated into the vernacular of this project, whether two sequences of packet lengths can be attributed to the same speaker with any degree of certainty.

Starting with X, Y, X', Y', and NN1, we use NN1's predict function on X' to create X''. We then partition X'' and Y' equally (in a stratified fashion) into x, x', y, and y' for the second stage of the authentication process.

Next, we compute the distances of each pair between x and x', and flatten the result into a 1-dimensional array called *dists*. Correspondingly, we create an array called *genuine*, such that for index i in *genuine*, *genuine*[i] is equal to True if *dists*[i] represents a pair from the same speaker, and False otherwise. If authentication is possible, we would expect the distances between sentences from the same speaker to be relatively small, while distances between sentences of different speakers would be larger. The best way to check for this relationship is via a Receiver Operating Characteristic (ROC) curve, which plots the True Positive (TP) rate in relation to the False Positive (FP) rate, where TP and FP are defined as:

$$TP = \frac{\#\ correctly\ identified\ samples}{\#\ of\ samples} \tag{3.3}$$

$$FP = \frac{\#\, incorrectly\, identified\, samples}{\#\, of\, samples} \tag{3.4}$$

Finally, we calculate such an ROC curve using *genuine* and *dists*. The results will be provided in Chapter 5.

3.3.4 Profiling

TIMIT provides two features for each speaker in addition to the unique speaker identification number: gender and dialect. Given that speech tends to differ based on these two categories, we attempted to learn to identify gender and dialect from the TIMIT dataset.

Gender

Starting with X, Y, X', Y', and NN1, we first calculated X'' by using NN1's predict function on X'.

For the second phase of gender profiling, we built a NN, called *GNN*, with architecture described in Table 3.2.

Table 3.2. GNN Architecture

Layer Number	Layer Type	# Nodes	Activation Function
1	Bidirectional Masking	n.a.	n.a
2	Dense	100	"relu"
2	Dense	100	"relu"
3	Dense	100	"relu"
4	Dense	100	"relu"
5	Dense	2	"sigmoid"

GNN was compiled with the Adam optimizer and binary cross-entropy loss function.

To avoid overfitting, we utilized stratified K-fold cross validation, with K=20 in this instance. Therefore, we calculated 20 train/test splits from X" and Y', and iterated through them; each round creating a new instance of GNN, training with 200 epochs, and finally testing it. The

scores from each round of testing were averaged together at the end. The final results are presented in Chapter 5.

Dialect

The TIMIT dataset contains labels for 8 different United States dialect groups: New England, Northern, North Midland, South Midland, Southern, New York City, Western, and Army Brat. We treated this as a standard multi-class classification problem and constructed a NN to solve it.

Starting with X, Y, X', Y', and NN1, we first calculated X'' by using NN1's predict function on X'.

The dialect NN, referred to as *DNN*, we built is very similar in structure to GNN. Its architecture is presented in Table 3.3.

Table 3.3. DNN Architecture

Layer Number	Layer Type	# Nodes	Activation Function
1	Bidirectional Masking	n.a.	n.a
2	Dense	100	"relu"
2	Dense	100	"relu"
3	Dense	100	"relu"
4	Dense	100	"relu"
5	Dense	8	"softmax"

DNN was compiled with the Adam optimizer and sparse categorical cross-entropy loss function.

To avoid overfitting, we utilized stratified K-fold cross validation, with K equal to 20. Therefore, we calculated 20 train/test splits from X'' and Y', and iterated through them; each round creating a new instance of DNN, training with 200 epochs, and finally testing it. The scores from each round of testing were averaged together at the end. The final results are presented in Chapter 5.

3.3.5 Amplitude Correlation

The final area of investigation for this project was the relationship between packet lengths and amplitude of the corresponding audio file. The underlying motivation for this line of questioning was to determine if we could detect periods of silence vs. speech happening for a user in a video call.

To accomplish this, we examined the correlation between a sequence of packet lengths and the sequence of amplitudes contained withing the corresponding .wav file from TIMIT. We started with a single label/list combination, where label is the absolute path to a TIMIT audio file, and list is the sequence of packet lengths collected during the transmission of that sentence. Using the *soundfile* Python library, we read in the .wav file referenced in label as a list of amplitudes and the sampling rate for the file. Figure 3.1 shows the results of plotting the amplitudes without any modification.

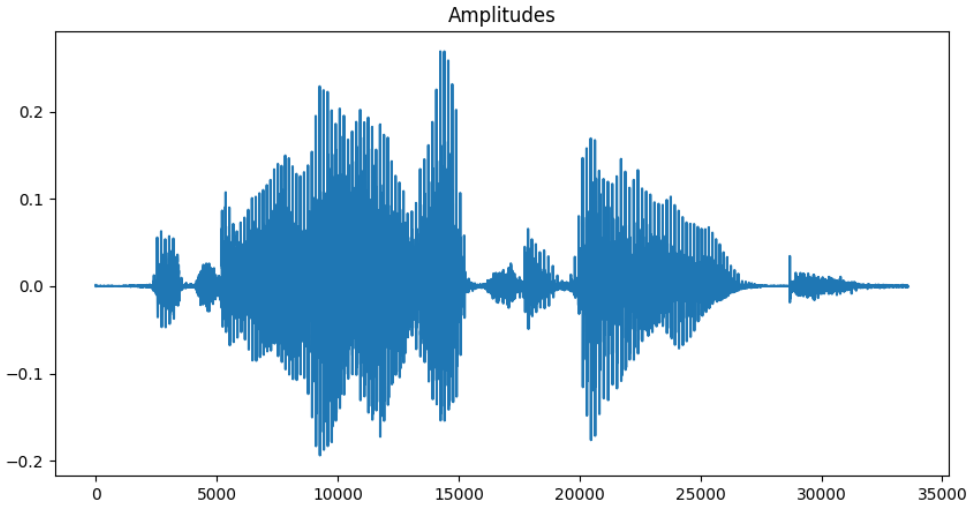

Figure 3.1. Amplitudes over time for a sample sentence. We calculated a moving average of these values to examine correlation with packet lengths.

The amplitude sequence contains many more values than the sequence of packet lengths, and therefore we cannot simply conduct a 1-to-1 correlation calculation. Our first solution was to

divide the amplitude sequence into n equally sized subsets, where n is equal to the number of packet lengths in *list*, and then calculate the mean for each of the amplitude subsets, providing us a moving average across the amplitude capture. This yielded two equally sized arrays for which we calculated a Pearson correlation coefficient. We performed this same process for each of the 6300 sentences in the TIMIT repository, and then calculated the mean, median, and standard deviation of the Pearson correlation coefficients.

Since we are most interested in detecting periods with no sound, which generally last much longer than 20 ms, we also explored dividing both the amplitude and packet length sequences into smaller subsets in order to reduce noise. To do this, we made a rough assumption that each packet represents 20 ms of time (see Chapter 4), and then recalculated the Pearson correlation coefficient for partitions of different time blocks (50 ms, 100 ms, 200 ms, etc.).

We present the results of this investigation in Chapter 5.

Pearson Correlation Coefficient

The Pearson correlation coefficient provides a measure for the linear relationship between two data samples. The coefficient r is calculated according to the equation below:

$$r = \frac{\sum(x - m_x)(y - m_y)}{\sqrt{\sum(x - m_x)^2(y - m_y)^2}} \tag{3.5}$$

where m_x is the mean of vector x and m_y is the mean of vector y [20]. Translating this into our amplitude correlation problem, we can consider x the sequence of packet lengths, and y the sequence of amplitudes. Therefore, calculating the Pearson correlation coefficient is an effective and popular method to measure the linear relationship between these two sequences.

The Pearson correlation coefficient is a number between -1.0 and 1.0, where -1.0 indicates a perfect negative relationship, 1.0 indicates a perfect positive relationship, and 0.0 indicates no correlation between the sequences. If the coefficient lies between ± 0.50 and ± 1.0, the sequences are considered to be strongly correlated. If the coefficient lies between ± 0.30 and ± 0.49, the sequences are considered moderately correlated. Any value below ± 0.29 indicates a low level of correlation [21].

CHAPTER 4:
Data Analysis

Before discussing the results of each experiment, we believe it will be beneficial to consider what the extracted data looks like. Data collection, described in Section 3.2, was performed for three of the most popular video conferencing platforms: Skype, Zoom, and FB Messenger. For each of the data captures, there are 6300 rows of data, corresponding to the 6300 unique speaker/sentence combinations in the TIMIT dataset. Each row of data contains a sequence of tuples in the following form: (timestamp, packet length). The sequences vary in length depending on the length of the sentence they represent, the talking speed of the speaker, and the sampling frequency of the audio codec. Section 4.1 details some general statistics for the resulting dataset we worked with, while Section 4.2 provides visualizations of the data in different formats.

4.1 Data Statistics

First, we examined the overall size of the data we collected, by looking at the number of tuples in each sequence, and the time span for each sample. We present these features in Tables 4.1 and 4.2, respectively.

Table 4.1. Sequence Lengths

Software	Mean	Median	Standard Deviation
Skype	163.33	157.0	44.78
Zoom	159.32	153.0	44.04
FB Messenger	116.32	112.0	31.3

Table 4.2. Sequence Times (s)

Software	Mean	Median	Standard Deviation
Skype	3.14	3.02	0.87
Zoom	3.23	3.12	0.86
FB Messenger	3.22	3.12	0.86

The lengths and times we observed from these sequences correspond to the information we have concerning the audio codecs in use by Skype, Zoom, and FB Messenger. This gives us confidence in the completeness and correctness of the capture. Both the SILK codec, used by Skype and Zoom, and the Opus codec, used by Zoom and FB Messenger, default to encoding 20 ms of speech per packet [9], [10].[4] Using the values from Table 4.1 and Table 4.2, we calculated the estimated sampling rates used by each of the software platforms, presented in Table 4.3. Both Skype and Zoom have values very close to 20 ms/packet, whereas FB Messenger has values almost a third higher at 27 ms/packet. However, while the mean and median are very close to the values we expect, the large standard deviations indicate that the amount of time each packet represents changed significantly during the course of data collection.

Table 4.3. Calculated Sampling Rates (ms/packet)

Software	Mean	Median	Standard Deviation
Skype	19.22	19.24	19.43
Zoom	20.27	20.39	19.53
FB Messenger	27.68	27.86	27.48

Next, we examined the magnitude of each packet length contained within the 6300 sequences of tuples in Table 4.4. Significantly, the values vary greatly between each of the platforms, which is curious given they are theoretically encoding the same information.

[4] As noted in Section 2.2.1, SILK can vary the number of frames per packet based on network conditions, between 20 and 100 ms per packet. Anecdotally, we do not see this occur in our data sample based on the observed time/sequence length ratio.

These large differences are likely the result of platform-specific design choices, such as metadata included in each packet. Since we do not test across platforms in this project, there is not a need to normalize the lengths.

Table 4.4. Packet Lengths

Software	Mean	Median	Standard Deviation
Skype	100.5	100.5	18.43
Zoom	233.98	233.98	71.71
FB Messenger	138.63	138.63	41.73

In Table 4.5, we examined the packet inter-arrival times for each platform: the difference in time between each packet arriving during transmission. As expected, these values align very closely to those presented in Table 4.3, with very slight variations. The similarity between the values, especially the high standard deviations, presented in each table affirm our previous observation that packet times vary significantly. This indicates that 1) each platform made large changes in how many milliseconds each packet encodes during the data capture or 2) that timestamps are an unreliable metric to use in training. Either case would have negative consequences on the outcome of this project.

Table 4.5. Packet Inter-arrival Times (ms)

Software	Mean	Median	Standard Deviation
Skype	19.36	23.08	12.57
Zoom	20.43	20.81	27.48
FB Messenger	27.95	30.74	9.58

4.1.1 Histograms

To view this data in a different way, we constructed histograms to illustrate the distribution of sequence lengths (i.e. number of packets per sequence) and time duration across the three platforms, shown in Figures 4.1 and 4.2, respectively.

In Figure 4.1 we see the sequence length distributions are very closely correlated for Skype and Zoom, while FB Messenger tends to have shorter sequence lengths. This corresponds with the data presented in Tables 4.1 and 4.4, which show shorter average sequence lengths and longer average packet lengths for FB Messenger, indicating that there is more information contained within each packet for that platform.

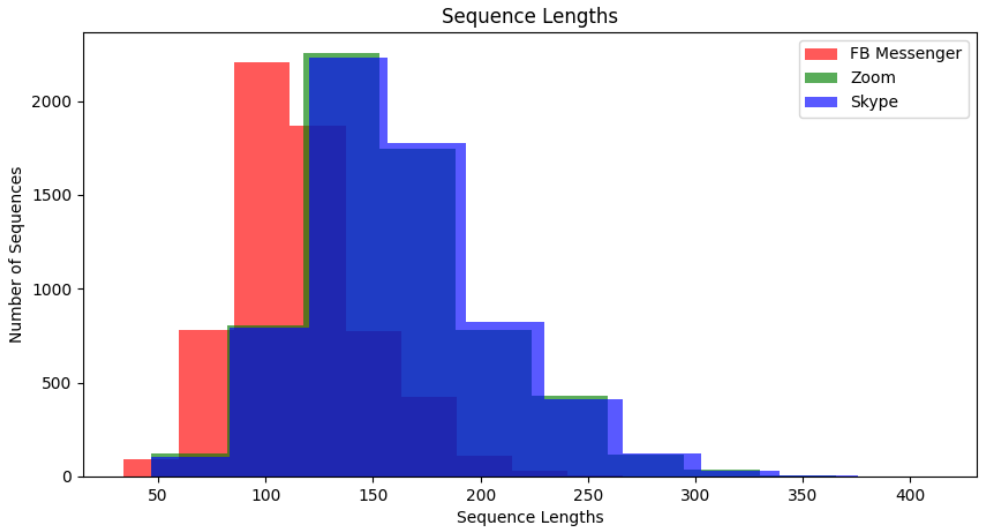

Figure 4.1. Histogram of sequence lengths. We see the distribution of sequence lengths across the three platforms. Notice that Skype and Zoom are very closely correlated, while FB Messenger tends to have a heavier distribution toward shorter lengths.

In Figure 4.2, we see that all three platforms have sequence times very close to each other. Intuitively, this makes sense because they are all transporting audio of the same length, and when using real-time protocols, must necessarily then take the same amount of time to transmit regardless of platform.

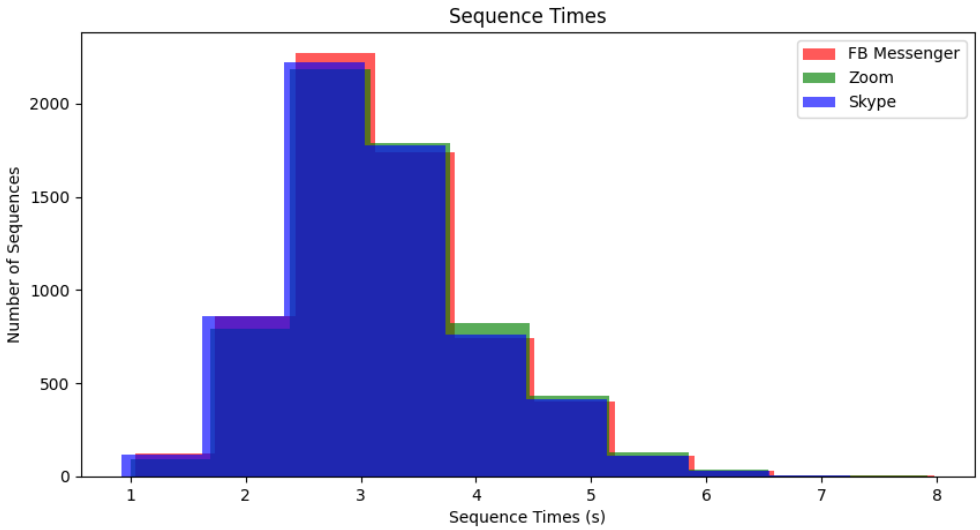

Figure 4.2. Histogram of sequence times. The distributions of sequence times across the three platforms are all very similar.

4.2 Visualization

As all of our efforts focused on extracting sequences of packet lengths from various speaker and sentence combinations and identifying the patterns that arose, we logically wondered how different these sequences are from one another. To get a glimpse at the data and how it relates to itself, we modeled three pairs of randomly selected sentences: two different sentences spoken by different people, two different sentences spoken by the same person, and the same sentence spoken by two different people. The following sections will examine these sentence combinations from a variety of perspectives.

4.2.1 Packet Lengths

First we examined the plots of packet lengths over time. This is the fundamental relationship we investigated and trained our models on throughout the course of the project.

Figure 4.3 shows a different sentence spoken by different people. As we might expect, the

two sequences vary quite significantly; there are local maxima in Sentence 1 in the same time as a local minima forms in Sentence 2.

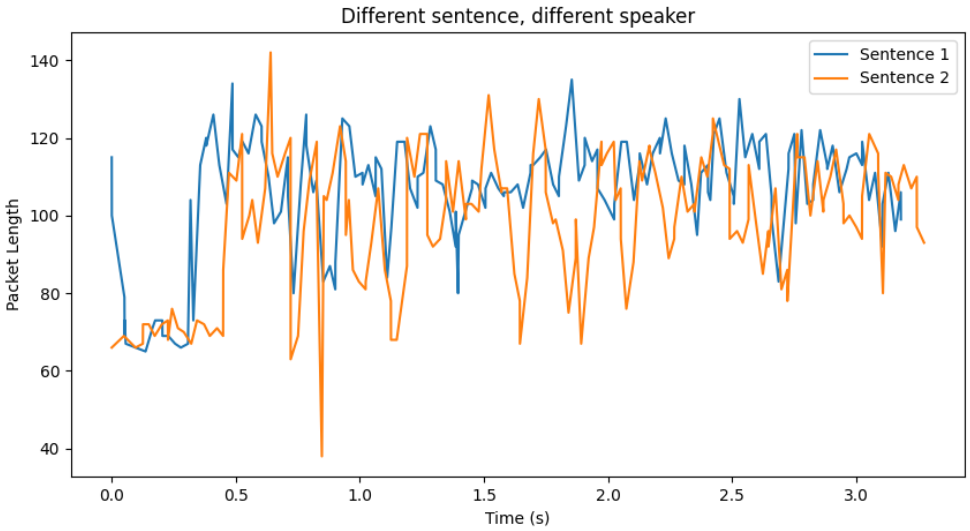

Figure 4.3. Variance in packet lengths over time for different sentences with different speakers. Two different speakers, one male from New England (Sentence 1), the other a woman from North Midland (Sentence 2), speak two different sentences. This graphic illustrates the changes in packet lengths over time for each speaker and sentence.

Figure 4.4 shows a different sentence spoken by the same person. Immediately we can tell the sentences are different as Sentence 2 lasts much longer than Sentence 1. However, we also see a tighter grouping of the two lines; despite being different sentences, they track much more closely with each other, which may indicate that the idiosyncratic differences between each person's speech can be accurately represented by the packet-length/phoneme exploit.

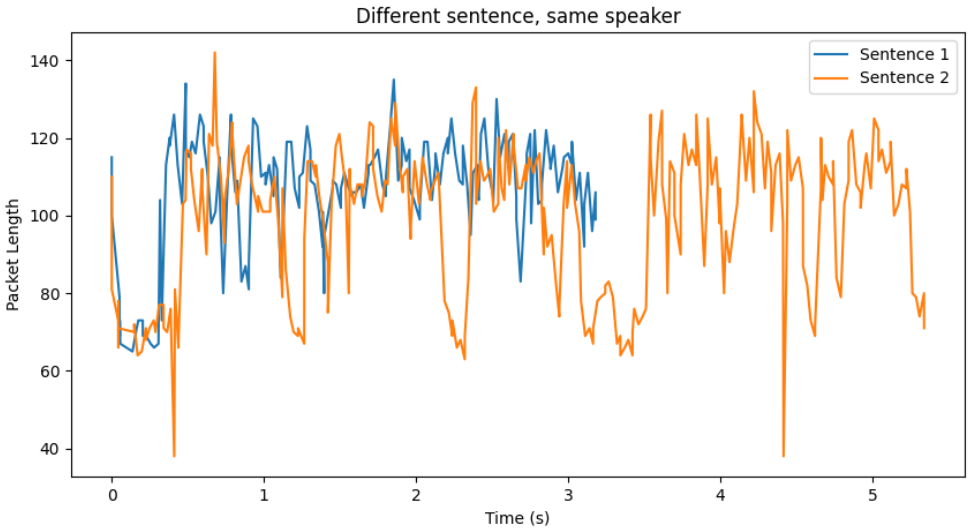

Figure 4.4. Variance in packet lengths over time for different sentences with the same speaker. One speaker, a male from New England (Sentence 1 and 2), speaks two different sentences. This graphic illustrates the changes in packet lengths over time for each sentence.

Figure 4.5 illustrates the same sentence spoken by two different people. We see, in line with the observations made with Figures 4.3 and 4.4, that there appear to be more significant differences when the speaker is different than when the sentence is different. Here, when the sentence is the same, we again see the phenomena where one sentence peaks while the other troughs.

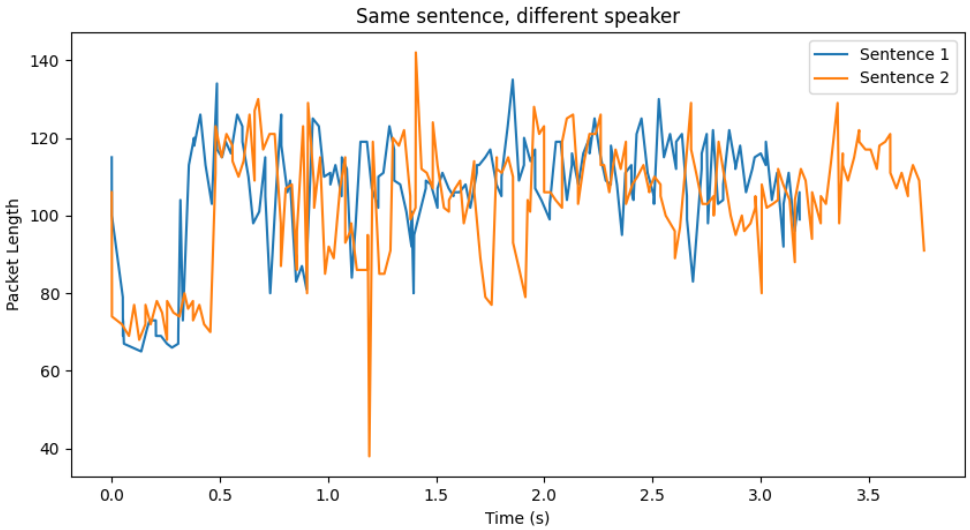

Figure 4.5. Variance in packet lengths over time for the same sentence spoken by different speakers. Two different speakers, one male from New England (Sentence 1), the other a woman from North Midland (Sentence 2), speak the same sentence. This graphic illustrates the changes in packet lengths over time for each speaker.

These plots were useful in helping us visualize the data and how it changes with regards to sentence and speaker selection. However, since we only examined 3 out of 19,216,900 possible sentence combinations, we can only use these plots as helpful examples and not definitive sources for the entire dataset.

4.2.2 Packet Inter-arrival Times

Though we primarily examined packet lengths in this project, we also took an interest in packet inter-arrival times, or the difference in time between one packet arriving and the next. Since the combination of packet lengths and inter-arrival times is what theoretically informs the sampling rate of the packet in question (which should be most closely tied to specific phonemes) we investigated the feasibility of using this approach while training. In this section, we perform the same visualization analysis as in Section 4.2.1, but plot

inter-arrival times over time instead of packet lengths over time. Figures 4.6, 4.7, and 4.8 provide this data.

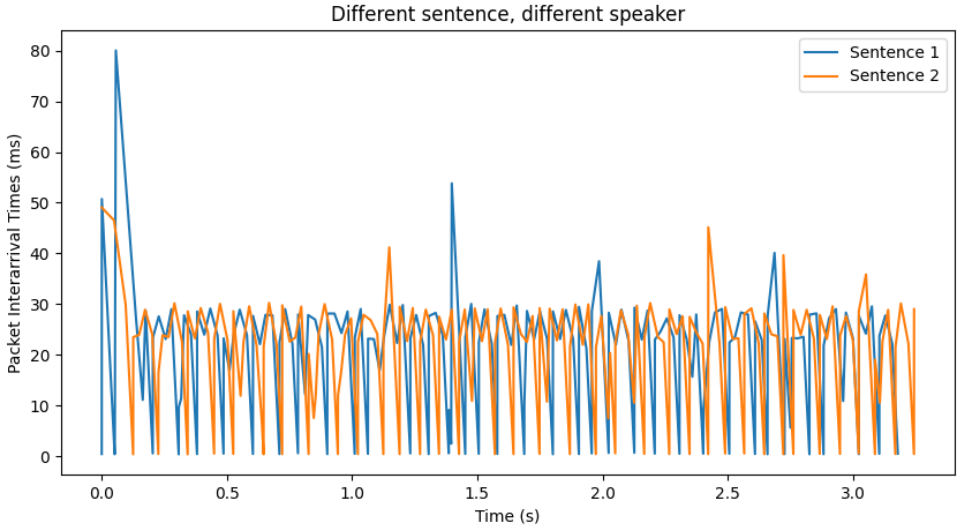

Figure 4.6. Variance in packet inter-arrival times over time for different sentences spoken by different speakers. Two different speakers, one male from New England (Sentence 1), the other a woman from North Midland (Sentence 2), speak the same sentence. This graphic illustrates the changes in packet inter-arrival times over time for each speaker.

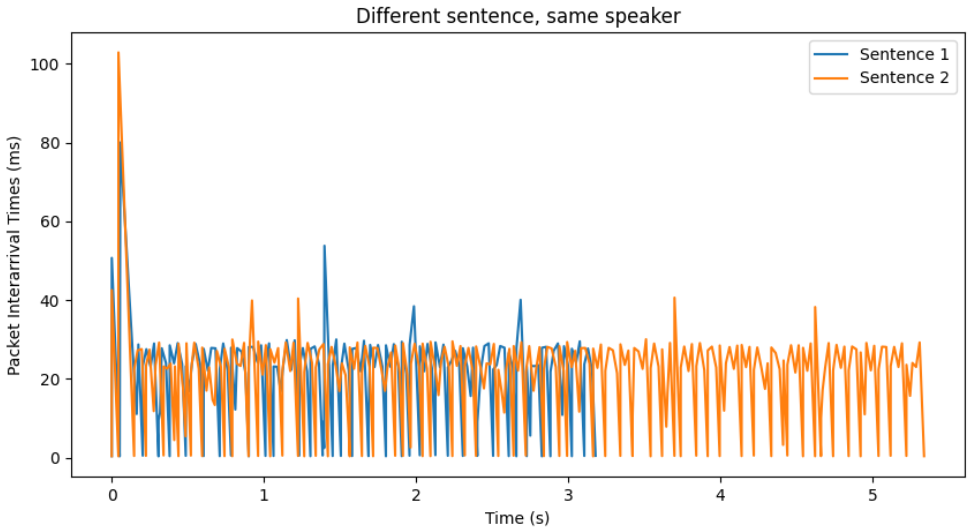

Figure 4.7. Variance in packet inter-arrival times over time for different sentences spoken by the same speaker. One speaker, a male from New England (Sentence 1 and 2), speaks two different sentences. This graphic illustrates the changes in packet inter-arrival times over time for each speaker.

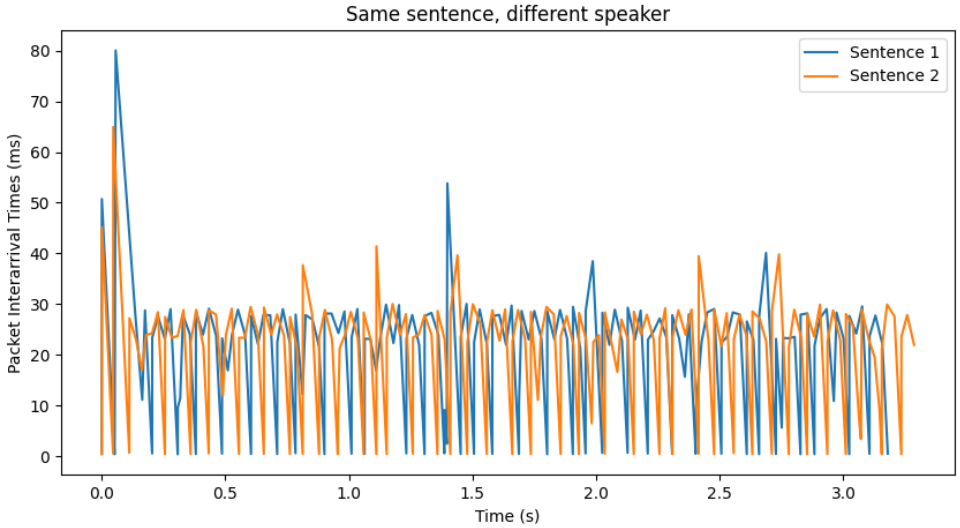

Figure 4.8. Variance in packet inter-arrival times over time for the same sentence spoken by different speakers. Two different speakers, one male from New England (Sentence 1), the other a woman from North Midland (Sentence 2), speak the same sentence. This graphic illustrates the changes in packet inter-arrival times over time for each speaker.

We see an interesting phenomena in all three of these plots, which obscures any patterns that might actually exist in the data. About every third point, the inter-arrival time drops down to near zero and then shoots right back up immediately after. This happened for every sentence we sampled, and is irregular behavior that we did not expect. While the reason for this continual, order of magnitude change in packet inter-arrival times is not certain, the implications are clear. If we cannot trust the packet inter-arrival times to accurately represent the time that each packet encodes, we cannot use these values to calculate an encoding rate. We are left with packet lengths alone as a training metric.

4.2.3 Cross-correlation

We also calculated cross-correlation between the sequences to get another view on their relationships to each other. Cross-correlation compares the values of two time series at

different time shifts, or lags. Correlation with lag k is defined as:

$$\sum_{n} x[n+k] \cdot y^{*}[n] \tag{4.1}$$

where $y*$ is the complex conjugate of y [22].

The following figures illustrate the correlation between three categories of sentence combinations: different sentences spoken by a different speaker, different sentences spoken by the same speaker, and the same sentence spoken by different speakers. These results are presented in Figures 4.9, 4.10, and 4.11, respectively.

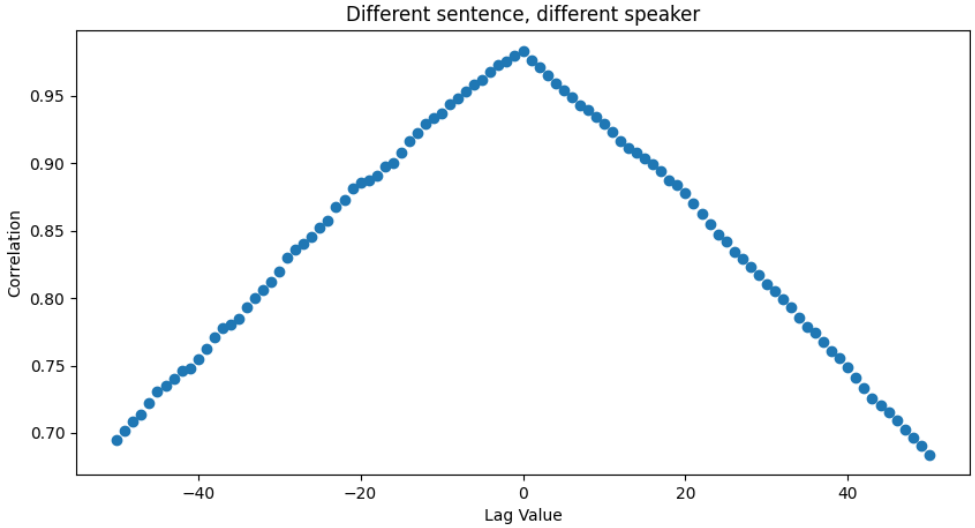

Figure 4.9. Cross-correlation of packet-length sequences for different sentences spoken by different speakers. Two different speakers, one male from New England (Sentence 1), the other a woman from North Midland (Sentence 2), speak two different sentences. This graphic illustrates the cross correlation between packet lengths of the two sequences.

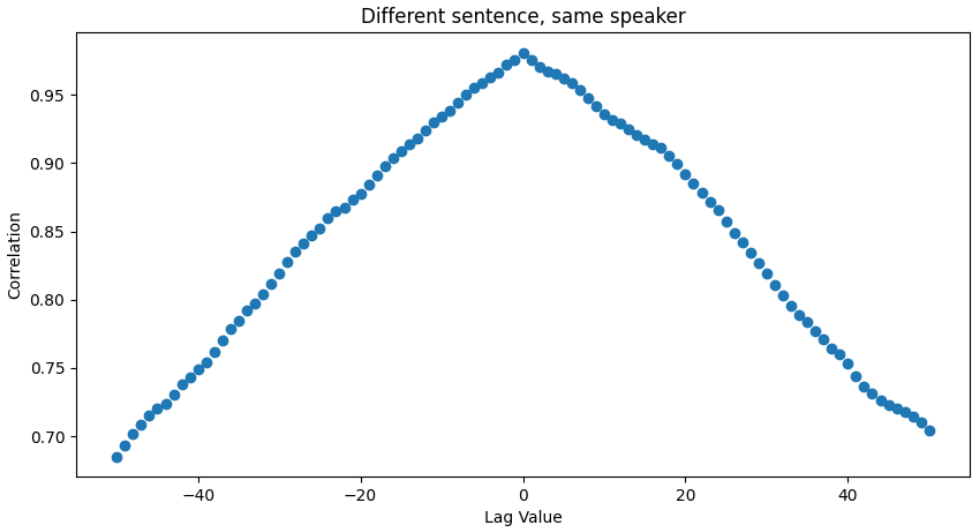

Figure 4.10. Cross-correlation of packet-length sequences for different sentences spoken by the same speaker. One speaker, a male from New England (Sentence 1 and 2), speaks two different sentences. This graphic illustrates the cross correlation between packet lengths of the two sequences.

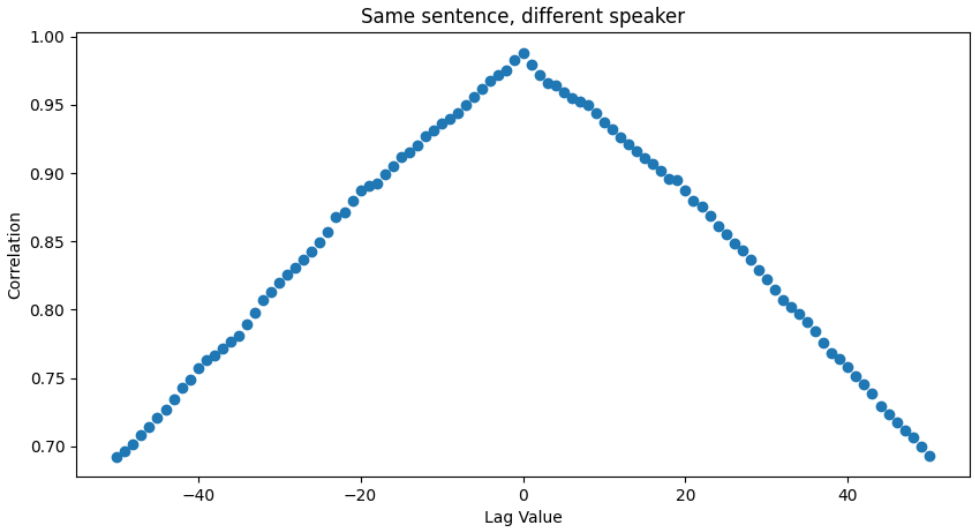

Figure 4.11. Cross-correlation of packet length sequences for the same sentence spoken by different speakers. Two different speakers, one male from New England (Sentence 1), the other a woman from North Midland (Sentence 2), speak the same sentence. This graphic illustrates the cross correlation between packet lengths of the two sequences.

We see that all three plots look very similar, which gave us an indication that cross-correlation between sequences of packet lengths would not be a worthwhile relationship to explore further as we developed our training methods.

CHAPTER 5:
Results

In this chapter, we will examine the results from each of our four domains of interest: identification, authentication, profiling, and audio amplitude correlation. We will show that our models for authentication, identification, and profiling have slightly better performance than a "random" classifier, but do not achieve significant levels of accuracy. However, we do discover and present a high correlation between audio amplitudes and packet lengths.

5.1 Identification

Identification performance is primarily measured by classification accuracy. We also calculate recall, precision, and F1 as secondary performance metrics. The following equations will define each of these metrics.

$$Accuracy = \frac{\#\,Correct\ Predictions}{\#\,Total\ Predictions} \tag{5.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision \times Recall} \tag{5.4}$$

To test the identification models we built (described in Chapter 3), we reserved 20 of the 630 total speakers, with 10 sentences per speaker, for the second phase of training and testing, with a 75% allocation to training. This resulted in 150 samples used to train and 50 to test. Table 5.1 presents the results of this testing. All metrics are presented on a scale from 0 to 1.

Table 5.1. Identification Results

Software	Accuracy	Recall	Precision	F1
Skype	0.04	0.04	0.0344	0.0341
Zoom	0.133	0.00667	0.0133	0.00889
FB Messenger	0.04	0.04	0.044	0.0412

An initial baseline to test any model's performance against would be a random guesser: for each example, randomly pick a label from the pool of potential labels. With 20 possible labels and 50 samples, we can calculate the expected accuracy of this method in the following manner: $\frac{\frac{1}{20}*50}{100} = 0.025$. Our identification model beat this simple metric for all three of the examined video chat platforms. However, performance does not progress much beyond that starting point.

5.2 Authentication

We assess the performance of our authentication approach (discussed in Chapter 3) by examining the ROC curves developed by computing distances between each pair of test sentences and applying a label (True or False) to that pair. Because an ROC curve plots the true positive rate against the false positive rate, a good result would show a high true positive rate relative to the false positive rate, especially on the left end of the graph. Figures 5.1, 5.2, and 5.3 present these curves.

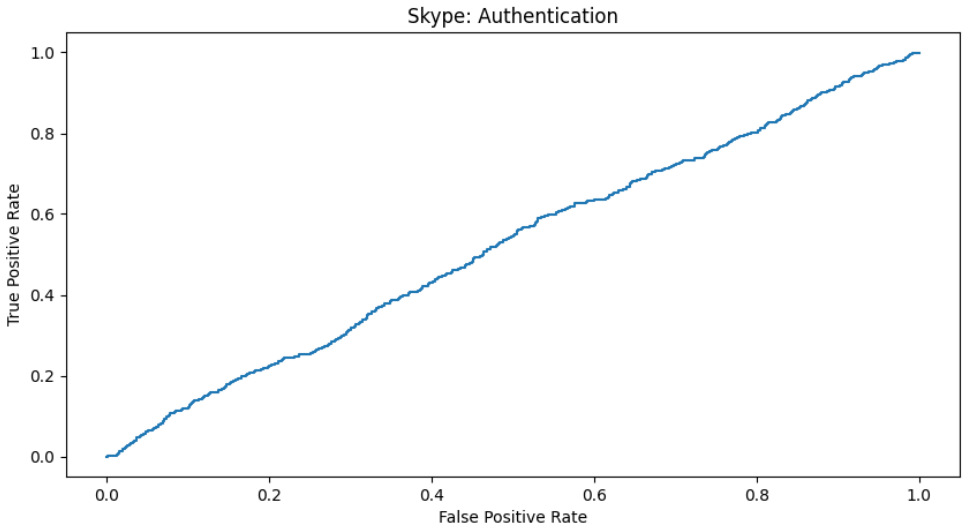

Figure 5.1. Skype ROC curve

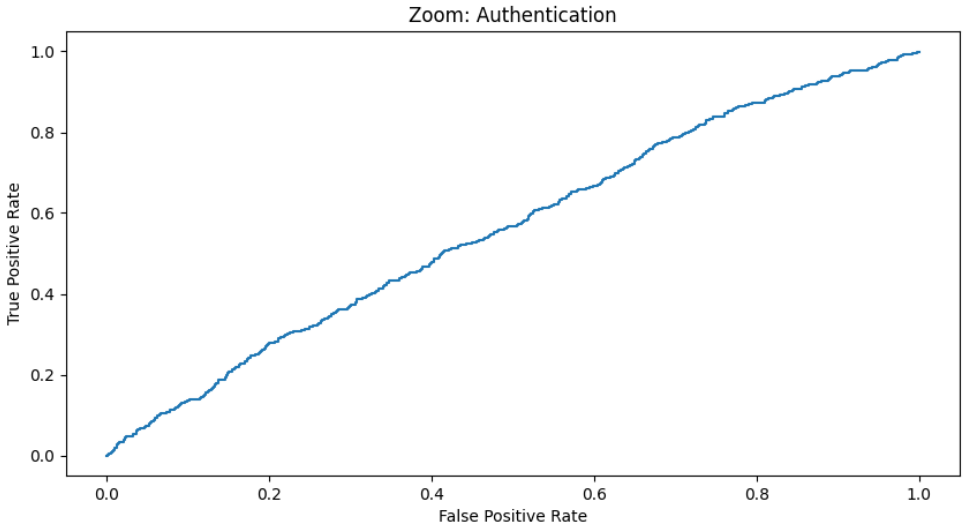

Figure 5.2. Zoom ROC curve

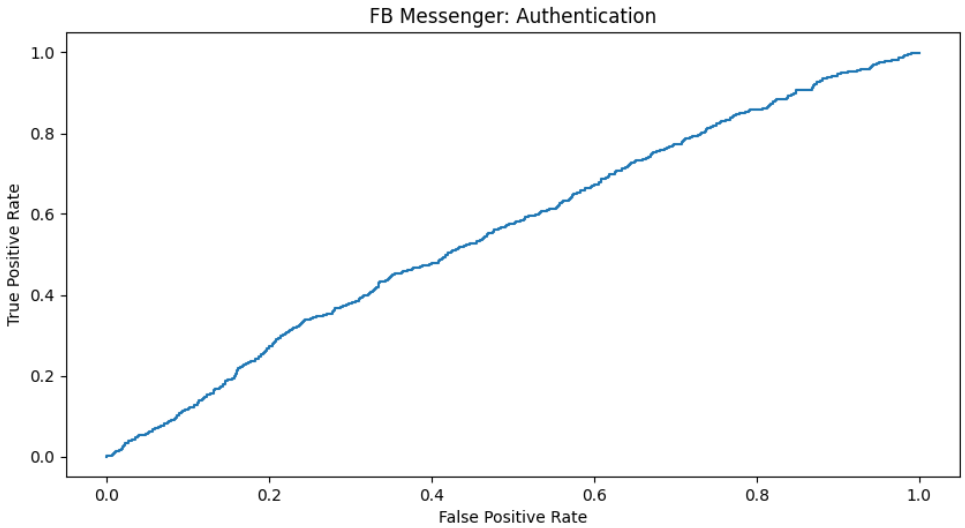

Figure 5.3. FB Messenger ROC curve

Another useful way to examine the results of an ROC curve is to calculate the area underneath the curve. Since we hope to see the true positive rate higher than the false positive rate as a sign of success, we would look for an area closer to 1.0 as a good result. Intuitively, an area of 0.5 would indicate a random model, where the true positive rate and false positive rate are completely in step. The areas for the curves presented above are contained in Table 5.2.

Table 5.2. Authentication Results

Software	Area Under ROC Curve
Skype	0.569
Zoom	0.547
FB Messenger	0.534

Based on a visual analysis of the graphs and the quantitative results provided in Table 5.2, we do not see any significant success in this area, with the curves barely beating the baseline y=x line, and the areas barely rising above 0.5.

5.3 Profiling

Before examining the results of our profiling models, we must examine the underlying male/female and dialect distribution of speakers in the TIMIT dataset. This data is provided in Table 5.3.

Table 5.3. Gender and Dialect Distribution of Speakers in TIMIT [11]

Dialect Region	# Male	# Female	Total
New England	31 (63%)	18 (27%)	49 (8%)
Northern	71 (70%)	31 (30%)	102 (16%)
North Midland	79 (77%)	23 (30%)	102 (16%)
South Midland	69 (69%)	31 (30%)	100 (16%)
Southern	62 (63%)	36 (37%)	98 (16%)
New York City	30 (65%)	16 (35%)	46 (7%)
Western	74 (74%)	26 (26%)	100 (16%)
Army Brat	22 (67%)	11 (33%)	33 (5%)
Total	**438 (70%)**	**192 (30%)**	**630 (100%)**

5.3.1 Gender

Results from our attempt at gender profiling are presented in Table 5.4. From Table 5.3, we know that the data is weighted 70% toward male speakers, which means the most basic baseline for our model would be a random generator that guesses male 70% of the time, yielding an accuracy of 0.70. The accuracy for all of our tests fell very close to this baseline value, which leads us to believe our model was not able to learn anything other than the underlying distribution of male to female speakers in the dataset, instead of any discriminative patterns in packet length sequences between the genders.

Table 5.4. Gender Profiling Results

Software	Accuracy	Recall	Precision	F1
Skype	0.654	0.654	0.594	0.600
Zoom	0.6833	0.6833	0.470	0.555
FB Messenger	0.690	0.690	0.625	0.610

5.3.2 Dialect

Results from our dialect profiling model are presented in Table 5.5. Because of the unbalanced distribution of dialects in the TIMIT dataset, establishing an initial "random" baseline to compare our results to is not as straightforward as with gender or identification. There are two options for this: 1) choose one of the eight labels with equally random chance for each sentence 2) choose the label with most occurrences for every sentence. Based on the option we choose, there will be differences across the four metrics we use to evaluate. The random baseline metrics are presented in Table 5.6.

Table 5.5. Dialect Profiling Results

Software	Accuracy	Recall	Precision	F1
Skype	0.164	0.164	0.088	0.106
Zoom	0.161	0.161	0.0875	0.0950
FB Messenger	0.167	0.167	0.095	0.0858

Table 5.6. Dialect Random Baselines

Option	Accuracy	Recall	Precision	F1
2	0.102	0.102	0.128	0.101
1	0.167	0.167	0.0279	0.0478

From the results presented in Tables 5.5 and 5.6, we see our model beating Option 1 in

accuracy, recall, and F1, while falling short of precision. We achieved parity with Option 2 in accuracy and clearly beat Option 2 in the remainder of the metrics. From these observations, we can safely claim our model to be as good or slightly better than a truly random classifier. However, its performance is still generally low.

5.4 Amplitude Correlation

As discussed in Chapter 3, we investigate the correlation between audio amplitude and packet length sequences broken into subsets based on time: 20 ms up to 500 ms.[5] Plotting segment size against the mean Pearson correlation coefficient allows us to see the effect of different spacing and select the optimal size for correlation based on the video conferencing software being examined. Figures 5.4, 5.5, and 5.6 present these lines for Skype, Zoom, and FB Messenger, respectively.

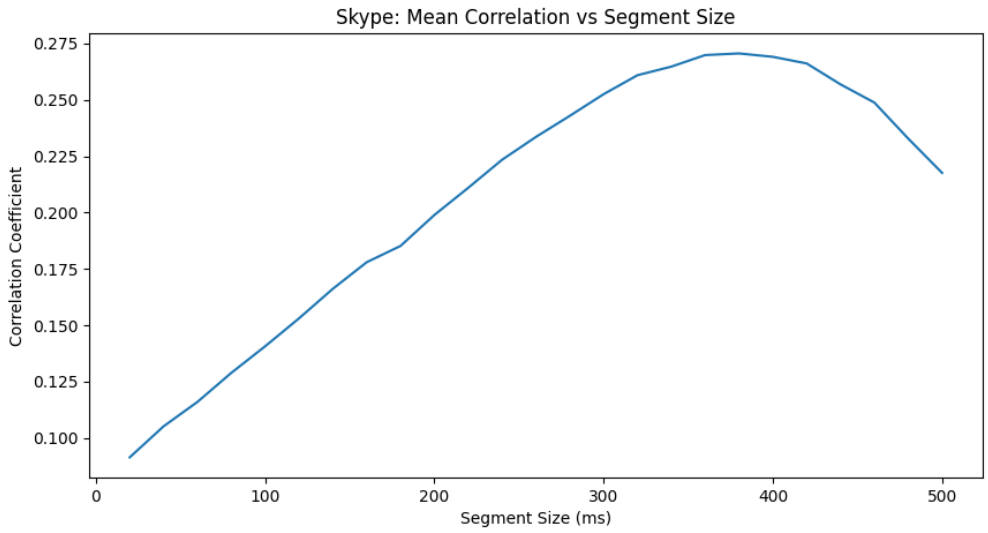

Figure 5.4. Skype correlations vs. segment size. Correlation peaks for Skype when we divide data into 380 ms chunks.

[5]We roughly estimate 1 packet equals 20ms based on calculations performed in Section 4.1. For example, to break the data into chunks of 40ms we took the total length ℓ of the packet length sequence, divided ℓ by 2 ($\frac{40\ ms}{20\ ms\ per\ packet} = 2\ packets$), and then separated both the amplitude and packet length sequences into $\frac{\ell}{2}$ subsets.

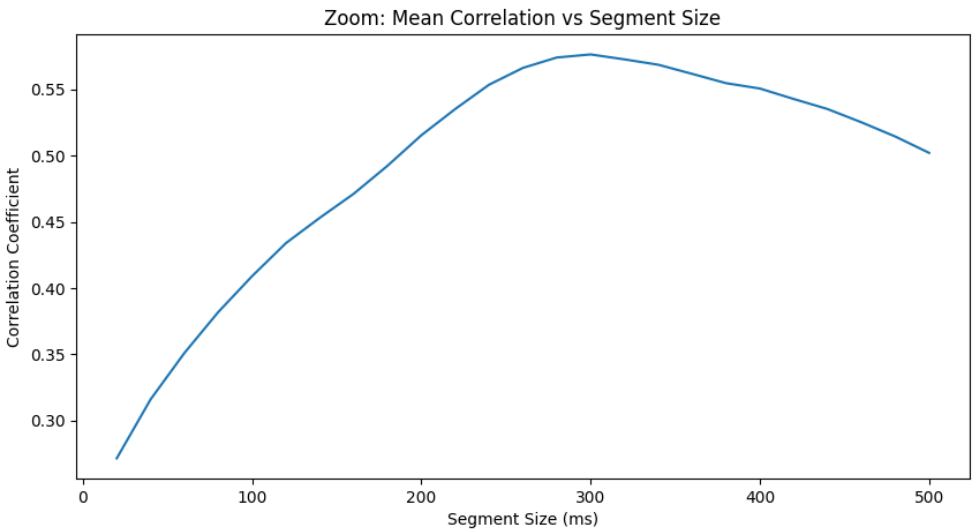

Figure 5.5. Zoom correlations vs. segment size. Correlation peaks for Zoom when we divide data into 300 ms chunks.

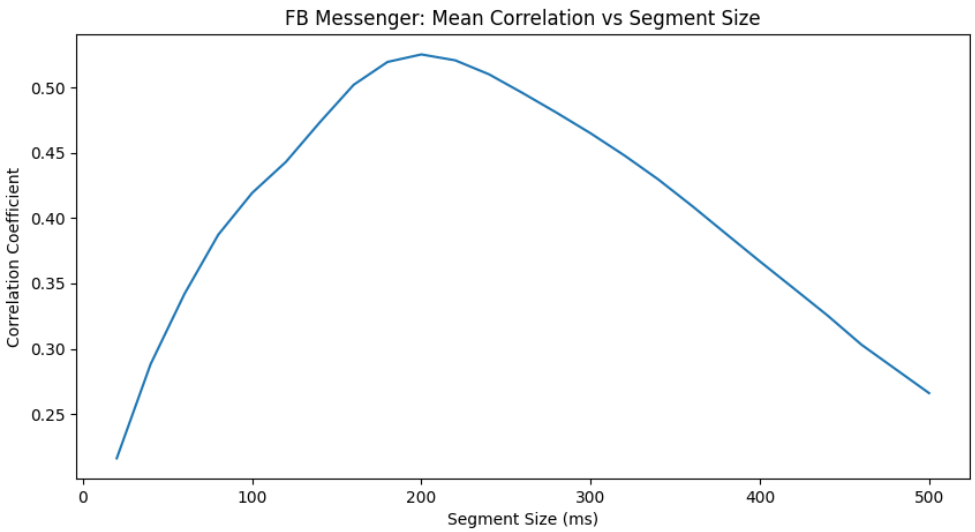

Figure 5.6. FB Messenger correlations vs. segment size. Correlation peaks for FB Messenger when we divide data into 200 ms chunks.

From these plots, we are able to determine the optimal segment size for each software platform. For Skype, we see optimal results at 380 ms (19 packets). For Zoom, optimal correlation occurs at 300 ms (15 packets). Finally, for FB Messenger, the optimal correlation happens at 200 ms (10 packets). Using these values, we calculate the maximum mean correlation for each platform, and present the results in Table 5.7.

Table 5.7. Maximum Correlations

Software	Mean	Median	Standard Deviation
Skype	0.271	0.317	0.292
Zoom	0.576	0.317	0.185
FB Messenger	0.526	0.538	0.161

As explained in Section 3.3.5, a Pearson correlation coefficient between ±0.50 and ±1.0 is considered strong correlation, a coefficient between ±0.30 and ±0.49 shows moderate

correlation, and a value below ±0.29 shows a low level of correlation between the two sequences [21]. Based on these thresholds, we see high correlation for Zoom and FB, and moderate to low correlation for the Skype packet sequences.

5.4.1 Example Correlation

To see the effect that different size subsets have on correlation in a more accessible manner, we examine a single sentence, looking at plots of its amplitude and packet lengths broken into subsets based on three different time intervals (20 ms, 100 ms, and 200 ms) in order to visualize how the correlation of packet lengths and amplitude changes in response to different time divisions.

Figure 5.7 represents our first attempt at correlation, treating each packet length as a discrete data point and breaking the amplitude array into the same number of subsets. This approach is very sensitive to changes, and there does not appear to be a high level of correlation, as sometimes the lines appear to track together, while at others they move completely opposite from each other. By visualizing plots like this, we decided to investigate the effect that creating fewer data points, and therefore smoothing the lines, would have on correlation.

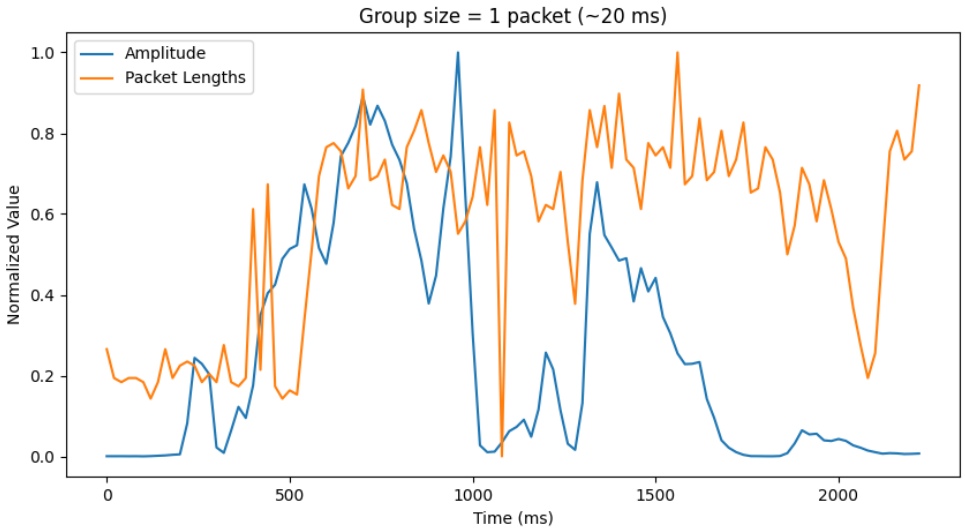

Figure 5.7. Example amplitude/packet length correlation with 20ms groupings. At such high granularity, correlation does not appear to be very high, and the lines appear to move independently of each other. The Pearson correlation coefficient for this plot is 0.286.

Figure 5.8 shows the results of reducing the number of data points by a factor of 5. Put another way, we create subsets that roughly represent 100 ms of data, using the conversion factor of 20 ms per packet. With this small change to the data, we immediately see much smoother lines, and importantly, growing correlation. There are still instances where the lines move in opposite directions, but overall they track together much more strongly.

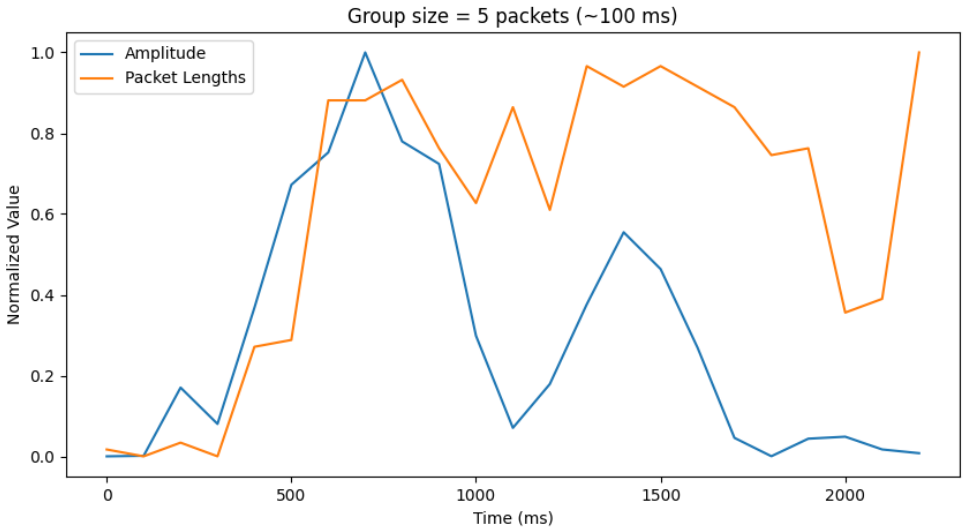

Figure 5.8. Example amplitude/packet length correlation with 100ms groupings. Splitting the data into groups of 100ms, we see the lines become smoother and a greater level of correlation to be revealed. The Pearson correlation coefficient for this plot is 0.379.

Figure 5.9 shows a more drastic reduction in data by a factor of 10, or splitting the data into groups of 200 ms. At this lower level of granularity, we see very smooth lines and even higher correlation.

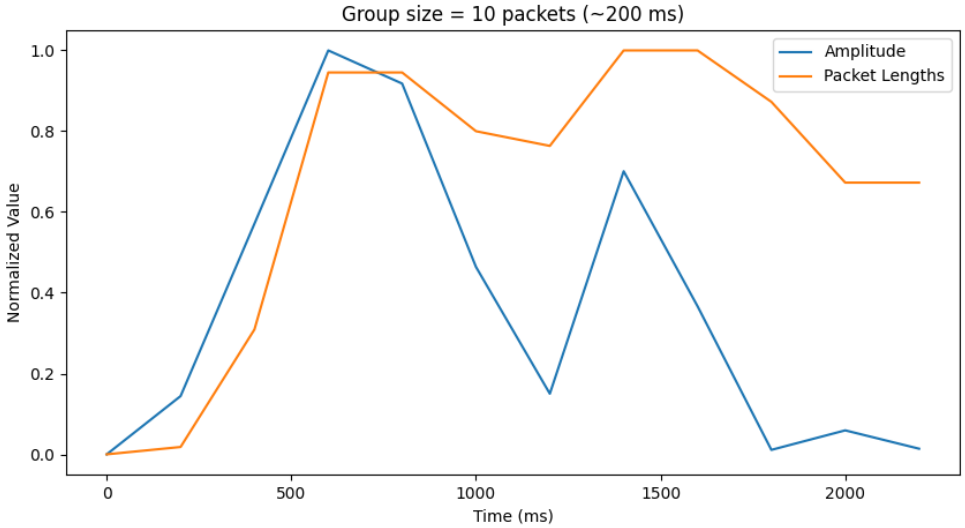

Figure 5.9. Example amplitude/packet length correlation with 200ms groupings. At this level of granularity, we see very smooth lines and even higher correlation. The Pearson Correlation Coefficient for this plot is 0.466.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
Conclusions

The goal of this project was to investigate and develop methods to perform identification, authentication, profiling, and amplitude correlation analysis based solely on captured, encrypted video conferencing traffic from platforms such as Skype, Zoom, and FB Messenger. To accomplish these tasks, we sought to leverage a serious leak of information that occurs when encoding and sending audio across the Internet: due to the use of VBR audio codecs, there is a high correlation between the size of an audio packet and the phoneme (building block of speech) that it encodes. Because SRTP does not pad the packet lengths before encryption, this correlation is carried forward and accessible to a third party eavesdropper.

We did not see any appreciable levels of success in identification, authentication, or profiling. However, we obtained very encouraging results by identifying a strong correlation between our extracted sequences of packet lengths and the amplitudes of the corresponding audio files, given the appropriate level of granularity. We achieved this granularity by splitting both the packet length sequences and the amplitudes into a smaller number of equally time-spaced subsets, which smooths out noise in both sequences while still providing enough detail to constitute meaningful results.

The remainder of this final chapter will elaborate on the implications of our results in Section 6.1, some limiting factors that may have contributed to our poor results in Section 6.2, mitigation against the exploits we explored throughout this paper in Section 6.3, and finally some areas of future work in Section 6.4.

6.1 Applications

Had all of the areas explored in this paper been successful, this section would be much more robust and important — it does not take much imagination to understand the serious implications of a third party listener being able to perform authentication, identification, or profiling simply by analyzing a stream of encrypted video conferencing traffic. However, since our efforts in these areas were unsuccessful, we will not elaborate on their implications in this section, and will instead focus on amplitude correlation.

The most interesting and potentially serious use of the correlation between packet lengths and amplitude would be to train a model that can reproduce the amplitude given an input sequence of packet lengths. Done perfectly, this would render the encryption moot, as we could replay the audio exactly as it was recorded. While intriguing, this is probably not possible for a number of reasons. First, the average number of packets for a sentence was around 150, while the average number of amplitude data points for an audio file was 49,000. This imbalance in granularity was proved by the extremely low correlation that happens on the left end of each plot in Section 5.4, where each packet is considered in its own group. Therefore, we do not expect packet lengths to convey enough information to reproduce the original amplitude.

A second way that this correlation could be deployed is by identifying periods of silence in a video call. Given the high correlation we achieved when examining groups of roughly 200-300 ms, we feel very confident in our ability to identify periods of silence for that same time period and longer. Pauses and periods of silence are central to a person's speaking style, and also generally unique to a speaker. Some people talk quickly, with few pauses, while others speak slowly or pause for effect often. Similar to the keystroke biometrics explored in *TypeNet*, a person might be identified through the frequency and timing of pauses while speaking [13].

Knowing when a user is or isn't speaking, can also be very useful, especially in group settings. In a call with many participants, the leader of the group (teacher, boss, or an enemy commander) will generally spend the most time talking. This information could be helpful when inferring the structure of a criminal or terrorist organization under investigation.

6.2 Limitations

Although amplitude estimation through packet size shows promise, user identification, authentication, and profiling remain challenging problems that can possibly be addressed through extensions to this work. This section will examine in detail the largest roadblocks we encountered during this project, and Section 6.4 will propose potential solutions.

6.2.1 Data Size

The most obvious limitation to this project was the size of the TIMIT dataset. While TIMIT contains an amazing diversity of voices and phonetically rich phrases, along with the accompanying phonetic breakdown for each sentence, the size is a major limiting factor when trying to use the data in a machine learning environment. From our data collection process, we only obtained 6300 samples of packet length sequences and corresponding labels—a typical machine learning project will train on millions of samples and then evaluate performance with a large, separate test set as well. In this project, we had to suffice with breaking up these 6300 samples into three groups: one to train our feature extraction model (NN1), one to train the secondary classifier, and finally a third group to test on. Seeing the first phase as the most important, we chose to allocate most of the data to that effort (6100 samples), but even that extreme weighting of data was not enough to drive training loss below 0.87 for any epoch. With the remaining 200 samples split between second phase training and testing, there was not much opportunity for a model to learn any patterns that the first phase successfully exposed, if any.

6.2.2 Rate Calculation

Section 4.2.2, which examines plots of our data capture based on packet inter-arrival times, highlights another limitation of the collected data. During our data collection process, we were able to extract a length and timestamp for each packet. As discussed in Section 2.1.2, the primary leak of information occurs due to the use of VBR codecs, and how they select different frequencies for different phonemes, such as a lower rate for a fricative like “f” or “s” than a vowel. Theoretically, through dividing the packet length by the corresponding packet inter-arrival time, we can derive an encoding rate, and therefore get closer to the source of the information leak. However, this did not work in practice.

When we closely examine Figures 4.6, 4.7, and 4.8, we see rapid fluctuation between packet inter-arrival times, from values close to 0 all the way up to 50 ms and above. With some inter-arrival times being orders of magnitude above or below the others, our ability to calculate the encoding rate with any degree of certainty disappears.

One explanation for this large disparity in inter-arrival times could be fragmentation happening at the first hop before we capture the traffic. If this were the case, fragments from

the same packet would have very small inter-arrival times, while fragments from different original packets would be further spaced out. However, a quick study of the captured traffic reveals the Don't Fragment (DF) bit present in every IP header carrying a UDP payload, which rules out this theory.

A second explanation for this phenomena is simple router congestion and scheduling causing some packets to be sent immediately, while other have to wait their turn. This is plausible, and supported by the fact that all of our traffic captures included packets not related to the video call, which had to be filtered out.

Regardless of the underlying reason for fluctuating inter-arrival times, it clearly limits our ability to get closer to the source information leak, forcing us to rely on packet length alone. In a machine learning setting, having more information (i.e. more features) is almost always conducive to obtaining good results. In this project, we had to operate with an extremely small feature space, which limited the ability of our models to succeed.

6.2.3 Variable Packet Timing

As discussed in Section 2.2.1, the SILK audio codec (used by Skype and optional for Zoom) has the option to vary the number of frames per packet based on network conditions: each packet could contain 20, 40, 60, 80, or 100 ms worth of encoded audio [3]. If this option is used, then packet length is no longer an indication of the underlying encoding bit rate; a larger packet could be the result of a higher encoding rate, or be because the packet contains a longer segment of audio, or some combination of the two that makes things even more challenging. Unfortunately, we have no way to know whether this option was used by Skype or Zoom during our data collection[6], but if it was then our ability to infer information from the sizes of packets would be severely limited.

6.3 Mitigation

The single easiest way to prevent the leak of information that forms the basis of this project and previous works is to pad the packets to a uniform length before encrypting.

[6]We assume that Skype and Zoom utilized the SILK codec, while FB Messenger used the Opus codec during our data collection. When examining the data presented in Chapter 4, we see that Skype and Zoom have very similar features, while FB Messenger is very different. We attribute this to the fact that FB Messenger was using a different audio codec.

SRTP intentionally does not pad packets because of the consequences that would have on bandwidth use. If SRTP padded every packet to the standard Maximum Transmission Unit (MTU) of 1500 bytes, this would cause an incredible increase of bandwidth use — for context, more than 90% of the packets we captured for this project had fewer than 300 bytes, so padding to 1500 bytes would cause a 5x increase in size. Also, if SRTP chose instead to pad to a smaller value, that could cause network slowdowns by artificially lowering the MTU, forcing more packets to be created and then be routed. Neither of these options is desirable, which is why SRTP does not pad packet lengths even though padding is generally considered to be a best practice in the security community. While padding packets would be costly in general use cases, this could be a viable option in instances when security is such a priority that network usage and performance is not a concern.

A second method of mitigation may already be in place, although perhaps not to specifically address the issue of information leaks. As described in Section 6.2, the SILK codec has the option to change the amount of time each packet encodes based on network conditions. If this option is used and the time per packet is changed dynamically during a call, then this would prevent an adversary from correlating packet length to the underlying encoding rate used by the codec because longer length could be due to higher frequency or simply more time. Without this key relationship in place, the entire chain of exploitation breaks down. A system could easily be configured to create this behavior purposefully and randomly to obfuscate the encoding frequencies in use. Additionally, this approach would come at a much lower network cost than configuring SRTP to pad all of its packets, and is therefore a much more reasonable general-purpose solution to the information leak problem.

6.4 Future Work

We believe there is still more work to be done in this area, and many of our suggestions for future work will be informed by the discussions in Sections 6.1 and 6.2. The first recommendation would be to apply the same methodology used in this project to a more appropriately sized data set that numbers in the millions of samples instead of the thousands. To collect and label the requisite audio samples would be an intensive process, but only with a larger set of data can we know whether the poor performance of our models was due to the limited data set or a flawed training paradigm.

The second area of investigation is building a model to label segments of traffic as conveying audio or silence. This project has laid the groundwork by proving a high correlation between packet lengths and audio amplitude, but the process of capturing both long periods of silence and audio transported by video conferencing software and then building models to classify them fell outside the scope and time frame of this project.

List of References

[1] S. Karapantazis and F.-N. Pavlidou, "VoIP: A comprehensive survey on a promising technology," *Computer Networks*, vol. 53, no. 12, pp. 2050–2090, 2009 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128609001200

[2] M. Schroeder and B. Atal, "Code-excited linear prediction(CELP): High-quality speech at very low bit rates," in *ICASSP '85. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1985 [Online], vol. 10, pp. 937–940.

[3] A. M. White, A. R. Matthews, K. Z. Snow, and F. Monrose, "Phonotactic reconstruction of encrypted VoIP conversations: Hookt on fon-iks," in *2011 IEEE Symposium on Security and Privacy*, 2011 [Online], pp. 3–18.

[4] "Zoom revenue and usage statistics (2020)," Mar 2021 [Online]. Available: https://www.businessofapps.com/data/zoom-statistics/

[5] C. Castrillon, "This is the future of remote work in 2021," Jan 2021 [Online]. Available: https://www.forbes.com/sites/carolinecastrillon/2021/12/27/this-is-the-future-of-remote-work-in-2021/?sh=5b51819b1e1d

[6] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st ed. USA: Prentice Hall PTR, 2000.

[7] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, 2008 [Online], pp. 35–49.

[8] F. Uenuma, "The long history of video chats-before coronavirus and Zoom," May 2020. Available: https://time.com/5834516/video-chat-zoom-history/

[9] "SILK codec." Available: https://www.vocal.com/speech-coders/silk/

[10] C. Hart, "Opus codec: The audio format explained," Mar 2021 [Online]. Available: https://www.wowza.com/blog/opus-codec-the-audio-format-explained

[11] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren, "DARPA TIMIT acoustic phonetic continuous speech corpus cdrom," 1993.

[12] L. Khan, M. Baig, and A. M. Youssef, "Speaker recognition from encrypted VoIP communications," *Digital Investigation*, vol. 7, no. 1, pp. 65–73, 2010 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1742287609000796

[13] A. Acien, J. V. Monaco, A. Morales, R. Vera-Rodriguez, and J. Fierrez, "Typenet: Scaling up keystroke biometrics," *2020 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 1–7, 2020 [Online].

[14] S. Chen, R. Wang, X. Wang, and K. Zhang, "Side-channel leaks in web applications: A reality today, a challenge tomorrow," in *2010 IEEE Symposium on Security and Privacy*, 2010 [Online], pp. 191–206.

[15] S. Cecconello, A. Compagno, M. Conti, D. Lain, and G. Tsudik, "Skype & type: Keyboard eavesdropping in Voice-over-IP," *ACM Trans. Priv. Secur.*, vol. 22, no. 4, Dec. 2019 [Online]. Available: https://doi.org/10.1145/3365366

[16] rtaft, "Share computer sound on skype," 2020 [Online]. Available: https://askubuntu.com/questions/1231447/share-computer-sound-on-skype

[17] alex232, "Play audio output as input to zoom," Sep 2020 [Online]. Available: https://unix.stackexchange.com/questions/558149/play-audio-output-as-input-to-zoom

[18] 2021 [Online]. Available: https://dpkt.readthedocs.io/en/latest/

[19] S. Das, "Image similarity using triplet loss," Mar 2021 [Online]. Available: https://towardsdatascience.com/image-similarity-using-triplet-loss-3744c0f67973

[20] "scipy.stats.pearsonr," April 2021 [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html

[21] "Pearson's correlation coefficient," Jun 2020 [Online]. Available: https://www.statisticssolutions.com/pearsons-correlation-coefficient/

[22] "matplotlib.pyplot.xcorr¶," May 2021 [Online]. Available: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.xcorr.html

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California