

## **Distribution Agreement**

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter now, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Zachary Zaiman

March 23, 2023

AudioStrike: Acoustic Identification of Keystrokes to Enhance  
End-to-End Session Integrity

By

Zachary Zaiman

Ymir Vigfusson Ph.D.  
Adviser

Computer Science

Ymir Vigfusson Ph.D.  
Adviser

Emily Wall Ph.D.  
Committee Member

Kristin Williams Ph.D.  
Committee Member

2023

AudioStrike: Acoustic Identification of Keystrokes to Enhance  
End-to-End Session Integrity

By

Zachary Zaiman

Ymir Vigfusson Ph.D.  
Adviser

An abstract of  
a thesis submitted to the Faculty of Emory College of Arts and Sciences  
of Emory University in partial fulfillment  
of the requirements of the degree of  
Bachelor of Science with Honors  
Computer Science  
2023

## Abstract

AudioStrike: Acoustic Identification of Keystrokes to Enhance  
End-to-End Session Integrity  
By Zachary Zaiman

The lateral movement strategy is one of the most pervasive attack techniques in a modern hacker’s arsenal. Generally, a point of entry is established through a phishing or social engineering attack to gain access to a target’s broader network from where more confidential and valuable information is obtained. Time and time again this method of exploitation has beaten the most complex systems with state-of-the-art intrusion detection software and security infrastructure due primarily to human error. To effectively defend against lateral movement attacks, we propose AudioStrike, a continuous and frictionless keystroke authentication architecture that utilizes the natural acoustic emanations of a user’s keyboard. We specifically show a proof of concept of this system on a single typist that achieves a 0.87 ROCAUC score of classifying keystrokes on three regions of the keyboard and can identify a potential attack within 5 keystrokes with high probability.

AudioStrike: Acoustic Identification of Keystrokes to Enhance  
End-to-End Session Integrity

By

Zachary Zaiman

Ymir Vigfusson Ph.D.  
Adviser

A thesis submitted to the Faculty of Emory College of Arts and Sciences  
of Emory University in partial fulfillment  
of the requirements of the degree of  
Bachelor of Science with Honors  
Computer Science

2023

## Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Ymir Vigfusson, for his unwavering support, encouragement, and guidance throughout my undergraduate honors thesis. I would also like to thank my committee members, Dr. Kristin Williams and Dr. Emily Wall, for their time, feedback, and constructive criticism, which have significantly improved the quality of my thesis. I would also like to thank Dr. Judy Gichoya, who sparked my interest and passion for research. Furthermore, I would like to acknowledge the invaluable contributions of other teachers at Emory, particularly those in the computer science department, who have enriched my academic experience through their courses, lectures, and mentoring. Their passion for teaching and research has fostered my curiosity and passion for the field of computer science. Thank you all for your support, encouragement, and contributions to my academic journey.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background</b>	<b>7</b>
2.1	Side Channel Attacks . . . . .	7
2.2	Security by Surveillance . . . . .	8
2.3	Threat Models . . . . .	8
2.3.1	Local Compromise . . . . .	9
2.3.2	Root Compromise . . . . .	9
2.3.3	Physical Compromise . . . . .	10
<b>3</b>	<b>Materials and Methods</b>	<b>12</b>
3.1	AudioStrike System . . . . .	12
3.2	Data Collection . . . . .	14
3.2.1	User Interface . . . . .	14
3.2.2	Data Collector . . . . .	15
3.2.3	Back-end . . . . .	23
3.3	IRB Study Design For Crowd Sourcing . . . . .	24
3.4	Model Training . . . . .	26
3.4.1	Metrics . . . . .	27
<b>4</b>	<b>Results</b>	<b>30</b>

4.1	Data Exploration . . . . .	30
4.2	Model Evaluation . . . . .	32
<b>5</b>	<b>Related Works</b>	<b>36</b>
5.1	Security Considerations . . . . .	36
5.2	Authentication . . . . .	37
5.3	Convolutional Neural Networks . . . . .	38
5.4	Audio Signal Processing . . . . .	40
5.5	Crowd Sourcing . . . . .	41
<b>6</b>	<b>Discussion</b>	<b>44</b>
6.1	System Validation . . . . .	44
6.2	Ethical Considerations . . . . .	46
6.3	Limitations . . . . .	47
6.4	Future Work . . . . .	48
6.5	Conclusion . . . . .	49
	<b>Appendix A Full Keystroke Distribution</b>	<b>51</b>
	<b>Bibliography</b>	<b>55</b>



# List of Figures

3.1	AudioStrike Validation Architecture . . . . .	12
3.2	Login Page . . . . .	15
3.3	Consent Page . . . . .	16
3.4	Registration Page . . . . .	17
3.5	Configuration Window . . . . .	18
3.6	Experiment Window . . . . .	19
3.7	AudioStrike Collector Architecture . . . . .	28
3.8	Keyboard Split . . . . .	29
4.1	Example Spectrograms . . . . .	32
4.2	Training and Validation Convergence . . . . .	33
4.3	ROC Curve . . . . .	34
5.1	CNN Architecture [26] . . . . .	39

# List of Tables

2.1	Comparison of Threat Models and Proposed Solutions . . . . .	11
4.1	Keystroke Distribution . . . . .	30
4.2	Keystroke Distribution After Binning . . . . .	32
4.3	Classification Metrics . . . . .	33
6.1	Attacker vs AudioStrike Predictive Power . . . . .	45
6.2	Event Probability . . . . .	46
A.1	Keystroke Distribution . . . . .	51

# List of Algorithms

1	Keylogger Sub-process . . . . .	18
2	Recorder Sub-process . . . . .	20
3	Collector Sub-process . . . . .	21
4	Audio Slicer Sub-process . . . . .	21
5	Spectrogram Conversion Sub-process . . . . .	22
6	Submission Sub-process . . . . .	23
7	Categorize Keys as LHS or RHS . . . . .	31

# Chapter 1

## Introduction

The internet is a dangerous place. According to the Federal Bureau of Investigation's (FBI) annual internet crime report in 2021, over 18.7 billion dollars was lost in the United States between 2016 and 2021, increasing every year. From 2019-2020, the financial loss due to cyber-crime rose 16.7% and from 2020-2021, losses rose greater than 40% [1]. In the world, cybercrime resulted in a loss of 20 billion dollars in 2021 alone. In 2023, Cybersecurity Ventures predicts the global cost of cyber-crime damage will be upwards of 8 trillion dollars [32]. However, these statistics do not even include the estimated 150 billion dollars spent by organizations worldwide to protect their systems. Furthermore, they fail to capture the global environment of cyber-attacks as well as the unreported number of successful or ongoing attacks. In addition to the individual reports made to the FBI, several multinational companies have reported breaches to their organizations in 2022 and 2023. For example, Uber, a multi-million dollar ride-share company, was breached by a social engineering attack despite using the latest defensive techniques like multi-factor authentication (MFA) [21]. This shows that even a company as large and established as Uber can still be compromised by an attacker employing social and technical strategies to breach a system equipped with the most modern security defenses. Current cyber-security

solutions need to be improved.

Cybercrime not only affects large corporations but also impacts the average active member of society. One kind of malicious software typically embedded in a system through lateral movement or human error, is called ransomware. Ransomware is a program that shuts down services or steals data unless the affected party pays the attacker a certain sum of money or fulfills their demands. These kinds of attacks skyrocketed 800% both in popularity and sophistication during the COVID-19 pandemic [16]. Ransomware plagued the healthcare, education, financial, and technology industries. In particular, hospitals and other healthcare infrastructure terrorized by ransomware attacks lost control of private patient data and had operations moved back to paper, resulting in patient harm and worsening the efficiency and efficacy of patient care at the peak of the pandemic, when the entire industry was already strained. In addition, consistently targeted industries like healthcare do not have the robust technological infrastructure internally to defend and recover from these attacks, elongating the already difficult and expensive process of rebuilding the damage inflicted by an attacker.

The most difficult and problematic element of cyber-security is the overwhelming lack of resources to combat well-organized attacks. Even with existing methods intended to prevent cybercrime, security teams struggle to prioritize particular areas of security over others and develop new defense tactics at the rate nefarious actors discover and exploit new vulnerabilities.

According to an annual security report by IBM in 2023, 9/10 cyber-attacks involve some kind of human error [2]. Companies are targeted daily in attacks similar to the one against Uber in a manner that appears almost uniform. Actors seeking to exploit vulnerabilities first secure a foothold, use this foothold to penetrate their target's system, and subsequently access confidential data or plans until they reach their target user. Even with the development of highly advanced infiltration detection systems,

code vulnerability detectors, and other preventative measures, it is impossible to remove human error from the equation. Rather than omitting human interaction from the calculus of a secure system, the most effective solutions will leverage frictionless systems to ensure technical security as well as minimize the effects of human error as much as possible. Furthermore, the most effective systems will not change existing user interactions with a system or service but rather complement those that are already in place.

**Lateral movement** is a popular exploitation technique for modern attackers, especially against commercial targets. Attackers leverage a single breach to a user on a network to migrate from user to user, gradually performing reconnaissance, and monitoring user data and valuable assets while working to remain undetected for as long as possible. Because adversaries need to fly under the radar while accessing a system, they will often avoid techniques that will set off signature-based alarms. As such, attackers rely heavily on system access to steal passwords, escalate privileges via remote machines, and access benign tools - such as PowerShell or Windows Management Instrumentation - already installed on host devices [20].

While there are possible defenses to these kinds of attacks such as automated intrusion detection systems, cloud security frameworks, and machine learning-based monitoring systems [31], each depends heavily on complex algorithms and expensive computer infrastructure designed to detect potential attacks across an entire organization. Consequently, the ability for attackers to stay hidden inside a system and carry out this type of attack on an end-user remains quite high. This poses a difficult challenge for security researchers and engineers attempting to prevent lateral movement. In many cases, early detection is vital in stopping attackers from doing severe damage to a system. One way security teams try to hinder an attacker from compromising an endpoint or service is through authentication techniques [20].

Traditionally, authentication relies on one or more of the following: something

you know (such as a password or answer to a question), something you have (such as a physical key or card), and (something you are such as a retinal scan or other biometric) [23]. More modern authentication techniques like multi-factor authentication (MFA) relies on two or more of the above to authenticate, providing a system or service an additional source of truth during the authentication process. However, having two or more of these safeguards in place does little to prevent targeted social engineering and other types of cyber-attacks that are more technical.

Another layer to this challenge is security fatigue. Steven Furnell defines the term as

...the situation in which users of systems and staff in organizations can tire of dealing with security or encountering messages and warnings in relation to it [22].

Not only do researchers have to account for the increasing complexity of cyber-attacks, but they also have to do so without introducing substantial friction in their solutions.

It is also important to note the threat models for which these solutions are designed. Consider two-factor authentication (2FA) that uses an external authenticator app on a user's mobile phone. This threat model assumes that the mobile phone is not compromised so that even if a user's password was breached, an attacker would not be able to accept the authentication request. However, if the mobile device is compromised as well, 2FA has no effect. This predicament demonstrates the significance of assessing the level of permissions a malicious actor can obtain. If an attacker has the power to compromise a user account, administrator-level software solutions can prevent the escalation of malicious actions. Alternatively, if an attacker somehow manages to elevate their permissions to the administrator level, they can disable, bypass, or trick any existing protections.

Solutions exist both in industry and academia [33] to combat lateral movement on locally compromised or non-root compromised systems, but these solutions become

inadequate if a malicious attacker can elevate their permissions to the administrator level because the attacker would have complete control over the target’s operating system.

A possible complementary defense to this kind of attack is to leverage side channels. Side channels have traditionally been used as an exploitation strategy, leveraging a physical flaw in the implementation of a system to gain information or carry out some type of attack [15]. For example, there was a famous side channel attack from the 1950s known as operation ENGULF [28] where a bug placed by MI5 was used to decipher the Hagelin cipher machine based on its acoustic emanations. More recently, side channels have been used to exploit a variety of systems including procuring private keys [10], keystroke patterns [30], and even escalating to administrator permissions [19].

One particularly notable side channel is keyboard emanations. Keyboard emanations refer to the acoustic signals that are unintentionally generated by a keyboard when a user presses a key. Asonov and Agrawal showed that it is possible to use machine learning to detect what keys a user types based on sound using a mechanical keyboard. Although capturing these signals can be considered eavesdropping and is therefore ethically questionable, consider the enormous potential security applications. Assuming that the typing pattern for every user is different, the acoustic emanation of each keystroke could be used to verify the authenticity of a keystroke without needing user interaction on a channel completely outside of a malicious actor’s view.

With this in mind, **we propose AudioStrike, a system which leverages a convolutional neural network (CNN) audio classifier to predict the region of the keyboard where each keystroke was pressed.** Using that prediction, AudioStrike can give a confidence score on how likely it is that the keystroke is secure. We also show that given an arbitrarily long sequence of keystrokes, the probability



of fooling the system decreases exponentially relative to the number of keystrokes pressed.

The contributions of this thesis are as follows:

1. We describe the AudioStrike architecture for verifying keystroke authenticity on a root-compromised system (See chapter 3.1).
2. We provide under open-source a Windows and Mac (Intel/Silicon) tool for scalable and privacy-preserving keystroke data collection (See Chapter 3.2).
3. We detail the study design for large-scale user-centric evaluation. The results are pending IRB approval (See Chapter 3.3).
4. We analyze the results from a proof of concept evaluation on an AudioStrike prototype that shows the approach can achieve an average ROCAUC score of 0.87 from distinguishing left-hand side, right-hand side, and space keyboard regions allowing for detection of an attacker within 5 keystrokes (See Chapter 4).

All implementation ideas and experimentation was conducted by the author under the guidance of Dr. Vigfusson. No part of KeyStrike source code or intellectual property was used in this project.

# Chapter 2

## Background

### 2.1 Side Channel Attacks

Traditional exploits make use of flaws or vulnerabilities in an implementation of an application or service to breach a system. This often has to do with the logical implementation of a program rather than the physical implementation or configuration of the system. Side channel attacks on the other hand take advantage of the physical system to extract information or manipulate the execution of a program running on that system [7]. For example, Yuval Yarom and Naomi Benger showed that it was possible to use a side channel to allow an attacker to discern if a piece of information was requested previously to recover the private key from an elliptic curve algorithm implemented by the OpenSSL library [37]. Common defenses for these attacks involve removing the side channel or obfuscating the correlation between the side channel and the software-defined operation. Unfortunately, these defenses are not universal and do not always have the potential to be implemented without damaging the underlying functionality of the system.

In 2004, Dmitri Asonov and Rakesh Agrawal published a seminal paper on side-channel attacks. They trained a neural network to identify keystrokes on PC and

ATM keyboards using only the sound of key presses and releases [5]. To train the neural network, they used state-of-the-art audio processing techniques at the time and were able to classify keystrokes correctly in their experiments with 79% accuracy. Even though their experiments were conducted on a small scale, Asonov and Agrawal demonstrated the vulnerability of essential components of a computer that do not have the processing power or any obvious logical vulnerability. This poses another difficult challenge for security engineers and researchers: a system can be logically secure to the highest possible degree, but still be vulnerable to attacks through side channels.

## 2.2 Security by Surveillance

Side channels are most commonly used as a method of exploitation on logically sound systems. They have been a way for attackers to completely bypass any defenses on a system simply by moving around them. Side channels, however, have not been typically used for securing these systems. By leveraging side channels as a defense mechanism, security engineers and researchers have the potential to implement external defenses to their systems. This also removes any logical dependency from the defense to the system itself.

## 2.3 Threat Models

In this work, we consider three potential threat models: a locally compromised system, a root compromised system and a physically compromised system.

### 2.3.1 Local Compromise

We define a locally compromised system to be one that has been breached by a malicious actor but only at the user level. This means that the attacker has access to the user's data and permission levels but does not have administrator privileges. In this context, an attacker can impersonate the target user but cannot change or disable available defenses set by the system's administrator. Consequently, even without administrator privileges, if a system does not have the proper defenses, an attacker can still damage the system or organization they have breached.

To secure a locally compromised system, a new technology has been patented called KeyStrike [33]. KeyStrike uses an external device or administrator-controlled software to capture a user's keystrokes as they are entered. Before releasing the keystrokes to an application, every keystroke is signed by an external service to verify its authenticity. Essentially, 2FA is performed on every keystroke and will alert and suppress unverified keystrokes. This ensures continuous and frictionless authentication on a potentially locally compromised system as the software would continuously ensure that a user is who they present to be. Furthermore, since the software is run at the administrator level, a user would not need to interact with it. Therefore, a user with KeyStrike protection would not be vulnerable to a lateral movement attack as long as the attacker was not able to escalate their permissions on the target system.

### 2.3.2 Root Compromise

We define a root compromised machine to be one in which the attacker has user access as well as administrator access. This would allow them to bypass any defenses put in place against lateral movement attacks; therefore, it is almost impossible to programmatically defend against an attack as the attacker has full control of the operating system and could disable or alter the protection without the knowledge of the target. While the KeyStrike solution would work in a locally compromised

setting, if an attacker were to somehow elevate their permissions to an administrator, they could turn off KeyStrike and even modify its software to successfully verify all of the user’s keystrokes. To defend against a machine that may be root compromised, we propose AudioStrike, an acoustic-based keystroke verification system. The theory behind this system uses the same hypothesis that Asanov and Agrawal made in 2004: each key on a keyboard produces a slightly different sound. We build on this hypothesis, applying state-of-the-art convolutional neural networks (CNN) and audio signal processing techniques to differentiate between different keystrokes. We also explore the robustness of our model, propose a framework that utilizes it to perform secure a user against a root compromised attack, and validate the model against a single typist’s data.

### **2.3.3 Physical Compromise**

Finally, we define a physically compromised system to be one in which an attacker has physical access to the machine. These types of attacks are arguably the most difficult to defend against as there is no known way to distinguish a user from an attacker if they are both sitting at a terminal. If a machine is physically compromised, neither KeyStrike nor AudioStrike would be a suitable defense as both rely on external devices to identify and stop an attacker.

To defend against a physically compromised system we propose a strategy called: defender in the middle (DiTM). A defender in the middle uses the same properties as a man in the middle attack but leverages them to defend a system. It accomplishes this by placing a mobile device in between the keyboard and a user’s machine such that every keystroke would pass through the mobile device as an extra layer of security. That way, when a user leaves their machine and inevitably takes their mobile device with them, they are removing the opportunity for an attacker to leverage a physical attack vector.

Table 2.1: Comparison of Threat Models and Proposed Solutions

Threat Model	KeyStrike	AudioStrike	DiTM
Locally Compromise	✓	✓	✓
Root Compromise		✓	
Physically Compromise			✓

## Chapter 3

# Materials and Methods

### 3.1 AudioStrike System

After data has been collected and an accurate keystroke region classification model has been trained, the model can be deployed into the AudioStrike keystroke validation system.

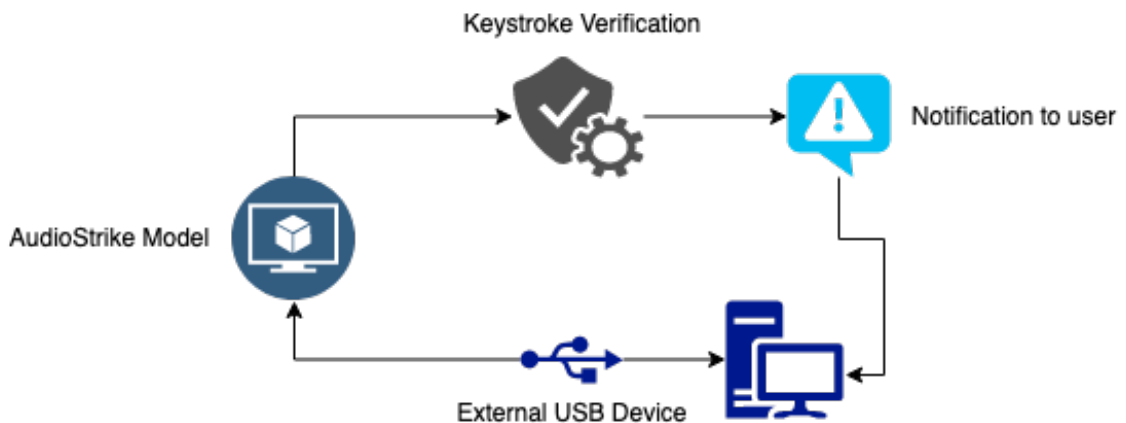


Figure 3.1: AudioStrike Validation Architecture

Figure 3.1 shows the proposed keystroke validation architecture. The AudioStrike processes would run on a trusted external device (like a smartphone) or a manufactured USB drive with the capability to have a microphone channel. At a high level,

the model trained in the above steps would either be quantized [24] and deployed on the external device, or deployed as an API on a trusted server unique to the user. This approach creates a secure channel that is inaccessible to a potential malicious actor who has access to the user’s machine. In either setting, the model deployed could then be used for inference and cross-referenced with the keystroke that was typed locally. If the system was disabled or if the probability that the key typed was on the wrong region of the keyboard, the system would alert the user on the external device that a possible attack was in progress and instruct them on actions to mitigate the threat.

There is an implicit trade-off between privacy and efficiency in the model quantization approach versus the external server approach. In the model quantization approach, a user’s model is stored only on a trusted external device. To accomplish deployment in the AudioStrike architecture, the model would have to be quantized or distilled in order to function properly in an environment that may not have access to resources typically needed for deep learning model training and inference like GPUs. This benefits the user’s privacy as there is one less hop required between registering a keystroke event and identifying a potential attack. However, model quantization often comes at the cost of model performance. Quantization is also not studied as robustly as typical deep learning so the existing resources and frameworks would make continuously training or updating an AudioStrike model challenging. In the external server approach, the model would have to be committed and deployed to an external server which may slightly increase prediction latency. However, the model performance would not decay like in the quantization approach and a server could leverage powerful computational resources to perform inference much more efficiently.



## 3.2 Data Collection

In order to train an effective model for AudioStrike, a large and diverse typing dataset is needed. Since public typing data sets with adequate baseline labels are not publicly available, we implemented a desktop application that could securely collect a user’s typing data. This application has three main components: the user interface, the underlying data collection utilities, and the cloud-hosted backend. Each component of the application enables the secure collection of typing data from many different participants in parallel. The full architecture can be seen in Figure 3.7.

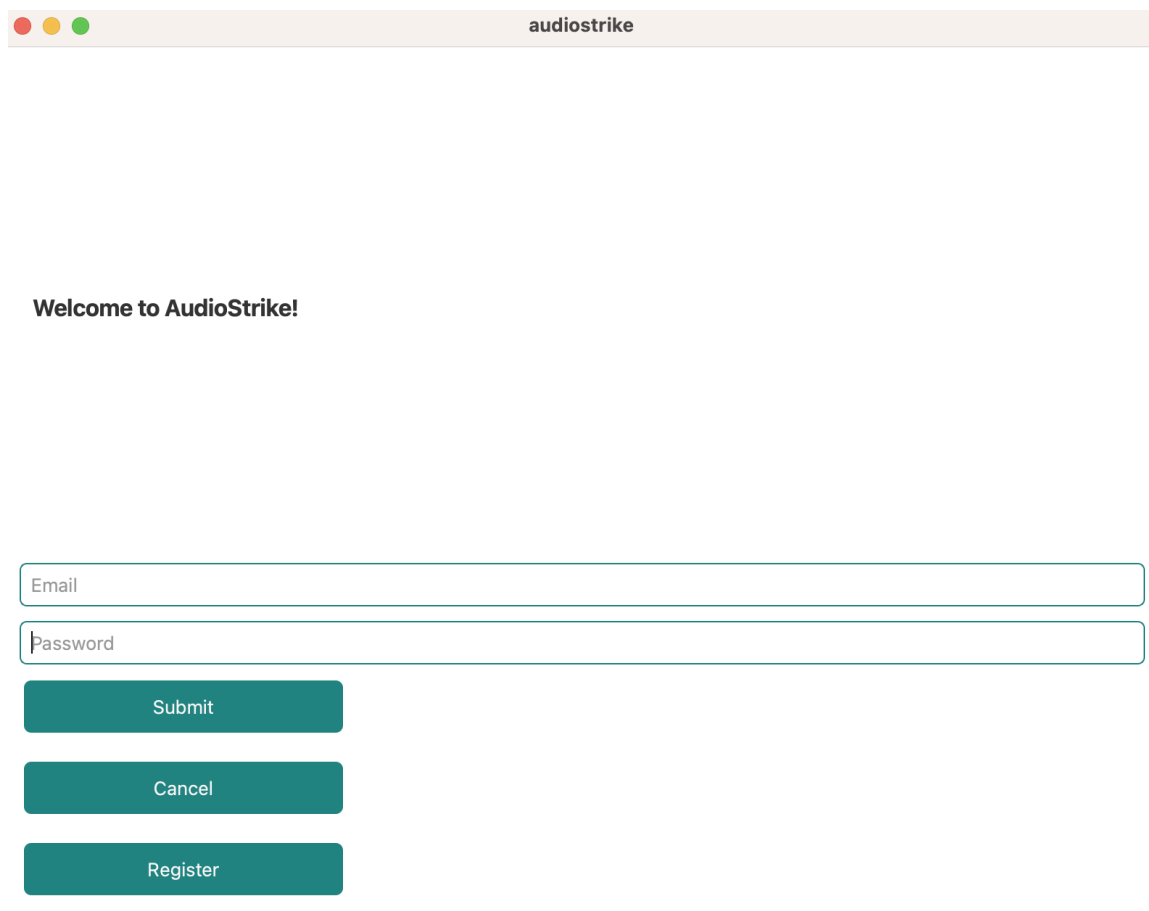
### 3.2.1 User Interface

The user interface (UI) for this application was designed to be as simplistic as possible. After downloading and opening the application, the user is prompted to either register for an account or log in to an existing account (shown in Figure 3.2). In the registration view, the user must also sign a consent form (described in chapter 3.3) to participate in the study. All data collected using this tool is assumed to be using a standard QWERTY keyboard.

To register, the participant must accept the study consent form (shown in Figure 3.3). Following acceptance, the application records basic demographic and contact information for each participant (shown in figure 3.4).

Once a user is successfully authenticated, they can access the collector configuration page (shown in Figure 3.5). On this page, the user can configure the collector with their local hardware. This includes I/O devices like a microphone and keyboard.

Once the user starts the experiment they are shown the final window (shown in Figure 3.6) which has three buttons: play, pause, and stop. By default, the application is in a running state but the user is able to pause a collection job and restart it or stop the experiment entirely and exit the application.



audiostrike

**Welcome to AudioStrike!**

Email

Password

Submit

Cancel

Register

Figure 3.2: Login Page

### 3.2.2 Data Collector

The main purpose of the data collector is to efficiently gather keystroke data in a secure and scalable manner. For each keystroke, the collector yields a data structure containing the key pressed and a spectrogram (an image representing the audio signal). These artifacts are then committed to the database. All of the processing and conversion occurs on the user's local machine before being committed to the cloud in order to preserve anonymity and prevent any reconstruction attacks from occurring on a user's data. For the same reason, time stamps are also stripped from the keystroke metadata before being committed to the server.

The spectrogram and keystroke objects are processed using a sequence of sub-

audiostrike

Introduction and Study Overview

Thank you for your interest in our behavioral science research study. We would like to tell you what you need to think about before you choose whether to join the study. It is your choice. If you choose to join, you can change your mind later and leave the study.

The purpose of this study is collect diverse typing data in order to train a robust deep learning model to aid in secure keystroke attestation on root compromised systems. The study is funded by Emory University Computer Science Department. This study will take about 1-3 months to complete, requiring no more than your normal time spent on daily web browsing.

If you join, you will be asked to install our "audiostrike" desktop client and operate your computer as normal. In the client you will be able to start and stop the data collection process at any time.

Storing and Sharing your Information

We will store all the data that you provide using a desktop-based client. The client performs all audio and keystroke processing on the participant's machine and ensure that no raw audio data is being transmitted to the study staff as well as strips timestamps so sequences of keystrokes cannot be reconstructed. We need this code so that we can keep track of your data over time. This code will NOT include information that can identify you (identifiers) aside from a registry of participants and hardware metadata. We will keep a file that links this code to your identifiers in a secure location separate from the data.

We will NOT allow your name or any other fact that might point to you to appear when we present or publish the results of this study.

I consent to participating in th

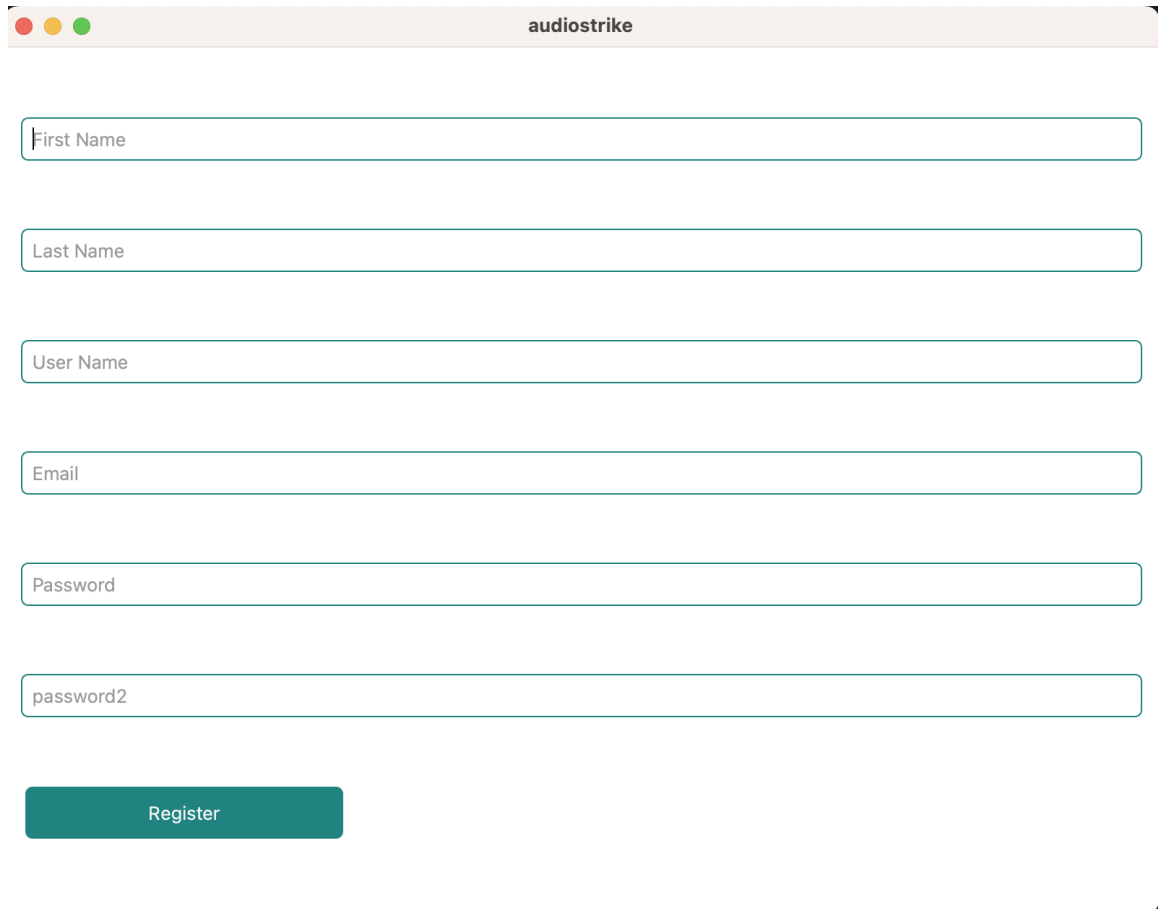
Accept

Reject

Figure 3.3: Consent Page

processes running in parallel. The processes communicate with each other as needed using thread-safe queues that sequentially transition ownership of the keystroke event object. Each sub-process has two components, a listener and a sender. The listener receives data from another sub-process, while the sender transmits a processed component from the received data to another sub-process for further analysis until eventually it is submitted to the back-end. The processes are submitted in the following progression. First, each keystroke is logged and serialized into a data model. Next, it is submitted to the audio processing queue. Lastly, before submission, it is sent to the spectrogram conversion queue.

1. Keylogger Sub-process



The image shows a web browser window titled "audiostrike". The page contains a registration form with the following elements:

- A text input field labeled "First Name".
- A text input field labeled "Last Name".
- A text input field labeled "User Name".
- A text input field labeled "Email".
- A text input field labeled "Password".
- A text input field labeled "password2".
- A teal-colored button labeled "Register".

Figure 3.4: Registration Page

The keylogger sub-process records all active keystrokes to the system and serializes them into a data model that is propagated through the rest of the sub-processes. For each keystroke, an event is recorded that captures the key pressed along with the time it was pressed and released. On release, a callback function is triggered to send the keystroke to the audio processing subprocess. This process also derives a feature for each keystroke's cleanliness. We define a clean keystroke as one which is pressed in isolation. For example, consider two keystrokes  $k_1$  and  $k_2$ .  $k_1$  was pressed at time  $t_1$  and released at time  $t_2$  while  $k_2$  was pressed at time  $t_3$  and released at time  $t_4$ . If  $t_1 \leq t_2 \leq t_3 \leq t_4$  then the keystroke is clean. If any of these time stamps overlap, the key is labeled as not

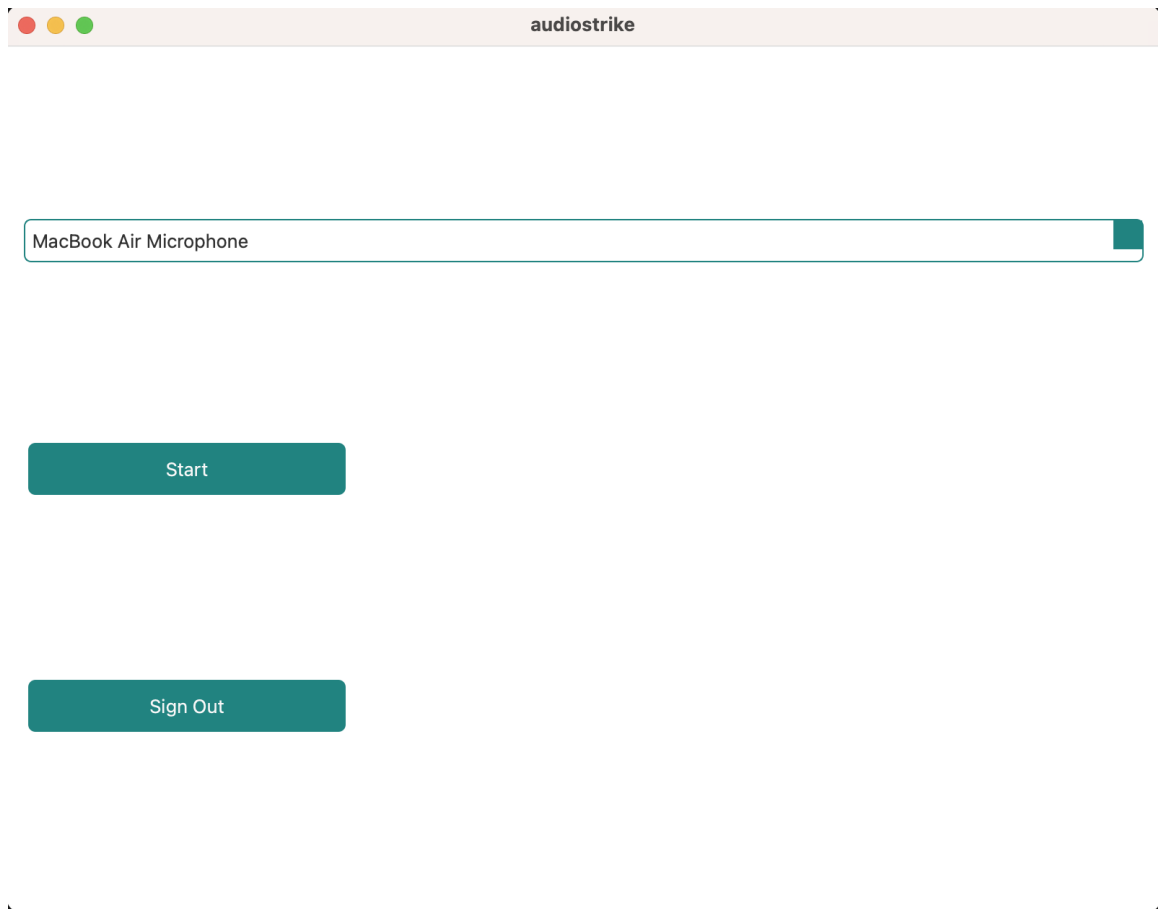


Figure 3.5: Configuration Window

clean and skipped in processing.

---

**Algorithm 1:** Keylogger Sub-process

---

**Input:** Active keystrokes from operating system

**Output:** Serialized keystroke data model

```

1 foreach keystroke do
2   Record keystroke events including time-pressed and time released; if
   keystroke is clean then
3     Derive feature for keystroke cleanliness; Callback audio processing
     subprocess with a keystroke;
4   end
5 end

```

---

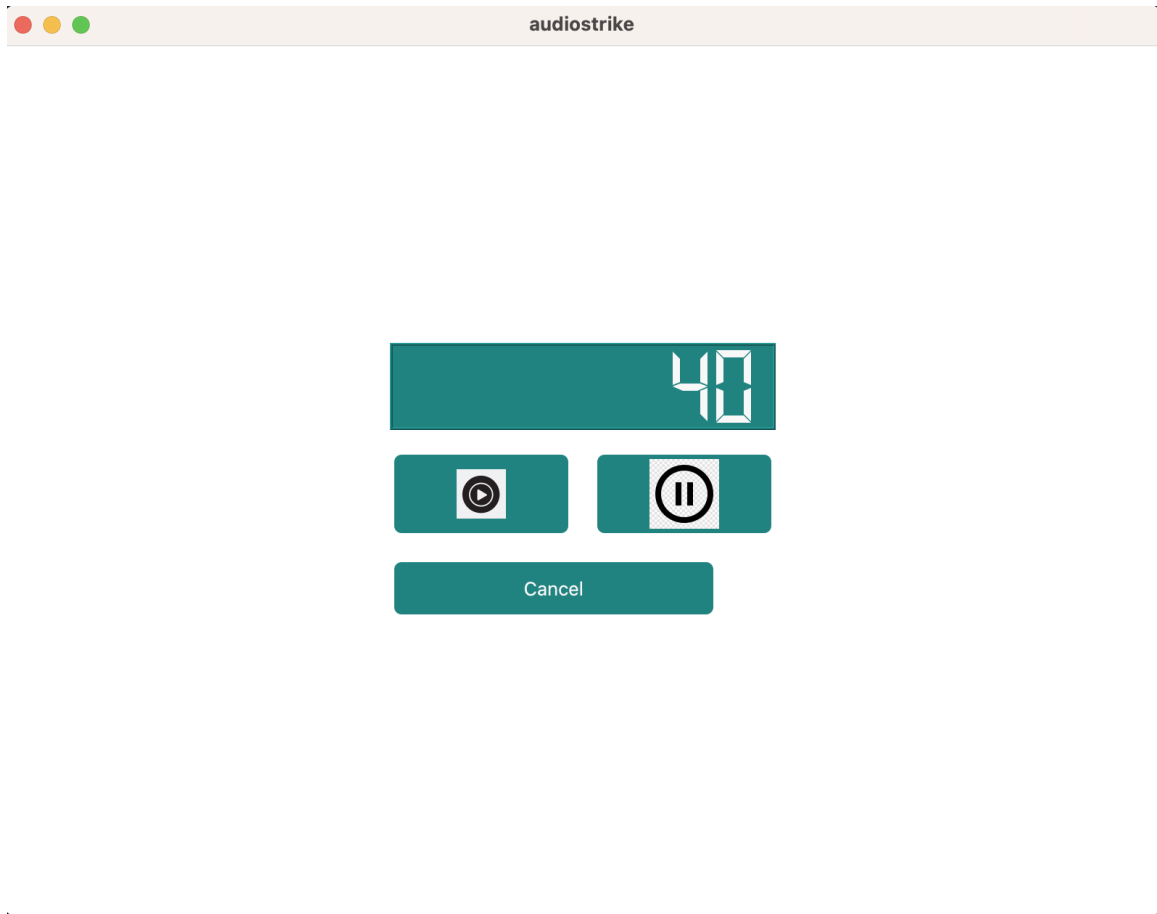


Figure 3.6: Experiment Window

## 2. Recorder Sub-process

The recorder sub-process maintains a ring buffer of audio frames that are streamed in real-time directly from the user's internal or external microphone selected during the experiment configuration process. The buffer maintains 2-3 seconds of audio that is continuously being overwritten by default but has the functionality to dynamically resize itself depending on the specifications of the user's machine and latency between the audio recording and keystroke logging. The ring buffer receives requests from the audio requester sub-process to send data between two given time points relative to when a key was pressed and released.

---

**Algorithm 2:** Recorder Sub-process
 

---

**Input:** Real-time audio frames from microphone

**Output:** Ring buffer of audio data

```

1 Initialize ring buffer with 2-3 seconds of audio data; while recording do
2   | Stream incoming audio frames to ring buffer;
3 end
4 foreach keystroke do
5   | Retrieve audio data for given frame numbers; Send audio data to the
   | spectrogram conversion queue;
6 end

```

---

### 3. Collector Sub-process

The collector sub-process manages both the recorder and key logger sub-processes. Its purpose is to provide pause and play functionality to a participant in the data collection process. Despite the safeguards in place to prevent data reconstruction attacks, if a user is typing sensitive information such as a password or credit card number, they may not want that data recorded. The collector sub-process can stop and start the keylogger and audio threads while maintaining a cache of objects that are actively being processed in the pipeline. As such, it enables the application to still commit previously typed data while safeguarding a user's privacy.

---

**Algorithm 3:** Collector Sub-process
 

---

**Input:** Active keystrokes and real-time audio data

**Output:** Serialized keystroke data model with cached objects

```

1 while collecting do
2   | if pause requested then
3   |   | Stop keylogger and audio threads;
4   | end
5   | if play requested then
6   |   | Start keylogger and audio threads;
7   | end
8   | Cache actively processed objects and continue processing;
9 end

```

---

#### 4. Audio Requester Sub-process

The audio requester sub-process's only job is to intercept incoming keystrokes and make requests to the recorder process that query the right amount of audio data from the active stream. After the audio data is parsed from the stream and added to the keystroke object, the audio slicer sub-process sends it to the spectrogram conversion queue.

---

**Algorithm 4:** Audio Slicer Sub-process
 

---

**Input:** Active keystrokes from keylogger sub-process

**Output:** Requests for audio data from recorder sub-process

```

1 foreach keystroke do
2   | Make a request to the recorder sub-process for audio data;
3 end

```

---

#### 5. Spectrogram Conversion Sub-process

The spectrogram sub-process receives keystroke objects with associated audio data. Before sending these objects to the submission queue, they are converted



to spectrograms in order to obfuscate a user’s real audio data and further preserve their privacy. This process applies a short Fourier transform to the audio signal, which converts the amplitude to decibels. Finally, the raw imaging data is converted in memory to PNG format and a unique random identifier is generated to use as the filename. After the spectrogram and filename are generated, they are cast into the keystroke data model. The sub-process finally sends the newly created keystroke object to the submission manager sub-process for further processing.

---

**Algorithm 5:** Spectrogram Conversion Sub-process

---

**Input:** Keystroke objects with associated audio data

**Output:** Spectrogram data and serialized keystroke data model

```

1 foreach keystroke do
2   |   Apply short Fourier transform to audio data; Convert raw imaging data
   |   to PNG format; Generate unique 64-bit UUID as filename; Add
   |   spectrogram and filename to keystroke data model; Add keystroke to
   |   active batch for submission;
3 end

```

---

## 6. Submission Sub-process

The submission subprocess’s job is to submit fully processed events to the server. Every event that occurs on the client’s machine, including logging in, registering a new user, starting a collection job, logging microphone and keyboard meta-data, and keystrokes are all submitted to this queue. In addition to submitting processed events, this sub-process also maintains a special data structure that is used to cache keystrokes before submitting them to the server. This data ensures that keystroke events are stripped of timestamps and shuffled so that the order of keys pressed cannot be reconstructed.

---

**Algorithm 6:** Submission Sub-process
 

---

**Input:** Fully processed keystroke data model with spectrograms

**Output:** Submitted events to server

```

1 Initialize keystroke cache data structure; while collecting do
2   |   Add keystrokes to cache; if cache is full or session is ending then
3     |   |   Shuffle keystrokes in the cache; Strip timestamps from keystrokes;
4     |   |   Submit keystrokes to the server; Clear cache;
5   |   end
6 end

```

---

### 3.2.3 Back-end

The back end of the application functions as a receiver for data processed and sent directly from the participant’s machine. There are two components used to store participant data: a PostgreSQL database to store tabular data and a cloud storage bucket to store the spectrograms for each job. Both cloud artifacts implement four data models that are described in more detail below. Some of these data models are used for experimental data collection while others are used to keep track of user information and authentication tokens.

#### 1. Keyboard

The keyboard data model stores metadata about the keyboard used by a participant in the collection job. Every time a keystroke is recorded by the application, it returns an identifier for the hardware used to trigger that event. This model assumes that only one keyboard is used for each collection job.

#### 2. Microphone

The microphone data model stores hardware metadata about the microphone used to record the audio signal created when a participant triggers a keyboard event. This model includes the microphone’s sample rate, input channels, and

low and high input and output latencies.

### 3. Keystroke

The keystroke data model stores the actual data that will be used in training. Each sample contains the physical key logged as well as a reference to the processed spectrogram that was committed to the server. The spectrogram is stored in a cloud storage bucket where the key to the object is the value stored in the relational database. The keystrokes are also tagged with a collection job identifier to group all keys from the same job together.

### 4. Collection Job

The collection job data model stores information about each individual job, including the date and time a job was started, a reference to the microphone and keyboard objects used in the job, and a reference to the user that started the job. A collection job, keyboard, and microphone object must be committed before any keystrokes are able to be logged.

## 3.3 IRB Study Design For Crowd Sourcing

Emory IRB **STUDY00005668** (undergoing review) details the study design for the AudioStrike data collection experiment. The study design has two components: a lab-based component and an at-home-based component. The collection of user-identifiable information is minimal, and no demographic or personally identifiable information will be collected to expedite the IRB approval process. However, a signed consent form is still required when registering with the data collection tool. Both components of the study use the same binaries and methods described above; the difference lies in the experiment setting.

The lab-based component will occur in a highly controlled environment, such as a computer lab or a psychology building, with specifics to be clarified in the

manuscript. Participants would be provided a computer with a uniform keyboard, mouse, and microphone. They would also be isolated from each other to minimize potential background noise and interruptions. Participants would be instructed to operate the computer as normal, following a generic prompt to ensure a consistent stream of keystrokes. Typing data would be collected for each participant for thirty-minute sessions using the data collection tool described in Chapter 3.2. The goal is to have 30-50 participants in the lab setting.

The at-home component could occur anywhere. Participants are encouraged to perform the data collection in a room without any background noise, and the data should be collected across multiple days in 30-minute increments. The user is expected to operate their computer as normal with the collector running in the background, performing natural tasks such as coding and writing. Because of the variable setting, many confounding variables are introduced, including non-uniform I/O devices like keyboards, mice, and microphones. Since each keyboard, mouse, and microphone combination are different, it is possible that the signals for a keystroke would not be consistent between hardware configurations. To alleviate this, the data collection tool also records hardware metadata. In doing so, during the proposed analysis, we can compare the data taken from specific hardware or normalize certain values based on hardware metadata. For example, if two microphones have different sample rates, it would be important to adjust the spectrogram representation of the audio signals to account for this difference. The aim is to have at least 100-200 participants in the at-home setting.

In addition, it is likely that the at-home data collection environment would be more prone to background noise and unexpected audio interruption depending on the physical environment where the user enables the data collection tool. For example, if a user decided to run the experiment in a highly crowded area, or was on a phone call at the time a keystroke event occurred, the spectrogram associated with a captured

keystroke would incorporate the unintended perturbation. While this could introduce spurious features in the data set, we hope that given a large volume of data, the model would be able to filter out the noise and discover the true signals of keystroke sound.

Both of these components are essential to collect a large sample of data from as many participants as possible while also ensuring the quality and integrity of the collected data. While the volume of the lab-based component is not anticipated to be as large as the volume of data collected in the at-home component, it serves a unique purpose: a baseline data set. The baseline data from the highly controlled environment can now be compared with the crowdsourced data to ensure quality and integrity. In addition, the data could now be stratified by keyboard or microphone type, presence of background noise, and other confounding variables to see which has the most effect on performance. The collection tool also collects metadata about the keyboard and microphone used, allowing for further contextualization of the model output.

### 3.4 Model Training

In this study, we used a basic Convolutional Neural Network (CNN) architecture with five convolutional blocks for multi-class classification. CNNs (described in more detail in Chapter 5.3) have been shown to perform particularly well with imaging data so we chose this architecture since we are using the image representation of the audio signal emanating from a keystroke. The model was trained using a categorical cross-entropy loss function and optimized using the Adam optimizer with a batch size of 32. We also implemented a learning rate scheduler to adjust the learning rate during training for better convergence. The model was trained on a dataset consisting of three keyboard region bins and the performance was evaluated using various metrics such as accuracy, precision, and recall. Furthermore, the CNN model was specifically

designed for keyboard region classification between the left, right, and space bar as shown in Figure 3.8.

This task required the model to accurately classify the position of keystrokes based on the corresponding keyboard region. After the data was collected, we split the full cohort into three separate sets: train, validation, and test. The train and validation sets were used during model training to assess the performance of the model after each epoch while the test set was removed from the training process so its final performance could be evaluated on unseen data.

### 3.4.1 Metrics

By utilizing the proposed methodology, we aim to maximize the following metrics on the test set: accuracy, F1 score, precision, recall, and the area under the receiver operating characteristic (ROCAUC). Accuracy measures the proportion of correct predictions made by the model over the total number of predictions. Precision measures the accuracy of positive predictions, recall measures the ability of the model to identify positive instances, and the F1 score balances both precision and recall to provide a single score that reflects the model's overall performance. The area under the receiver operating characteristic (ROCAUC) curve is a measure of the model's ability to distinguish between positive and negative instances (true positive rate versus false positive rate), where a higher value indicates better performance.

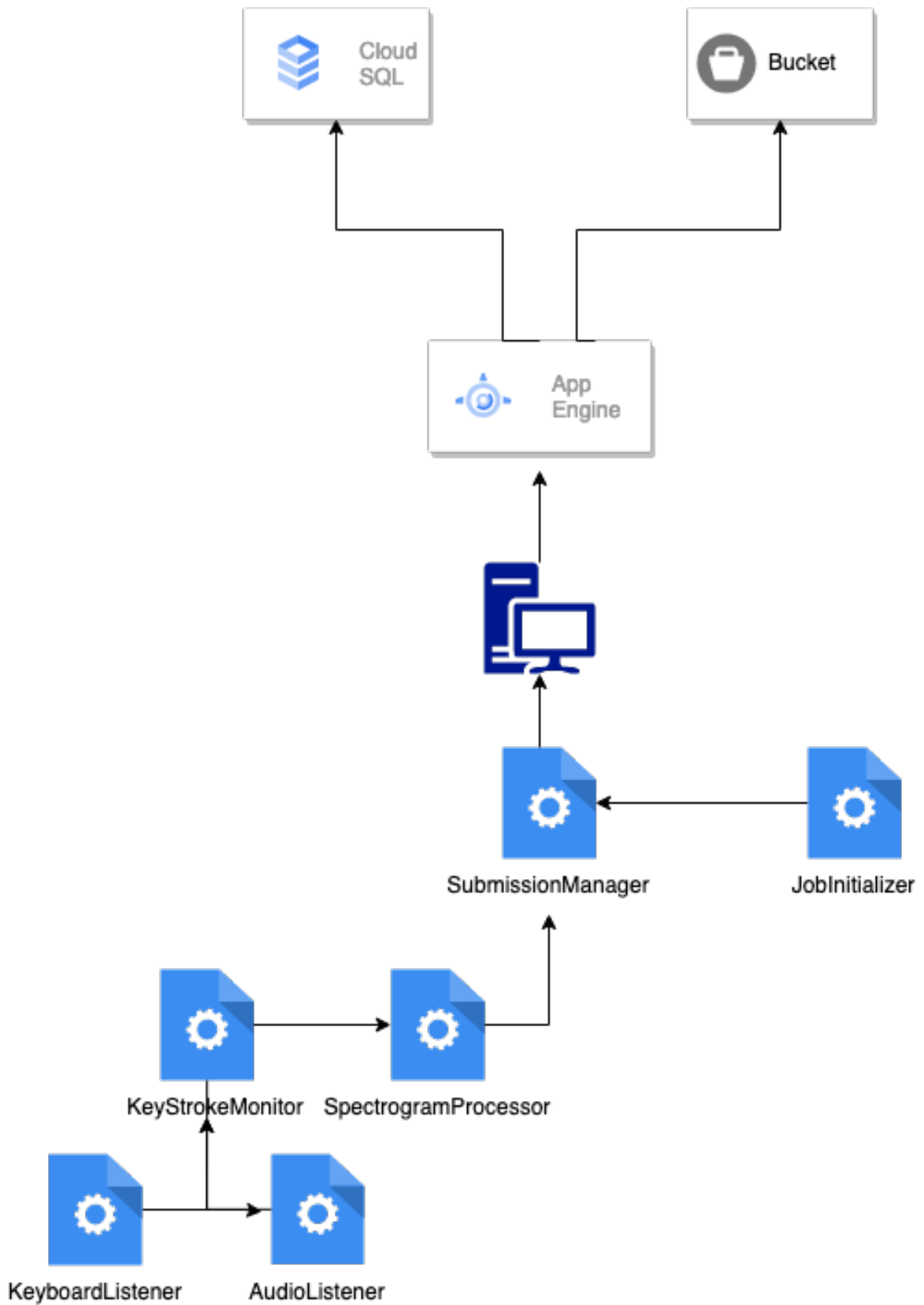


Figure 3.7: AudioStrike Collector Architecture

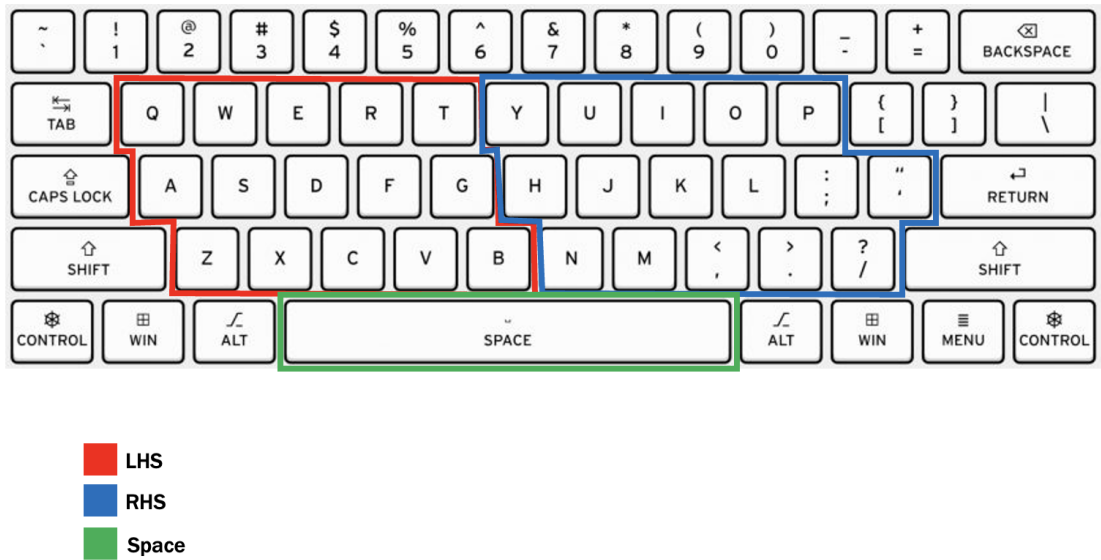


Figure 3.8: Keyboard Split



# Chapter 4

## Results

### 4.1 Data Exploration

Using the underlying algorithms from the data collection tool defined in Chapter 3.2, keystroke and spectrogram data were collected for a single typist over the course of one week. The typist used the same keyboard and microphone throughout the duration of the collection process. During the collection process, keystrokes were classified as either pure or impure (as mentioned in Chapter 3). Over the period of data collection, 30,548 unique keystrokes were registered and converted into spectrograms using the collection algorithms described in the previous chapter.

Table 4.1: Keystroke Distribution

	Grouped by Purity		
	Overall	Impure	Pure
Frequency	30548	11673	18875
Character, Frequency (%)	!	3 (0.0)	3 (0.0)
	”	7 (0.0)	7 (0.1)
	’	53 (0.2)	13 (0.1) 40 (0.2)
	#	16 (0.1)	16 (0.1)
	\$	2 (0.0)	2 (0.0)

As can be seen in Table A.1 (full table can be seen in appendix A), the fre-

quency of each key recorded varies greatly. This is due to the increased prevalence of certain characters in the English language over others [18]. Due to the variability in the frequency of each keystroke, the possible classes were binned into three categories: left-hand side (LHS), right-hand side (RHS), and space bar (as shown in Figure 3.8). The space bar was the only key chosen as a unique category since it is one of the most common and acoustically distinct keys. Algorithm 7 describes how each key was split from a QWERTY keyboard. In this case  $L = [\text{qwertasdfgzxcvb}]$  and  $R = [\text{yuiophjklmn,./;}]$ . It's important to note that character case was not considered during the assignment of keystrokes because pressing a capital letter or special character often requires the use of the shift button or other modifiers. The use of a shift key frequently marked capital and special character keystrokes as impure and therefore discarded before region assignment. In addition, characters that were not included in the left and right list or space categorization class were discarded from region assignment since their location often varies between keyboards.

---

**Algorithm 7:** Categorize Keys as LHS or RHS

---

**Input:** A key  $k$  and two lists,  $L$  and  $R$   
**Output:** The category of the key, either "left", "right", or "space"

```

1 if  $k == \text{"space"}$  then
2   | category  $\leftarrow$  "space";
3 end
4 else if  $k \in L$  then
5   | category  $\leftarrow$  "left";
6 end
7 else if  $k \in R$  then
8   | category  $\leftarrow$  "right";
9 end
10 else
11   | return
12 end
13 return category;

```

---

After the keys were binned into their respective categories and filtered for only pure keystrokes, we were left with the following distribution of keys in their respective

bins.

Region	Frequency
LHS	7251
RHS	5091
S	3226
Total	15568

Table 4.2: Keystroke Distribution After Binning

Using this distribution of keys, majority class undersampling was performed to account for the large class imbalance in the keyboard regions. Examples of randomly selected spectrograms from each region are shown below. It can clearly be seen that the space region has the most distinct differences from the other two regions. Examining these figures demonstrates a clear difference between the space spectrogram and the left and right spectrograms.

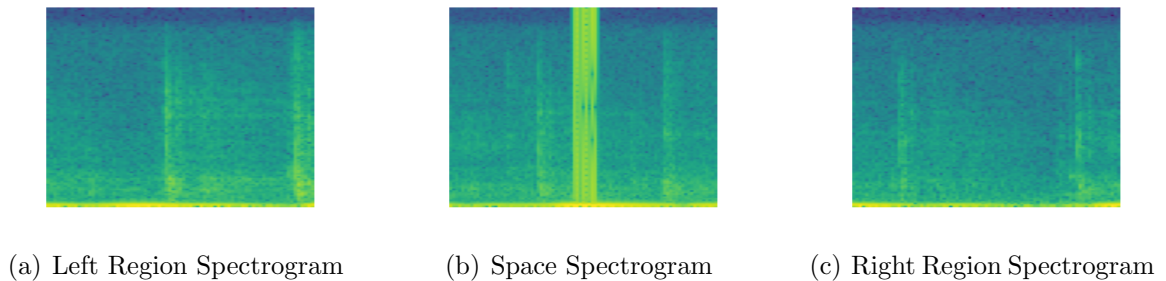


Figure 4.1: Example Spectrograms

## 4.2 Model Evaluation

The model described in chapter 3.2 was trained on the data set described in chapter 4.1. In this section, the model’s performance on the data set is explored. First, it is important to consider the training and validation loss and accuracy plots to ensure the model did not over-fit or under-fit on the training data and that each metric converged properly. Note that although accuracy is reported in the plots, the optimal threshold which optimizes precision and recall has not been determined.

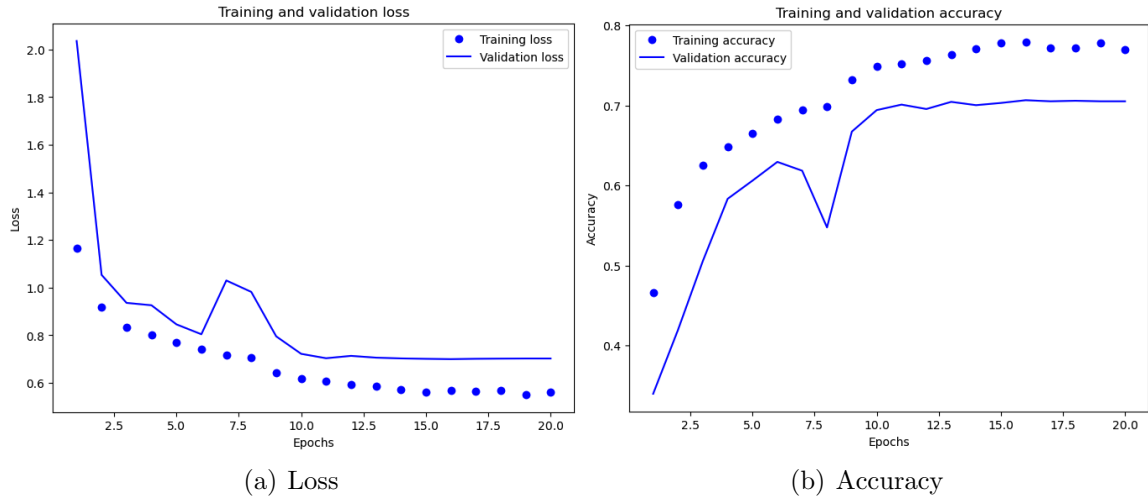


Figure 4.2: Training and Validation Convergence

From Figure 4.2 it can be seen that both the accuracy and the model loss during training and validation converge and only make marginal improvement after around ten epochs.

While the above analysis shows that the model did not overfit or underfit, it does not verify the performance on an unseen test set. As described in Chapter 3.4, a portion of the preprocessed data set was held out for evaluation after the model training process was complete. The model achieved the following performance when it was used to run inference on the holdout set. Note that for metrics that require a threshold like an accuracy, precision, recall, and f1-score, the optimal threshold was chosen to maximize the F1-score.

Table 4.3: Classification Metrics

Class	Precision	Recall	F1-score	Support
L	0.71	0.65	0.68	494
R	0.65	0.72	0.68	466
S	0.76	0.74	0.75	492
Accuracy	0.70			
Macro Avg	0.70	0.70	0.70	1452
Weighted Avg	0.71	0.70	0.70	1452

Thresholdless metrics were also calculated as an alternate way of evaluating model

performance. Specifically, the ROC curve and corresponding ROCAUC were examined. The ROC curve displays the relationship between the false positive rate (FPR) and true positive rate (TPR) at different thresholds for the model. A model that was trained on completely random data would have a ROCAUC score of 0.5 while a perfect model would have a ROCAUC score of 1.0. Another notable component of this metric is that it can only be used to evaluate a binary classification. In a multi-class classification model, the one vs. all approach is commonly used. The target class is treated as correctly classified while all other classes are considered to be incorrectly classified.

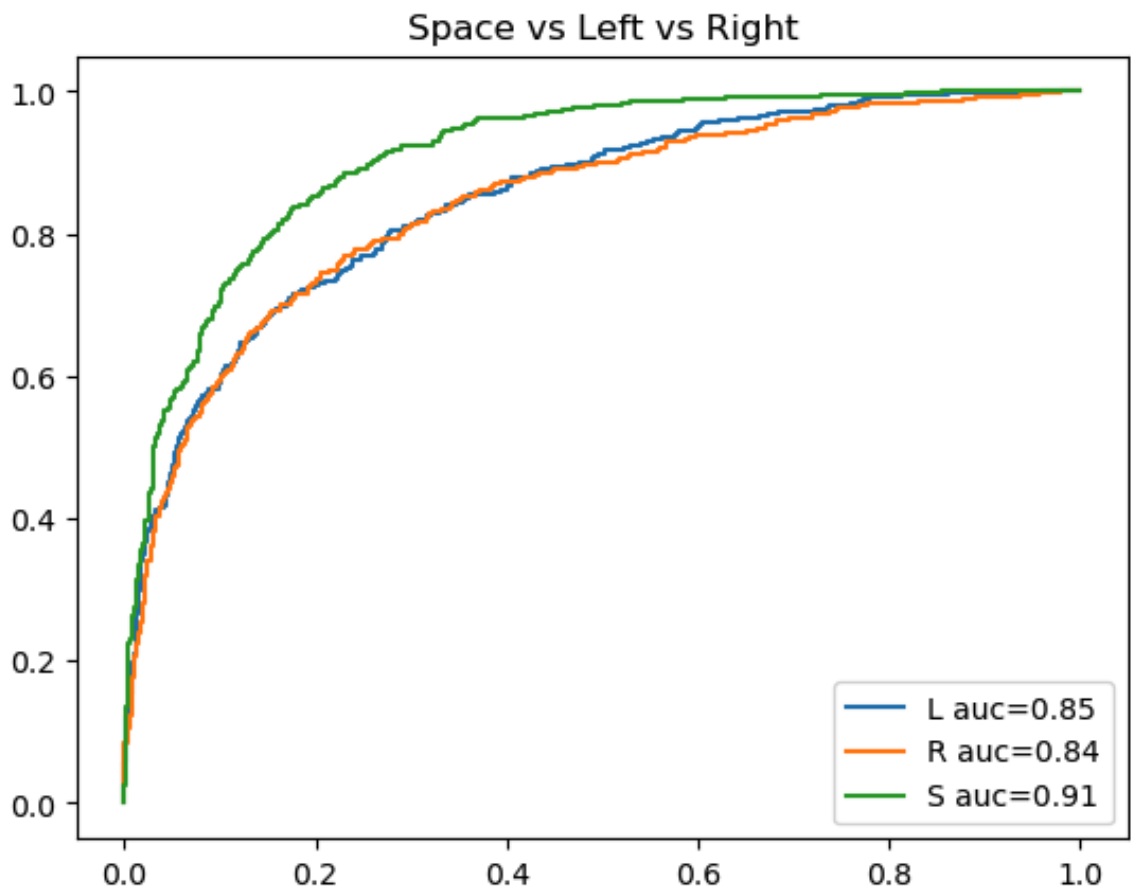


Figure 4.3: ROC Curve

Figure 4.3 shows the ROC curve for the model described above. The average

ROCAUC of all three classes is 0.87, demonstrating that the model can classify a keyboard region to a significant degree.

# Chapter 5

## Related Works

### 5.1 Security Considerations

Confidentiality, integrity, and availability, also known as the CIA triad is a term commonly found throughout computer security literature [27]. Confidentiality refers to the unauthorized access of information or use of an application or service on a computer system. Integrity refers to the unauthorized modification of data on a computer system. Availability refers to the unauthorized denial of service for some data, service, or application on a computer system. All three traditional definitions of these terms center around a malicious actor performing unauthorized operations on their target's machine or with their target's data. The definitions of these three terms have grown since their inception in the 1970s but the idea of each term remains more or less the same. Over time, Samonas and Cross have shown how these definitions have been subsumed in the academic literature as the concept of virtual identity [27]. The concept itself is especially relevant when considering various threat models and authentication techniques since underlying every method of authentication is the idea that if successful, some external services verify your identity. However, in a case where a target's virtual identity is compromised and an attacker authenticates as

their target, the principles of the CIA triad implode. Any defenses the user or service has becomes inadequate since for all intents and purposes, the attacker is now the target from the scope of the target's virtual identity.

The evolution of the CIA triad is important to consider in the context of this study the philosophical question security experts are trying to answer is what constitutes virtual identity. In the CIA model, if a malicious actor is able to authenticate as their target, they are considered to be one and the same. However, this is not truly the case.

## 5.2 Authentication

As mentioned in Chapter 1, authentication is often colloquially defined as having one or more of the following three things: something you know, something you have, and something you are. Something you know can be thought of as a password or answer to a security question. Something you have can be thought of as a physical key, authentication token, or one-time verification code unique to the user. Something you are refers to biometrics e.g. a fingerprint or retinal scan. While these attributes of authentication provide a first pass at securing a service or machine, a single authentication attribute in isolation can be trivial in actually stopping an attacker. Consequently, the trend of using multiple authentication attributes has become increasingly common. Examples of multiple authentication techniques include two-factor authentication (2FA) and multi-factor authentication (MFA). 2FA [4] is defined as having two of the key elements of authentication while MFA [14] is defined as having two or more of these attributes. Many varieties of these concepts exist in authentication literature, varying in terms of the number of attributes used and the modality of attributes being authenticated, as well as the frequency in which authentication is performed.



One method of authentication frequency that has been explored throughout multiple works is continuous authentication (CA). CA is a process that constantly evaluates the identity of a user on a system or service and can take a variety of forms depending on which authentication attributes a system or system administrator wants to leverage. [11]. For example, Deutschmann et al. propose an authentication system that leverages user interaction, such as keystrokes, mouse movement, and activity, on a machine in order to evaluate the "trust" a system ought to have in a particular user [8]. Additionally, they developed a model that can evaluate trust and differentiate between users based on behavioral biometrics, thereby making it even more difficult for attackers to bypass authentication checks.

Beyond the scale of more traditional systems, CA has also been applied to the internet of things devices (IOT), specifically wearable devices like smartwatches and fitness trackers. If a wearable is assumed to be secure in an authentication system's threat model and can communicate with the machine over some kind of secure channel, it can be used as an external monitor for continuous authentication [3]. Biometrics and IoT have even been combined to provide an even more secure level of security. For instance, Zhao et al. used cardiac biometrics obtained by a wearable as a method of enhanced security as opposed to evaluating a wearable (something you have) and a biometric (something you are) in isolation [39].

### 5.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) were first proposed by Lecun et al. [17] in a paper applying machine learning to the classification of images of ZIP codes on postage artifacts. CNNs are a variation of the traditional artificial neural network (ANN) that can extract sequential encodings from the input data using a variety of linear and non-linear transformations and matrix operations. When the idea of CNNs

was first proposed, the hardware needed to train a model to be able to capture complex spatial encoding was not cost-effective. However, with the increasing availability of graphical processing units (GPUs), extracting complex spatial or sequential encoding from images has become a relatively straightforward task.

There are three main components to a CNN: convolution, pooling, and fully connected dense layers [26].

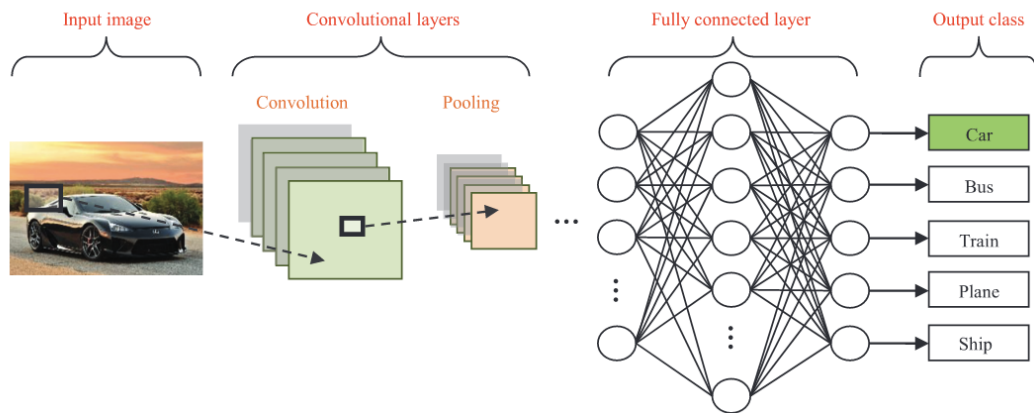


Figure 5.1: CNN Architecture [26]

The convolutional layer performs element-wise multiplication on the sequential input data with a kernel or mask to extract its feature representation. During the training process, the weights used to extract the features from the input are tuned in order to extract the most representative features for a given machine-learning task. In the process of convolution, the dimensionality of the input data is reduced as the kernel is normally much smaller in size than the input data, which allows the kernel to extract important spatial features. Pooling layers are often applied directly after a convolutional layer. Pooling layers further reduce the dimensionality of the input data by using an optimization function like the max or average to try and select spatially invariant features from the input data. By selecting an optimized feature from each space, the model can further extract important features. The last component of a

CNN is one or more fully connected layers. These layers function similarly to those in a vanilla ANN, using the input features extracted from the previous sequence of convolutional and pooling layers to perform a given machine-learning task.

The basic CNN architecture described above has been altered and adapted to different modular components in order to optimize feature extraction and improve the network's performance on a variety of classification tasks. Examples of the current state-of-the-art CNN architectures include residual neural networks (Resnet) [12], which are designed to leverage skip connections to avoid weight decay, and densely connected CNNs (Densenet) [13] which explicitly connect all layers in the model to one another in order to maintain a more robust feature representation and avoid losing intermediate features throughout the training process.

## 5.4 Audio Signal Processing

Audio signal processing has been a widely researched field with numerous studies and applications in various domains. In particular, this section will review different ways of preprocessing raw audio frames for use in AudioStrike's underlying CNN. One of the most common ways to process audio for use in state-of-the-art machine learning models is an audio spectrogram [25]. An audio spectrogram is a visual representation of a sequence of audio frames. In its most basic form, it depicts a signal's frequency vertical over a given time horizontally as well as the amplitude of the signal at the different time points (color). There are many variations of the simple spectrogram described above that have been shown to better extract auditory features. One such type of spectrogram is called a mel-spectrogram. Mel-spectrograms use the Mel Scale to transform the vanilla spectrogram described above [29]. The Mel Scale converts the raw audio data into a spectrogram such that all pitch differences would sound equally different to a listener. This is because humans have been shown to be able to

better perceive auditory differences at low frequencies than at high frequencies.

Another type of spectrogram that has been extensively studied is the Constant Q-Transform (CQT) spectrogram [38]. The CQT is a type of time-frequency analysis that uses a logarithmic frequency scale to provide a constant ratio of frequency resolution across all frequencies. This makes it well-suited for analyzing signals with non-stationary or time-varying spectral characteristics, such as musical notes with varying harmonics. The CQT works by decomposing a signal into a set of overlapping frequency bands that are centered on logarithmically spaced frequencies. The width of each frequency band is determined by a Q factor, which controls the frequency resolution of the transform. Higher Q factors result in higher frequency resolution at lower frequencies, while lower Q factors result in higher frequency resolution at higher frequencies. The CQT is similar to the vanilla Short-Time Fourier Transform (STFT) based spectrogram in that it divides a signal into overlapping windows and computes a Fourier transform for each window. However, unlike the STFT, which uses a fixed frequency resolution across all frequencies, the CQT uses a variable frequency resolution that is tailored to the specific spectral characteristics of the signal. One advantage of the CQT is that it provides a more accurate representation of the harmonic structure of a signal than the STFT, making it especially useful for analyzing musical signals with complex harmonics. Another advantage is that it allows for efficient computation of the spectrogram using fast Fourier transforms (FFTs), improving computational efficiency.

## 5.5 Crowd Sourcing

In order to build a generalizable model for keystroke detection, it is important to gather a diverse dataset of typing data on a variety of hardware systems. Approaching this task in an extremely controlled, live environment would be logistically challenging

and expensive. An alternative to strictly lab-controlled data collection is crowdsourcing. Findlater et al. [9] compare the differences in data collected in a crowdsourced versus a lab environment on different modalities of hardware. They find statistically significant differences in the quality of the data and the performance of participants on a performance task in a lab and in crowdsourcing settings.

It is also important to examine the literature around potential deficiencies when using large amounts of both manually and automatedly curated data to make informed decisions, especially with something as sensitive as computer security. Becker et al. from the MITRE corporation examined the potential faults of large amounts of data collected for different use cases across four different industries to complete a diverse set of tasks [6]. Overall, the authors reported ten main findings, the majority of which involved data quality. The first and arguably most important result that they mention is that the larger the volume of data, the more the quality decreases. This is important to note in the context of AudioStrike as the volume of data collected in the at-home component of the proposed study would most likely be many factors greater than the volume of data collected in the lab portion. Therefore, it is essential to be conscious of the data quality when evaluating the experimental results of the system.

Another interesting application of crowd-sourcing is the idea of gamification. Gamification is the use of game elements and mechanics in non-game contexts to increase engagement and motivation for the user. Overall, it is an effective method of keeping users entertained while still prompting them to achieve the goal of the study. In 2008, Luis Von Ahn proposed the concept of a *Game With A Purpose* (GWAP) [34]. GWAPS are games *...in which people, as a side effect of playing, perform tasks computers are unable to perform*. He then goes on to describe a general framework for effectively creating these types of applications and the abstract implementation guidelines that would make these games the most effective. He also proposes several

metrics to evaluate a GWAPs effectiveness such as throughput, defined as the average number of problems solved per hour, ALP, defined as the average overall amount of time played, and finally expected contribution, defined as the throughput multiplied by the ALP. We hope to include some of the gamification concepts described above in the AudioStrike data collection tool. They would encourage users to maintain active engagement while also continuing to gather typing data to train a more robust model.

Von Ahn et al. also proposed the idea of a reCAPTCHA. [36]. reCAPTCHA is a variation on a traditional CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) [35] but gamified to gather training data for optical character recognition (OCR) machine learning models. This study is particularly interesting because it leveraged a method that was already widely implemented in many web applications, but had been overlooked in terms of data collection. They were able to implement a system that did not change user default user interaction but was nonetheless able to collect data and utilize the potential of a preexisting system.

# Chapter 6

## Discussion

### 6.1 System Validation

After a model has been trained and achieves sufficient and robust performance in terms of classifying the region of the keyboard per user keystroke, it can then be used in the AudioStrike keystroke validation pipeline to monitor and alert a user if a potential attack is occurring. While the method of attack identification is mentioned in the architecture discussion of the AudioStrike validation system in Chapter 3.4, it does not mention how an attack is identified. To explain the attack identification process, first consider two vantage points: that of the attacker, and that of AudioStrike. Unless the attacker is physically monitoring the user typing, they would have no intuition regarding the distribution of keys typed by a user using AudioStrike aside from the natural distribution of keys of the language typed. Therefore, they would essentially have to guess the region where each key is pressed and hope for the best. Comparatively, the vantage point of AudioStrike is slightly different. Because a predictive model with accuracy  $A$  is deployed, the probability that the model guesses the correct keystroke would be the probability of the attacker random guessing among  $c$  possible keyboard regions, adjusted for the accuracy of the model  $A$ .

Attacker	AudioStrike
$\frac{1}{c}$	$(1 - A) \left(\frac{1}{c}\right) + A$

Table 6.1: Attacker vs AudioStrike Predictive Power

To prove that AudioStrike will always have a higher predictive power than an attacker, we can refer to the following inequality.

$$E_{Att} = \frac{1}{c} = (1 - A + A) \frac{1}{c} \leq (1 - A) \frac{1}{c} + A = E_{Aud} \quad \text{if } c \geq 1 \quad (6.1)$$

This shows that as long as the model can classify more than one class, no matter what the accuracy, it will always be at least as good as random guessing. Now that it is known that AudioStrike will always perform at least as well as random guessing, we can look at how much better it performs than an attacker.

Example: Consider a user running the AudioStrike deployment with the same model described in chapters 3 and 4. This model can accurately classify three regions of the keyboard  $c = 3$ . If an attacker was randomly guessing the region, they would have a  $\frac{1}{3}$  chance of guessing the correct keyboard region for every keystroke. Imagine the user is typing the string "deadbeef". Also, consider two possible events from the attacker's perspective:  $E_{keystroke}$  and  $E_{phrase}$ .  $E_{keystroke}$  represents the event that occurs when a single keystroke is pressed and  $E_{phrase}$  is the event that occurs when a phrase is pressed where  $E_{phrase} = \{E_{keystroke1}, E_{keystroke2}, \dots, E_{keystroken}\}$ .

Table 6.2 shows an example of how an attacker would need to attempt to breach AudioStrike. As exemplified in the test data, the probability that the attacker would accurately identify all keystrokes decreases exponentially relative to the number of keystrokes in a phrase. Assuming each keystroke is an independent event, after five keystrokes there is a 1.2% probability that an attacker can correctly inject keystrokes into the system and after just three more keystrokes this probability drops to 0.015%, making it almost impossible for an attacker to breach the system. With this model,



Table 6.2: Event Probability

Character	Keyboard Region	$P(E_{keystroke})$	$P(E_{phrase_i})$
d	Left	1/3	1/3 (= 1/3 <sup>1</sup> )
e	Right	1/3	1/9 (= 1/3 <sup>2</sup> )
a	Left	1/3	1/27 (= 1/3 <sup>3</sup> )
d	Left	1/3	1/81 (= 1/3 <sup>4</sup> )
b	Right	1/3	1/243 (= 1/3 <sup>5</sup> )
e	Right	1/3	1/729 (= 1/3 <sup>6</sup> )
e	Right	1/3	1/2187 (= 1/3 <sup>7</sup> )
f	Right	1/3	1/6561 (= 1/3 <sup>8</sup> )

an attack could be identified more than 99% of the time with only five keystrokes and could subsequently notify a user of a breach so they could take appropriate actions to mitigate the attack. The data above also generalizes to the equation below.

$$P(E_{phrase_n}) = \prod_{i=1}^n \theta_i \leq \varepsilon, \quad \theta = \frac{1}{c}, \quad c = \text{num classes} \quad (6.2)$$

In this equation  $n$  is the number of clean keys pressed,  $c$  is the number of classes or regions of the keyboard the model was trained to identify, and  $\theta$  is the probability of an attacker randomly guessing the correct region of the keyboard. Assuming each keystroke is an independent event, for an attacker to bypass the system, they would have to guess the region of the keyboard for every keystroke.

## 6.2 Ethical Considerations

With any large-scale data collection and AI modeling study, it is always paramount to consider the ethical implications on the participants whose data is being collected as well. Furthermore, it is imperative to also consider potential malicious uses of a model trained on keystroke data. In the case of AudioStrike, both of these issues must be addressed. In the data collection study design, we ensure data privacy and security through a number of features in the data collection tool. The first feature is batching. When any keystroke event occurs, the data is first added to a buffer of events. When

the buffer reaches capacity (default value is 1000 keystroke events), the time stamps associated with each keystroke are stripped and the events are randomly shuffled. This makes it virtually impossible to reconstruct what a participant types if a breach in the backend tool occurred. The second major safeguard added to the tool is the location of spectrogram processing. All processing occurs on the participant’s machine, which ensures no raw audio data is being collected, only an imaging representation of the audio data. While it has been shown that spectrograms can be decomposed back into their corresponding raw audio signals, the shuffling feature as well as the property that only slices of the audio data are transmitted whenever a key is struck occurs, make it highly unlikely that someone could reconstruct complete audio signals.

Another consideration is if the model weights are reconstructed or stolen. Depending on the granularity (i.e. number of keyboard regions the model can predict), it could be possible to use the model to eavesdrop rather than secure a system. While this is a large issue, there are two attributes to the design that make such an event a non-issue. First, the proposed architecture uses fine-tuned models for every specific user. As such, these models would be stored locally on an external device rather than on a server. This means that even if an attacker were able to obtain a base model, the model would not be robust enough to accurately classify every user’s keyboard region. Second, the performance of models trained to classify each key individually is poor, making it difficult to reconstruct exactly what someone is typing from knowing just three regions of the keyboard.

### **6.3 Limitations**

This study is focused on exploring the potential viability of a system that uses a side channel, specifically acoustic, to perform continuous and frictionless authentication of each user. The majority of the results of this study are based on data that has been

collected from one study participant. Because of this, there are a variety of different confounding variables that could explain both the performance of the model and why the model may not generalize well to other users.

One possible confounding variable is the homogeneous typing patterns of each individual user. Typing has been shown as a behavioral biometric, almost acting as a fingerprint for the user. It is possible that the model is learning the pattern of the data on which it was trained rather than the audio signal itself. This is not inherently a bad thing, as long as the model is fine-tuned on the user and continuously retrained to avoid data drift and account for a user's typing patterns. Another potential confounder is the geometric position of the microphone in relation to the keyboard. Since the model described in Chapters 3 and 4 was trained only on one user's data, the microphone and keyboard remained constant. However, for this system to be widely adopted, it would have to be robust enough to account for audio signals from different kinds of microphones located at varying distances from the keyboard. Furthermore, the model would also need to be robust enough to account for different keyboard types, such as external mechanical keyboards or embedded keyboards that are found in laptop computers.

While these limitations would prevent the adoption of the specific model to a diverse group of users, that is not the main contribution of this manuscript. The goal of the manuscript is to propose a system for continuous and frictionless authentication on a root compromised system and provide a framework for evaluating and training more accurate and robust models for this system.

## 6.4 Future Work

Continuous and frictionless keystroke authentication under a root-compromised threat model is still a highly complex and open question. To continue the investigation

of these problems, we hope to deploy the system mentioned in Chapter 3 of the manuscript to collect data on a diverse sample of users to create a comprehensive and representative dataset. Once the data has been collected, the next step would be to retrain the model described in Chapters 3 and 4 on the new data and evaluate the model performance, stratifying on different possible confounders acquired during the data collection process. Possible confounders include but are not limited to, microphone metadata, keyboard type, and proximity of the microphone to the keyboard. Following the updated training, the system would be prospectively validated on a new set of users to determine its efficacy. One important aspect of this system is the need for continuous retraining to ensure that it remains effective over time. This requires a secure way of collecting and processing new keystroke data from users and integrating this data into the training process without compromising user privacy or security. Given the complex nature of this issue, careful consideration of various factors such as data privacy, data security, and system performance is required. In addition to the above steps, further research could be done to investigate the potential of integrating other biometric modalities such as voice or facial recognition, which would generate an even more robust and secure authentication system.

## 6.5 Conclusion

In this manuscript, we proposed a framework for continuous and frictionless authentication of a root-compromised machine using an acoustic side channel. More specifically, we used the sound of a keystroke pressed and subsequently released. We showed that this architecture is viable after curating a typing data set consisting of keystrokes and corresponding spectrograms for a single typist and training a CNN to predict the typing region per keystroke. In addition, we also proposed a study design (pending IRB approval) and implementation of a scalable and secure data collection system to

improve the robustness of the models trained on a single typist's data as well as a method of prospectively validating the system on crowd-sourced data.

While the performance and robustness of the models described throughout this manuscript may not be able to generalize on other users since they are only trained on data from a single typist, the results suggest that with a more diverse and comprehensive dataset, a robust base model could be developed. Using transfer learning, this model could then be applied and fine-tuned to achieve optimal performance for a specific user.

On many occasions, widely used means of authentication such as 2FA and the more generalized version MFA, have been shown to be easily exploitable. Without a continuous method of authentication that can defend against root compromised systems, users will not be able to be confident that they are not under attack. Furthermore, if the system is not easy to use and requires little user interaction, users will not experience as much "security fatigue" and will be able to better protect themselves from malicious actors while maintaining their usual online routines. While one solution will never be enough to prevent any form of cyber attack, AudioStrike could provide a major step in enhancing security and defining a true virtual identity by providing a cost-effective<sup>2</sup>, continuous, and complete authentication system.

# Appendix A

## Full Keystroke Distribution

Table A.1: Keystroke Distribution

	Grouped by Purity		
	Overall	Impure	Pure
Frequency	30548	11673	18875
Character, Frequency (%)	76 (0.2)	13 (0.1)	63 (0.3)
!	3 (0.0)	3 (0.0)	
”	7 (0.0)	7 (0.1)	
”””	53 (0.2)	13 (0.1)	40 (0.2)
#	16 (0.1)	16 (0.1)	
\$	2 (0.0)	2 (0.0)	
&	2 (0.0)	2 (0.0)	
(	5 (0.0)	5 (0.0)	
)	4 (0.0)	4 (0.0)	
,	125 (0.4)	59 (0.5)	66 (0.3)
-	58 (0.2)	5 (0.0)	53 (0.3)

*Continued on next page*

---

	Grouped by Purity		
	Overall	Impure	Pure
0	24 (0.1)	9 (0.1)	15 (0.1)
1	28 (0.1)	11 (0.1)	17 (0.1)
2	20 (0.1)	5 (0.0)	15 (0.1)
3	21 (0.1)	11 (0.1)	10 (0.1)
4	32 (0.1)	21 (0.2)	11 (0.1)
5	21 (0.1)	6 (0.1)	15 (0.1)
6	6 (0.0)	1 (0.0)	5 (0.0)
7	12 (0.0)	1 (0.0)	11 (0.1)
8	14 (0.0)	2 (0.0)	12 (0.1)
9	11 (0.0)	2 (0.0)	9 (0.0)
:	12 (0.0)	12 (0.1)	
;	26 (0.1)	3 (0.0)	23 (0.1)
=	26 (0.1)	5 (0.0)	21 (0.1)
ı	1 (0.0)	1 (0.0)	
?	13 (0.0)	13 (0.1)	
@	1 (0.0)	1 (0.0)	
[	4 (0.0)	2 (0.0)	2 (0.0)
\\	5 (0.0)	1 (0.0)	4 (0.0)
a	1614 (5.3)	735 (6.3)	879 (4.7)
b	531 (1.7)	125 (1.1)	406 (2.2)
c	786 (2.6)	231 (2.0)	555 (2.9)
d	903 (3.0)	496 (4.2)	407 (2.2)
e	2131 (7.0)	1035 (8.9)	1096 (5.8)

---

*Continued on next page*

---

	Grouped by Purity		
	Overall	Impure	Pure
f	649 (2.1)	144 (1.2)	505 (2.7)
g	702 (2.3)	177 (1.5)	525 (2.8)
h	1024 (3.4)	334 (2.9)	690 (3.7)
i	1458 (4.8)	871 (7.5)	587 (3.1)
j	328 (1.1)	23 (0.2)	305 (1.6)
k	420 (1.4)	131 (1.1)	289 (1.5)
key.backspace	1805 (5.9)	216 (1.9)	1589 (8.4)
key.caps	8 (0.0)	4 (0.0)	4 (0.0)
key.cmd	101 (0.3)	35 (0.3)	66 (0.3)
key.ctrl	13 (0.0)	6 (0.1)	7 (0.0)
key.enter	792 (2.6)	61 (0.5)	731 (3.9)
key.esc	10 (0.0)	4 (0.0)	6 (0.0)
key.shift	491 (1.6)	253 (2.2)	238 (1.3)
key.space	4076 (13.3)	850 (7.3)	3226 (17.1)
key.tab	127 (0.4)	12 (0.1)	115 (0.6)
l	1050 (3.4)	435 (3.7)	615 (3.3)
m	625 (2.0)	216 (1.9)	409 (2.2)
n	1394 (4.6)	583 (5.0)	811 (4.3)
o	1574 (5.2)	1126 (9.6)	448 (2.4)
p	454 (1.5)	202 (1.7)	252 (1.3)
q	87 (0.3)	16 (0.1)	71 (0.4)
r	1077 (3.5)	577 (4.9)	500 (2.6)
s	1214 (4.0)	557 (4.8)	657 (3.5)

---

*Continued on next page*



---

	Grouped by Purity		
	Overall	Impure	Pure
t	1709 (5.6)	962 (8.2)	747 (4.0)
u	706 (2.3)	334 (2.9)	372 (2.0)
v	309 (1.0)	84 (0.7)	225 (1.2)
w	584 (1.9)	247 (2.1)	337 (1.8)
x	169 (0.6)	24 (0.2)	145 (0.8)
y	539 (1.8)	314 (2.7)	225 (1.2)
z	150 (0.5)	15 (0.1)	135 (0.7)
~	2 (0.0)	2 (0.0)	
]	3 (0.0)		3 (0.0)
‘	18 (0.1)		18 (0.1)
key.alt	6 (0.0)		6 (0.0)
key.down	20 (0.1)		20 (0.1)
key.left	118 (0.4)		118 (0.6)
key.right	34 (0.1)		34 (0.2)
key.up	109 (0.4)		109 (0.6)

---



---

# Bibliography

- [1] Internet crime complaint center(ic3): Annual reports. URL <https://www.ic3.gov/Home/AnnualReports>.
- [2] Ibm x-force threat intelligence index 2023. 2023. <https://www.ibm.com/downloads/cas/DB4GL8YM>.
- [3] Fatimah Hussain Al-Naji and Rachid Zagrouba. A survey on continuous authentication methods in internet of things environment. *Computer Communications*, 163:109–133, 2020. ISSN 0140-3664. doi: <https://doi.org/10.1016/j.comcom.2020.09.006>. URL <https://www.sciencedirect.com/science/article/pii/S0140366420319204>.
- [4] Fadi Aloul, Syed Zahidi, and Wassim El-Hajj. Two factor authentication using mobile phones. In *2009 IEEE/ACS International Conference on Computer Systems and Applications*, pages 641–644, 2009. doi: 10.1109/AICCSA.2009.5069395.
- [5] Dmitri Asonov and Rakesh Agrawal. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 3–11. IEEE, 2004.
- [6] David Becker, Trish Dunn King, and Bill McMullen. Big data, big data quality problem. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2644–2653, 2015. doi: 10.1109/BigData.2015.7364064.

- [7] Hossein Bidgoli. *Threats, vulnerabilities, prevention, detection, and Management*, pages 249–256. Wiley, Hoboken, NJ, 2006.
- [8] Ingo Deutschmann, Peder Nordström, and Linus Nilsson. Continuous authentication using behavioral biometrics. *IT Professional*, 15(4):12–15, 2013. doi: 10.1109/MITP.2013.50.
- [9] Leah Findlater, Joan Zhang, Jon E. Froehlich, and Karyn Moffatt. Differences in crowdsourced vs. lab-based mobile and desktop input performance data. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 6813–6824, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346559. doi: 10.1145/3025453.3025820. URL <https://doi.org/10.1145/3025453.3025820>.
- [10] Daniel Genkin, Adi Shamir, and Eran Tromer. Rsa key extraction via low-bandwidth acoustic cryptanalysis. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, pages 444–461, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-44371-2.
- [11] Lorena Gonzalez-Manzano, Jose M. De Fuentes, and Arturo Ribagorda. Leveraging user-related internet of things for continuous authentication: A survey. *ACM Comput. Surv.*, 52(3), jun 2019. ISSN 0360-0300. doi: 10.1145/3314023. URL <https://doi-org.proxy.library.emory.edu/10.1145/3314023>.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL <https://arxiv.org/abs/1512.03385>.
- [13] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL <http://arxiv.org/abs/1608.06993>.

- [14] Xinyi Huang, Yang Xiang, Elisa Bertino, Jianying Zhou, and Li Xu. Robust multi-factor authentication for fragile communications. *IEEE Transactions on Dependable and Secure Computing*, 11(6):568–581, 2014. doi: 10.1109/TDSC.2013.2297110.
- [15] François Koeune and François-Xavier Standaert. A tutorial on physical security and side-channel attacks. *Foundations of Security Analysis and Design III: FOSAD 2004/2005 Tutorial Lectures*, pages 78–108, 2005.
- [16] Michael Lang, Lena Yuryna Connolly, Paul Taylor, and Phillip J. Corner. The evolving menace of ransomware: A comparative analysis of pre-pandemic and mid-pandemic attacks. *Digital Threats*, Aug 2022. ISSN 2692-1626. doi: 10.1145/3558006. URL <https://doi.org/10.1145/3558006>.
- [17] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, dec 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541. URL <https://doi.org/10.1162/neco.1989.1.4.541>.
- [18] Stewart Lee. Essays about computer security. page 181, 1999. URL <https://www.cl.cam.ac.uk/~mgk25/lee-essays.pdf>. Unpublished.
- [19] Onur Mutlu and Jeremie S Kim. Rowhammer: A retrospective. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(8):1555–1571, 2019.
- [20] Palo Alto Networks. What is lateral movement? <https://www.paloaltonetworks.com/cyberpedia/what-is-lateral-movement>, accessed 2023.
- [21] Lily Hay Newman. The Uber hack’s devastation is just starting to reveal itself, Sep 2022. URL <https://www.wired.com/story/uber-hack-mfa-phishing/>.

- [22] Hamed Okhravi. Moving target defense. In *Encyclopedia of Cryptography, Security and Privacy*, pages 1–4. Springer Berlin Heidelberg, Berlin, Heidelberg, 2023.
- [23] Aleksandr Ometov, Sergey Bezzateev, Niko Mäkitalo, Sergey Andreev, Tommi Mikkonen, and Yevgeni Koucheryavy. Multi-factor authentication: A survey. *Cryptography*, 2(1), 2018. ISSN 2410-387X. doi: 10.3390/cryptography2010001. URL <https://www.mdpi.com/2410-387X/2/1/1>.
- [24] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.
- [25] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):206–219, 2019. doi: 10.1109/JSTSP.2019.2908700.
- [26] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29(9):2352–2449, 2017. doi: 10.1162/neco\_a\_00990.
- [27] Spyridon Samonas and David Coss. The CIA strikes back: Redefining confidentiality, integrity and availability in security. *Journal of Information System Security*, 10(3), 2014.
- [28] Adi Shamir and Eran Tromer. Acoustic cryptanalysis, 2004.
- [29] S. S. Stevens, J. Volkman, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937. doi: 10.1121/1.1915893. URL <https://doi.org/10.1121/1.1915893>.

- [30] Zhang Tao, Fan Ming-Yu, and Fu Bo. Side-channel attack on biometric cryptosystem based on keystroke dynamics. In *The First International Symposium on Data, Privacy, and E-Commerce (ISDPE 2007)*, pages 221–223, 2007. doi: 10.1109/ISDPE.2007.48.
- [31] Zhihong Tian, Wei Shi, Yuhang Wang, Chunsheng Zhu, Xiaojiang Du, Shen Su, Yanbin Sun, and Nadra Guizani. Real-time lateral movement detection based on evidence reasoning network for edge computing environment. *IEEE Transactions on Industrial Informatics*, 15(7):4285–4294, 2019. doi: 10.1109/TII.2019.2907754.
- [32] Cybersecurity Ventures. Cybercrime damages to cost the world \$8 trillion usd in 2023. [https://www.einnews.com/pr\\_news/606505844/cybercrime-damages-to-cost-the-world-8-trillion-usd-in-2023](https://www.einnews.com/pr_news/606505844/cybercrime-damages-to-cost-the-world-8-trillion-usd-in-2023), 2022.
- [33] Ymir Vigfusson. Devices, systems and methods for securing communication integrity. US Patent Application No. 63/330,875, 2023. Patent (Pending).
- [34] Luis Von Ahn and Laura Dabbish. Designing games with a purpose. *Communications of the ACM*, 51(8):58–67, 2008.
- [35] Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. Captcha: Using hard ai problems for security. In *Eurocrypt*, volume 2656, pages 294–311. Springer, 2003.
- [36] Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008. doi: 10.1126/science.1160379. URL <https://www.science.org/doi/abs/10.1126/science.1160379>.
- [37] Yuval Yarom and Naomi Benger. Recovering OpenSSL ECDSA Nonces Using the FLUSH+RELOAD Cache Side-channel Attack. Cryptology ePrint Archive,

Paper 2014/140, 2014. URL <https://eprint.iacr.org/2014/140>. <https://eprint.iacr.org/2014/140>.

- [38] J. Youngberg and S. Boll. Constant-q signal analysis and synthesis. In *ICASSP '78. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 375–378, 1978. doi: 10.1109/ICASSP.1978.1170547.
- [39] Tianming Zhao, Yan Wang, Jian Liu, Yingying Chen, Jerry Cheng, and Jiadi Yu. Trueheart: Continuous authentication on wrist-worn wearables using ppg-based biometrics. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 30–39, 2020. doi: 10.1109/INFOCOM41043.2020.9155526.