ADDRESSING PRIVACY CHALLENGES IN MODERN AUDIO-VIDEO COMMUNICATION SYSTEMS AND APPLICATIONS

by

MOHD AMJAD SABRA, B.S.

DISSERTATION
Presented to the Graduate Faculty of
The University of Texas at San Antonio
In Partial Fulfillment
Of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

COMMITTEE MEMBERS:
Murtuza Jadliwala, Ph.D., Chair
Anindya Maiti, Ph.D.
Ravi Sandhu, Ph.D.
Wei Wang, Ph.D.
Palden Lama, Ph.D.

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Sciences
Department of Computer Science
May 2023

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Murtuza Jadliwala for his patience and immense knowledge, which he used to mold me and my research to its very best version. His support helped me develop the analytical skills needed for my dissertation. Without Dr. Jadliwala, I would not have this dissertation. Dr. Jadliwala played a huge part (and was the reason) in how I entered the field of privacy in audio-visual communication. Additionally, Dr. Jadliwala believed in me and mentored me as an undergraduate (before joing the PhD. program), and later gave me the opportunity to enter into a Ph.D. program after graduating with my B.S. degree. I am very grateful for the trust and everything Dr. Jadliwala has done for me.

I would also like to acknowledge Dr. Anindya Maiti's insightful comments and encouragement, which helped me produce my research work. Dr. Maiti has been with me throughout most of my Ph.D program (even before joining the Ph.D. program) and guided me throughout my research. Dr. Maiti was not only a mentor but a friend. I am very fortunate to have known Dr. Maiti, and I am very grateful and very appreciative of all the mentoring and advice I received from him both in my Ph.D. career and in my day-to-day life.

I would also like to extend my thanks to the other committee members, Dr. Ravi Sandhu, Dr. Wei Wang and Dr. Palden Lama for being part of my committee and receiving feedback for my work.

I would like to thank my labmates and other PhD students whom I worked with. Finally I would like to thank my family for supporting me and being with me.

May 2023

ADDRESSING PRIVACY CHALLENGES IN MODERN AUDIO-VIDEO COMMUNICATION SYSTEMS AND APPLICATIONS

Mohd Amjad Sabra, Ph.D.
The University of Texas at San Antonio, 2023

Supervising Professor: Murtuza Jadliwala, Ph.D.

Catalyzed by advances in communication and Internet technology, web-based audio-video calling has become a mainstream method of remote communication. Recently, the trend has seen a further boost due to the COVID-19 pandemic, whereby audio-video calls by means of applications such as Skype and Zoom have become the default medium for professionals to confer remotely and for students to attend lectures from home. In addition, modern Virtual Reality (VR) devices and systems take this on step further by enabling applications that allow users to remotely co-locate and communicate in the same virtual space or world. Despite their extreme popularity and utility, audio, video and sensor data made available by these modern communication systems and applications could contain sensitive information about the participants, their surroundings or their current activity and context, and can present significant user-privacy challenges if not appropriately protected. This dissertation, at a high level, studies and demonstrates the feasibility of several novel user-privacy threats in popular web-based video-calling (e.g., Skype and Zoom) and virtual reality (e.g., VR Chat) applications and proposes novel mitigation and protection measures against these threats.

Specifically, this dissertation has already made the following three significant contributions: (i) First, the dissertation investigated if an adversary, who is at one end of an online video call, can infer some potentially sensitive information about the participant at the other end which is not trivially visible/audible from the call? More specifically, the dissertation evaluated the feasibility of inferring keystrokes of a target user on a traditional QWERTY keyboard by just observing their video feed on a video calling application. This was accomplished by modeling commonly observed typing behaviors during a video call, and utilizing them to construct a novel video-based keystroke and typing detection framework. A text inference framework then uses the keystrokes detected from the video to predict words that were most likely typed by the target user. The proposed keystroke/typing detection and text inference frameworks were then empirically evaluated using data collected from a large number of human subject participants in several practical settings and scenarios. Finally, multiple techniques to mitigate such keystroke inference attacks from video calls were also propose and evaluated. (ii) Second, a popular privacy feature in online video calls virtual backgrounds or background filter was extensively studied to understand how effective it was in protecting users' actual backgrounds, and the sensitive information therein. For that, a novel background reconstruction framework, which reconstructs the real background in a video call that has a virtual background blended in, was first designed. Then, a through investigative analysis of the virtual background feature was accomplished by employing the real background (partially) reconstructed by this framework to carry out four different privacy attacks, namely, location inference, specific object tracking, generic object inference, and text inference attack. Finally, by means of video call data collected from real human subject participants (in a variety of different settings and parameters) and prerecorded videos collected in the wild, the performance of the proposed inference frameworks was empirically verified. As before, several mitigation strategies were also proposed and evaluated. (iii) The third aspect of this dissertation research focuses on investigating the privacy of user identities in virtual reality (VR) applications, where users may be recorded by an adversary while using worn-out-of-band motion sensors, such as smartwatches and smartphones. To address this issue, we record the video of virtual avatars to represent the users' movements in the VR application. We then use existing activity machine learning classifiers to classify video and motion data, and correlate both data streams using the Hamming distance and the Spearman Rank Correlation Coefficient. However, we have discovered that the time complexity of our correlation algorithm is not practical for large-scale applications. To overcome this challenge, This dissertation presents optimized correlation algorithm that balances speed and accuracy, while being feasible for large-scale data sets.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Catalyzed by the ubiquity of the Internet, audio-video calling has become a mainstream method of remote communication [39]. The trend has recently seen a further boost due to the COVID-19 pandemic [7], whereby audio-video calls became the default medium for professionals to confer remotely and for students to attend lectures from home. Nonetheless, audio-video calls (often referred to as just video calls) should be considered privacy sensitive as participants may have spoken or displayed private information during the call. To protect video calls from eavesdropping threats, secure video calling protocols are usually end-to-end encrypted. However, even if we disregard widespread system weaknesses [3, 45], end-to-end encryption may be ineffective when an adversary is present at one end of the video call.

Can an adversary, who is at one end of a video call, infer some potentially sensitive information about the participant at the other end which is not trivially visible/audible from the call? Modern video calling softwares such as Skype [29], Hangouts [10] and Zoom [43] already provide features such as background-blurring to enable users to potentially blur/hide everything in the users' background, except their body. In this work, we want to investigate what sensitive information can be inferred by just observing a target users' body and physiological features in an audio/video call. More specifically, we would like to investigate the feasibility of inferring keystrokes of a target user on a traditional QWERTY keyboard by just observing their video feed on a video calling application such as Skype, Hangouts and Zoom.

Prior efforts in the literature have shown that the sound (audio) of keystrokes typed during a video call can be exploited to infer the text typed [57, 77]. But, audio-based keystroke inference is not very practical primarily because of naturally occurring (audio) noises in an audio-video call signal, such as background sound and participants talking [57]. Moreover, such audio-based attacks may not work for the relatively *quieter* membrane and dome-switch keyboards, because (i) the low amplitude keystroke audio emanations are not effectively captured by many entry-level microphones [152], and (ii) non-vocal low amplitude audio frequencies are often filtered out by many video calling applications [13, 44].

We believe that the video signal in such calls is less prone to naturally occurring noises and can be exploited for effective keystroke inference attacks. It is also a relatively unexplored modality for keystroke inference.

Keystroke inference is not the only thing a user must be worried about while in a video call. This dissertation also studies the effectiveness of a popular privacy-preservation technique in online video calls - *virtual backgrounds*. A virtual background (also sometimes referred to as a *background filter*) is an image overlay which can be used by online video call participants to hide their real or actual backgrounds. In other words, the virtual background feature, which is available in most online video conference applications, allows video call participants to cover their actual backgrounds with an image of their own choice such that the overlay image hides or obscures most, if not the entire, actual background. Video calling applications accomplish this by separating the foreground (containing the user) and the background using image processing techniques such as *image matting* [164]. Virtual backgrounds is a very popular add-on feature and is extensively employed by a large percentage of video call users to obscure personal or intimate objects in their background or to anonymize sensitive context such as location [34, 35]. This highlights an important underlying issue: *how effective is the virtual background (or background filter) feature in protecting users' actual backgrounds, and the sensitive information therein, in online video calls?* We address this issue by undertaking an investigative analysis of the virtual background feature in popular video calling applications to study how they perform against our novel real background reconstruction strategy and related privacy attacks.

This dissertation also looks at another modular of privacy within visual-audio communications, the privacy of identity. More specifically, identity privacy in virtual reality (a more modern form of visual-audio communication). Virtual Reality (VR) is changing the paradigms of human-computer interaction, and has become a ubiquitous consumer technology [9, 16, 17, 19, 28, 30, 38]. Most prevalent VR systems today (e.g., Meta Quest, HP Reverb and Sony PlayStation VR), offer an immersive audio-visual experience where users can navigate around a digitally represented 3D space (i.e., a *virtual world*) using a VR headset. In most VR systems, users would navigate and interact with this virtual world using on-body (often, handheld) controllers that can track users' body movements in the real world and execute analogous movements in the virtual world. Reactions from users' navigation actions and interactions with the virtual world are relayed back to the user by means of video, audio, and haptic (e.g., vibrations) signals perceptible to the user through his/her VR device/headset.

VR systems enable several novel applications that were previously not possible using traditional desktop and mobile devices, such as immersive gaming [24], remote conferencing [16, 19], virtual tourism [46, 51], social networking [49, 50], visualizing 3D models [9, 28], and large open-world spaces [17, 30]. Unfortunately, at the same time new privacy and security challenges have also emerged in the VR space. For example, password inference from finger movements (using motion sensors) when typing a password in the virtual world can become a security problem if the same password is reused by the user in real world [80]. Some VR headsets also include eye-tracking, which can become an additional channel for inference of private data. For instance, it has been shown in recent research efforts that eye tracking or gaze data could be potentially misused to infer a user's personal information and traits such as gender, age, ethnicity, body weight, personality traits, drug consumption habits, emotional state, skills and abilities, fears, interests, and sexual preferences [99]. Personal gait and movement data collected from a VR headset can also be used in conjunction with Deepfake videos to create highly authentic looking fake videos [157], which can be further used to damage personal reputation [20, 138], conduct social engineering attacks [27, 160], and spread misinformation (fake news) [84, 97].

VR systems and applications can transport users from the real world to a virtual world, wherein the two worlds are seemingly disconnected from each other. *In this work, we study the potential of a new type of privacy threat targeting VR users by connecting their activities visible in the virtual world (enabled by some VR application/service) to their physical state sensed in the real world.* More specifically, we analyze the potential of carrying out a *de-anonymization* or *identification* attack on VR users by *correlating* visually observed movements of users' anonymous humanoid avatars in the virtual world with auxiliary data representing their context/state in the physical world. For such auxiliary information, we specifically focus on data available from motion sensors on-board mobile and wearable devices (e.g., smartphones and smartwatches) that users may be carrying on them while navigating or interacting with virtual worlds in VR applications. Our attack is further motivated by the fact that motion sensors on-board modern mobile devices, such as accelerometers and gyroscopes, can be recorded by any on-device app at a high frequency. This, consequently, enables easy misuse of such motion data by any on-device app, something which has been extensively documented in the security research literature [70, 81, 89, 90, 106, 108, 116, 121, 127, 143, 144]. Moreover, we hypothesize that fine-grained user movement captured by on-body motion sensors (such as those on a user's wrist in the form of a smartwatch and in a user's pocket in the form of a smartphone) are strongly correlated with the visual motions observed in the user's humanoid avatar in the virtual environment of a VR app, and thus can potential be used to de-anonymize users in virtual spaces.

We consider an attack scenario where the adversary is trying to de-anonymize a target user in the virtual world by visually tracking the motion of the user's avatar and then correlating it with labeled motion data streams belonging to a (large) set of users, which also includes motion data from the target user. We refer to this set of labeled motion data streams (belonging to a large set of users) as the target user's *confusion set*. Such an attack scenario could arise in many popular VR services such as Metaverse [17], VRChat [38] and MeetinVR [16] which enable a group of users to organize events, get-togethers and games in some virtual environment. A mobile (smartphone) app of an adversarial service provider can be used to stealthily record motion data of every user in the group for a particular event, while video data corresponding to a target user (in the group) can be captured by the adversary directly from the virtual environment/world, say, by participating or entering the same virtual environment/world as the target user. We propose a novel correlation framework to carry out the de-anonymization in such VR services, and comprehensively evaluate parameters such as effect of different actions/movements and the effect of using having their mobile device in different bodily locations (such as different pockets).

CHAPTER 2: PRIVACY OF ACTIVITY

**This chapter has previously appeared in publication with Network & Distributed System Security Symposium (NDSS), 2021. It was co-authored by Anindya Maiti, Ph.D., Murtuza Jadliwala, Ph.D. and has been reproduced here with minor revisions.*

2.1 Introduction

First, we study the privacy of user activity in a video call, more specifically, if keystroke inference is possible from visual data. Our contributions in this chapter includes modeling commonly observed typing behaviors during a video call, and utilizing them to construct a novel video-based keystroke and typing detection framework. We then create a text inference framework that uses the keystrokes detected from the video to predict words that were most likely typed by the target user (Section 2.5). We then comprehensively evaluate both the keystroke/typing detection and text inference frameworks using data collected from a large number of human subject participants in two different settings: (i) an *In-Lab* setting (Section 2.6) where the video call setup, including the device(s) used for the calls, participants' sitting position during the call, and the text typed by the participants is fixed, and (ii) an *At-Home* setting (Section 2.9) where the video call data is collected in a realistic environment without any constraints or requirements. Evaluation results for the In-Lab setting are outlined in Sections 2.7 and 2.8, while results for the At-Home setting appear in Section 2.10.

We also propose and evaluate multiple techniques which can help in the mitigation of such keystroke inference attacks from video calls (Section 2.11).

2.2 Related Work

The research literature is rich with various modalities of side-channel inference threats targeting different types of private information. We limit our literature review discussion in two closely related directions as follows.

Keystroke Inference Threats. This direction deals with the class of inference threats that aim to reconstruct text typed by a target user, using one or more modalities of side-channels. Keystroke inference attacks can have potentially dangerous consequences as the text typed can often be private in nature, and can sometime even contain sensitive information, such as credit card numbers, authentication codes, and addresses. Side-channel keystroke inference threats have utilized electromagnetic emanations from keyboards [153], optical or visual cues [75, 142, 148], Wi-Fi channel state information [54, 103], audio or acoustic signals from keyboards [57, 59, 65, 77, 88, 172, 173], typing-related table vibrations captured by a nearby sensor [112], smartphone motion sensors (to infer text typed on the smartphone) [71, 119, 128, 167], and wearable motion sensors [106, 108, 110, 154, 155]. Among these prior works, [77] and [57] are the most closely related research efforts to ours. Both works demonstrated the feasibility of accurate keystroke inference threats from keystroke sounds propagated over a video call. However, as mentioned earlier, such sound or audio based threats may not be practical because of naturally occurring interferences (such as participants talking) and background noise-cancellation techniques being used by many video calling applications [13, 44] that also eliminates/reduces the propagation of keystroke sounds.

Inference Attacks Using Visual Side-Channels. With the recent ubiquity of video capturing hardware embedded in many consumer electronics, such as smartphones, tablets, and laptops, the threat of information leakage through visual channel has amplified. Moreover, high-end digital cameras and lenses have also trickled down to the consumer market at competitive prices, making them easily accessible to an adversary. Earlier works leveraged upon optical emanations from monitors [100] or from eyes [60] to infer information such as content being displayed or watched. More recent works studied information leakage due to outdoor light effusions, such as inference of multimedia consumption [109, 166] and private data exfiltration using smart lights [109, 135]. A few research works also leveraged on visual side-channel for keystroke inference threats. Simon and Anderson [142] were able to infer keystrokes made on a smartphone based on visual movements, captured from the on-device camera, that occurred as a result of individual keystrokes. Similarly, Sun et al. [148] were able to infer keystrokes typed on a tablet just from visual observation of the rear side of the tablet. Chen et al. [75] leveraged on users' eye movements for touchscreen keystroke inference. To the best of our knowledge, This chapter was the first work that proposes and evaluates a keystroke inference framework that solely leverages body movements observable in video calls.

2.3 Background

In this section, we describe the different factors affecting typing-related body movements observable in a video call.

Muscles, Joints and Motor Control. Human bodily movements, clinically known as motor functions, are achieved primarily through movement of joints. Joints are formed where two or more bones are connected via ligaments, a flexible fibrous connective tissue which binds together bones and/or cartilages. These joints allow several degrees of freedom in the human body. With the help of muscles that wrap around the bones and connect to the bones via tendons, along with motor control signals from the brain that appropriately engages the required set of muscles, the human body can carry out tasks through coordinated joint movements. Tasks may range from basic balance and stability of the body to more complex actions, such as running or typing on a keyboard. Also, most tasks are actuated through movement of multiple joints simultaneously, rather than just one at a time.

When a user starts typing a sentence, the initial set of joint movements are significantly dependent on the keyboard positioning and user's typing habits (i.e., the set of motor control signals learned by the brain). A user can exercise an elbow joint, shoulder joint or joints from the Carpus and Metacarpus regions or a combination of all of them, that ultimately positions the user's finger on the initial key. The joint movements associated with keystrokes following the initial keystroke depends primarily on the user's typing style, e.g., *hunt-and-peck*, *touch-typing*, or *hybrid*.

Certain typing styles, such as hunt-and-peck, result in significant upper hand movements (not just fingers or wrist) between keystrokes, than others. For instance, in hunt-and-peck typing the elbow, shoulder, and Carpus joints are heavily utilized, whereas in touch-typing the Carpus, Metacarpus, and Phalanges joints are heavily utilized.

Reaction Force of a Keystroke. A common phenomenon observed across all typing styles is that whenever the user presses a key, a reaction force is produced in the opposite direction (Newton's third law of motion). This reaction force propagates throughout the arm and shoulder muscles and joints until the force is absorbed by the body. As a result, even if the user uses only the joints in the Phalanges bones to press a key, one can visually observe subtle arm and shoulder movements due to the reaction force exerted by the key on the user's hand. However, the *five fingers* of each hand are connected through different bones in the wrist, that have different joints in the Carpus area.

As a result, the reaction force of a keystroke propagates slightly differently through the arm and shoulder muscles and joints, depending on which finger was used to press the key. Visually, this translates to distinguishable types of upper hand movements (which is observable through a webcam during a video call) for key presses with different fingers.

Characteristics of the Available Call Data. During a typical video call, an adversary can leverage two different types of information for inferring keystrokes made during the call.

(1) Video Data or Feed: Most modern video calling applications employ a webcam to capture the visual and/or audio signals during the call. The webcam's camera sensor captures the visual image of the target subject and his/her surroundings as a series of two-dimensional images, also known as *frames*. In other words, a video is a chronological ordering of two dimensional images or frames, displayed in that order at a very high rate or speed (often measured in frames/second or fps in short). A typical modern webcam can capture 30 or 60 fps, and at 1280×720 (921,600 $pixels$) or 1920×1080 (2,073,600 $pixels$) resolution per frame. Based on the traditional position of a webcam during a video call, lateral movements of hand and shoulder can easily be observed in the captured video.

(2) Audio Data or Feed: The sound during a video call is typically captured either using a microphone sensor integrated within the webcam or using an external microphone. The captured sound often contains both the user's voice (or speech) and background/ambient noise, including sound related to the keystrokes made by the user or any other activities performed by the user

during the call. Video calling softwares also often implement audio optimizations, such as dampening/filtering of non-vocal frequencies [13, 44] and echo suppression/cancellation. Most modern microphones can record audio signals at $44,100Hz$ or more, and such high-fidelity audio can capture fine-grained audio characteristics. However, a microphone may not always be enabled by the user during a video call, as observed in the recent popularity of the push-to-talk feature provided by most video calling software that lets a user mute sound at the push of a button. Also, during multi-participant video conference calls, it is a common courtesy or etiquette for participants to mute their microphone when they are not actively speaking. Nonetheless, while our attack framework employs only the available video data for keystroke inference, we assume the availability of audio data for comparative evaluation with a prior work [77].

2.4 Adversary Model

The goal of an adversary in our setting is to infer keystrokes typed by a target user at the opposite end of a video conference/call by just employing the video feed from the call. To undertake a purely video-based keystroke inference attack, we assume that the adversary first records the video feed of the call where the target user was a participant, and that the target user typed private text on her/his keyboard during the call. More specifically, we assume a field-of-view of a typical desktop or laptop webcam where the video stream would consist of only visible upper body movements of the user (and not the actual keys being typed as then the attack would be trivial), as shown in Figure 2.2. Additionally, we assume that both shoulders and upper arms are within the field-of-view of the webcam, which is a practical assumption because desktop and laptop webcams are often positioned centrally with respect to the user. In addition to targeting participants in live video calls, an adversary could also potentially target videos obtained from public video sharing/streaming platforms such as YouTube [42] and Twitch [32]. Many live streamers interact with their laptop or desktop during a live video exposition, which could include sensitive typed information, and thus, potentially targeted by an adversary. As outlined earlier, the recorded video is nothing but a series of picture frames sampled at a constant frequency, and the adversary's goal is to utilize the

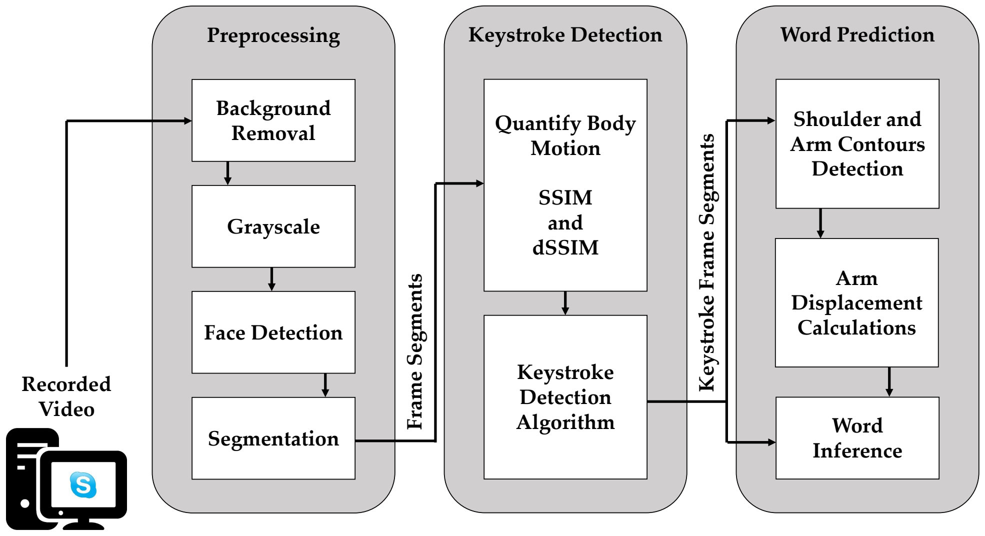


Figure 2.1: Overview of the proposed keystroke inference framework.

observable upper body movements across all the recorded frames to infer the private text typed by the target. The adversary does not have any other medium of inferring the private text typed, and must rely entirely on the video stream. This makes our attack more practical as it can target not only real-time video calls (both in a public and private setting), as well as, archived videos of live exposition/events.

2.5 Attack Framework

2.5.1 Overview

The adversary has to formulate two key procedures to draw a relationship between typing related body movements observable in the video and the text being typed: First, within the video stream the adversary must be able to accurately identify the occurrences of keystrokes based on the upper body movements. Second, by modeling the body movement characteristics immediately before, after, and in-between detected keystrokes, the adversary must be able to accurately predict words and sentences typed by the target user. Let's first intuitively describe how the adversary can accomplish these objectives by giving an overview of the different components of our video-based keystroke inference framework (Figure 2.1). We will later provide details of each of these components.

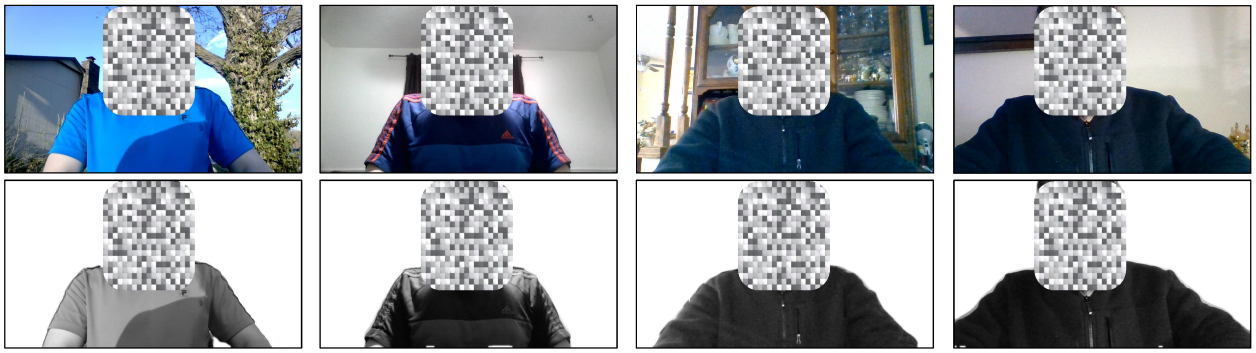


Figure 2.2: Example output of the background removal process, using `DeepLabv3` and `Microsoft COCO`, successfully applied in different indoor and outdoor settings. Top four images are the original frames, and bottom four images are corresponding frames after background removal.

The recorded video first undergoes multiple pre-processing steps in the following order: (i) *background removal*, (ii) conversion to *grayscale*, (iii) *face detection*, and (iv) *segmentation* of left and right arm regions based on their relative position with respect to the face. After pre-processing, the framework employs a keystroke detection algorithm based on Structural SIMilarity (*SSIM*) index [156] across all the frames in each of the left and right side video segments. Finally, the framework computes several motion features from the video segments immediately before and after each detected keystroke, and employs them in a dictionary-based prediction algorithm for word inference. Let's now provide details of each component.

2.5.2 Pre-processing

Given a video v composed of l_v frames recorded at $30Hz$, let us denote the set of frames in the video as $v = \{f_1, f_2, f_3, \ldots, f_{l_v}\}$. Assuming that the video resolution is constant, each frame in v is composed of m rows and n columns of pixel values such that each pixel $p_{i,j}$ (where $i \in \{1, 2, 3, \ldots, m\}$ and $j \in \{1, 2, 3, \ldots, n\}$) in a frame represents a RGB value. The RGB value of a pixel represents the hue (color), saturation, and brightness of that particular pixel in the frame. With this representation of a recorded video, we now describe the four pre-processing steps, in sequential order.

Background Removal. The background removal process is applied to all frames ($f_i \in v$), in

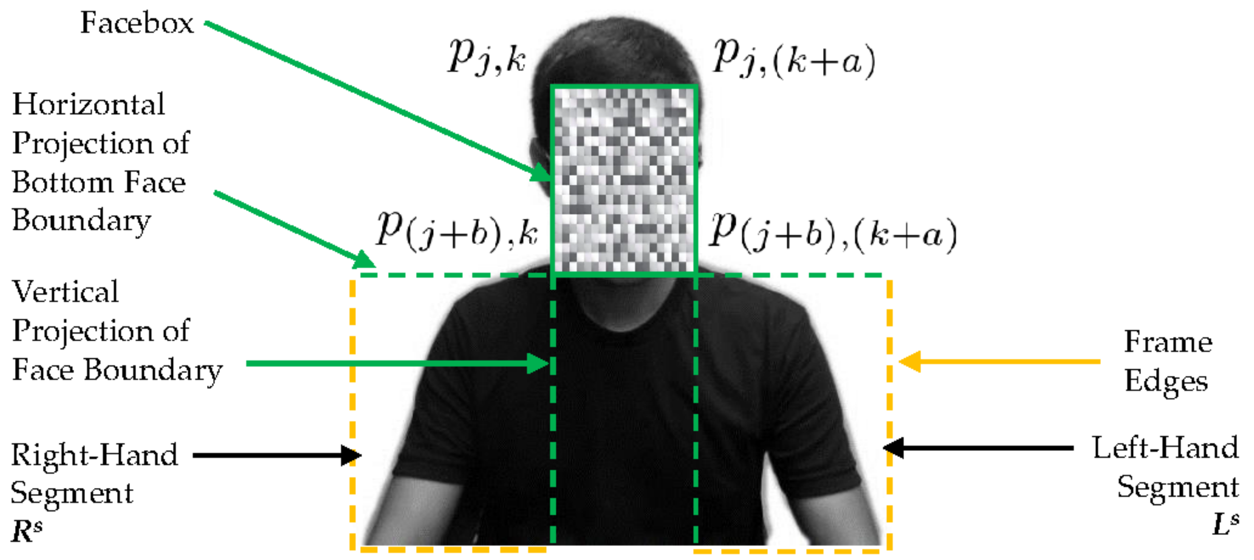


Figure 2.3: Face detection using `Faceboxes` and segmentation of left and right arms, in a frame from the captured video.

order to identify the location of the body in the frame. We utilize the `DeepLabv3` model [74] for this task, which employs *Atrous Convolution* with upsampled filters to extract dense feature maps and to capture long range context. Training of the model is done using he `Microsoft COCO` dataset [105], which contains a rich set of human body related training samples. With the background removed, we can focus purely on the body's relative movements vis-à-vis typing. Example outputs of this background removal process is shown in Figure 2.2. This background removal step makes our proposed framework agnostic to any moving elements in the background. Let's denote background-removed frames as $^{r}f_i$.

Grayscale. We next convert all background-removed frames ($\{^{r}f_1, ^{r}f_2, ^{r}f_3, \ldots, ^{r}f_{l_v}\}$) to colorimetric grayscale [147], using the RGB values of individual pixels in a frame. This conversion to grayscale simplifies all following steps by making them color-independent. Let's denote such background-removed grayscale frames as $^{rg}f_i$.

Face Detection. Our next objective is to focus specifically on the two arms, where typing related motion is most perceptible. However, as webcam setups may not be predetermined or homologous, we require a webcam and webcam setup agnostic methodology for automatically and accurately identifying arm regions across all background-removed grayscale frames ($\{^{rg}f_1, ^{rg}f_2, ^{rg}f_3, \ldots, ^{rg}f_{l_v}\}$).

To do so, we leverage on the consistency in relative position of the target user's arms with respect to their face (Figure 2.3). The intuition is that the left arm will be located around the bottom-right of the face in the frame, and similarly the right arm will be located around the bottom-left of the face in the frame. Face detection is a matured research topic, with several state-of-the-art frameworks and training datasets readily available. We utilize the CPU-friendly `Faceboxes` model [171], that employs Rapidly Digested Convolutional Layers (RDCL) and the Multiple Scale Convolutional Layers (MSCL), to detect target user's face in each frame. For training the `Faceboxes` face detection model, we used the `WIDER FACE` dataset [41], that consists of 12,880 diverse facial images.

Segmentation. The *facebox* generated by `Faceboxes` identifies the user's face and draws a rectangular boundary around it (solid green rectangle in Figure 2.3). The objective of this last part of the pre-processing is to utilize this *facebox* in order to automatically segment the left and right arms in the background-removed grayscale frame. Assume that in a given frame ${}^{rg}f_i$, the four vertices of the generated *facebox* are located at pixels $p_{j,k}$, $p_{j,(k+a)}$, $p_{(j+b),k}$, and $p_{(j+b),(k+a)}$, where a and b are the width and height of the *facebox* (in pixels), respectively. Using these *facebox* vertices, the left arm segment is calculated as the rectangular area of the frame enclosed within the pixels $p_{(j+b),(k+a)}$, $p_{(j+b),n}$, $p_{m,(k+a)}$, and $p_{m,n}$. Similarly, the right arm segment is calculated as the area of the frame enclosed within the pixels $p_{(j+b),1}$, $p_{(j+b),k}$, $p_{m,1}$, and $p_{m,k}$. Let's denote the left and right arm segments extracted from a frame ${}^{rg}f_i$ as L_i^s and R_i^s, respectively.

2.5.3 (Potential) Keystroke & Typing Activity Detection

Using the preprocessed left and right arm segments from all frames of the video (L_i^s and R_i^s, respectively, where $i \in \{1, 2, 3, \ldots, l_v\}$), our next objective is to precisely determine the time-stamps, i.e., the frames, when a keystroke was typed using either hand. This is non-trivial because the adversary does not have a view of the lower arm. In this section, we propose a novel keystroke detection algorithm which accurately detects keystroke events using only upper hand arm movements as observed in L_i^s and R_i^s. The proposed keystroke detection algorithm is applied independently for each arm, i.e., the left and right arm segments L_i^s and R_i^s ($i \in \{1, 2, 3, \ldots, l_v\}$) are processed independently for the keystroke detection task.

Algorithm 2.1 Keystroke detection algorithm.

```
1: Input: segmented arm frames stored in $armS[]$
2: Output: keystroke frames stored in $keystrokesFS[]$
3: procedure KeystrokeDetect
4:     $armS[]$                                          ▷ $R^s$ or $L^s$
5:     $ssimList[]$                                      ▷ Series of SSIM scores
6:     $ssimDiff[]$                                      ▷ $ssimL[i] - ssimL[i+1]$
7:     $keystrokesFS[]$                                  ▷ Store keystroke containing $R^s$ or $L^s$
8:     for $i$ in range($armS.size() - 1$) do
9:         $ssimScore = SSIM(armS[i] - armS[i+1])$
10:        $ssimList.append(ssimScore)$
11:    end for
12:    for $i$ in range($ssimList.size() - 1$) do
13:        $ssimDiff.append(ssimList[i] - ssimList[i+1])$
14:    end for
15:    $mean \leftarrow mean(ssimDiff)$
16:    $std \leftarrow standardDeviation(ssimDiff)$
17:    for $i$ in range($ssimDiff.size() - 1$) do
18:        $zScore = (ssimDiff[i] - mean)/std$
19:        if $zScore > \phi_a$ and $zScore < \phi_b$ then
20:            if $ssimDiff[i]$ is a local max then
21:                if local min exists between $i \rightarrow i+2$ then
22:                    if $zScore(localmin) < \phi_c$ then
23:                        $keystrokesFS$.append($armS[i]$)
24:                    end if
25:                end if
26:            end if
27:        end if
28:    end for
29: end procedure
```

Intuition. Every time the target user presses a key on her/his keyboard, she/he undertakes some degree of hand movement, the extent of which may vary depending on the typing style and position of the key on the keyboard. This movement may be from a resting position, or from an earlier keystroke using the same hand. Moreover, every keystroke lasts for a few milliseconds, until the user depresses the key, and during this time there is little to no movement. Finally, after the keystroke is completed, the user's hand moves on to another key or back to a resting position. Intuitively, we should be able to observe this pattern of body movements in the video (v). Accordingly, our keystroke detection algorithm (Algorithm 2.1 is designed based on empirically observed characteristics of the (left or right) arm segments immediately before, during, and immediately after a keystroke. The empirically observed characteristics that we leverage upon, as described below, are fairly independent of the typing style. For simplicity, going forward we will describe the keystroke detection process only for the left-hand. The process for the right hand is identical.

Quantifying Body Motion. $SSIM$ [156] is a well-known metric for measuring the similarity between two images, and a high $SSIM$ index between consecutive frames would denote insignificant body movements, and vice versa. To quantify left-hand body movements across consecutive frame segments, we compute $SSIM$ index between every L^s_i and L^s_{i+1} ($i \in \{1, 2, 3, \ldots, l_v - 1\}$). This results in a series of $SSIM$ indices $SSIM^{L^s} = \{L^s_1 \bowtie L^s_2, L^s_2 \bowtie L^s_3, \ldots, L^s_{l_v-1} \bowtie L^s_{l_v}\}$, where $\bowtie$ is the $SSIM$ operator. To understand the rate of change in body movements across consecutive frame segments, we also compute the discrete derivative of $SSIM^{L^s}$ as $dSSIM^{L^s} = \{(L^s_1 \bowtie L^s_2) - (L^s_2 \bowtie L^s_3), (L^s_2 \bowtie L^s_3) - (L^s_3 \bowtie L^s_4), \ldots, (L^s_{l_v-2} \bowtie L^s_{l_v-1}) - (L^s_{l_v-1} \bowtie L^s_{l_v})\}$. In terms of body motion detected between the frame segments, $SSIM^{L^s}$ may be viewed as the 'speed' and $dSSIM^{L^s}$ as the 'acceleration'. Similarly, we also independently compute $SSIM^{R^s}$ and $dSSIM^{R^s}$ for the right hand.

Observed Characteristics. We observed a consistent pattern in the $dSSIM^{L^s}$ measurements, which is in line with our intuition outlined earlier. We observed that:

(1) If the video is recorded at 30 fps, and a keystroke occurred in frame t, there exists a local maxima in $dSSIM^{L^s}$ at $(L^s_{t-2} \bowtie L^s_{t-1}) - (L^s_{t-1} \bowtie L^s_t)$. This is a result of the increase in body motion immediately before a keystroke followed by the lack of body motion for the duration of the key press.

(2) The above local maxima is followed by a local minima within the next 0.05 sec. For a video captured at 30 fps, this means that the local minima occurs among the next two elements of $dSSIM^{L^s}$, i.e, $(L^s_{t-1} \bowtie L^s_t) - (L^s_t \bowtie L^s_{t+1})$ or $(L^s_t \bowtie L^s_{t+1}) - (L^s_{t+1} \bowtie L^s_{t+2})$. This is a result of the lack of body motion for the duration of key press followed by the body motion immediately after a keystroke when the user's hand moves on to another key or back to a resting position. If no local minima is detected within this time frame, it would imply that the

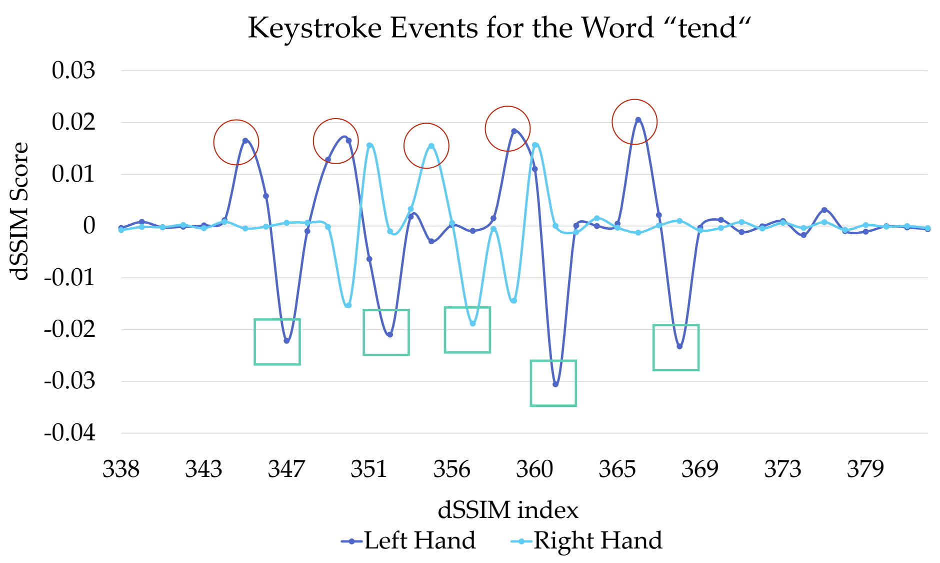


Figure 2.4: An example of $dSSIM^{L^s}$ during a typed word. Circles represent the actual keystroke event, whereas squares represent the local minima within two frames of every keystroke.

user's body movements are likely not related to keystrokes.

An example of this pattern can be observed in Figure 2.4.

Keystroke Detection Algorithm. We utilized the above observed characteristics to design a keystroke detection algorithm (Algorithm 2.1) that automatically labels frames where keystrokes potentially happened. In addition to the above observed characteristics, Algorithm 2.1 also employs a *filtering technique* to eliminate body movements that are not related to typing activity, but may still trigger a false positive. Algorithm 2.1 filters based on statistical analysis of magnitudes in $dSSIM^{L^s}$. According to this filtering technique, a frame t is considered to be a keystroke frame if:

(1) $dSSIM^{L^s}$ at $(L^s_{t-2} \bowtie L^s_{t-1}) - (L^s_{t-1} \bowtie L^s_t)$ lies between $\phi_a \sigma_{dSSIM^{L^s}}$ and $\phi_b \sigma_{dSSIM^{L^s}}$, in addition to being the local maxima. $\sigma_{dSSIM^{L^s}}$ is the standard deviation in the distribution of magnitudes in $dSSIM^{L^s}$, and optimal values of ϕ_a and ϕ_b can be determined empirically.

(2) The local minima following t in $dSSIM^{L^s}$ $((L^s_{t-1} \bowtie L^s_t) - (L^s_t \bowtie L^s_{t+1})$, or $(L^s_t \bowtie L^s_{t+1}) - (L^s_{t+1} \bowtie L^s_{t+2}))$ is less than $\phi_c \sigma_{dSSIM^{L^s}}$. Again, $\sigma_{dSSIM^{L^s}}$ is the standard deviation in the distribution of magnitudes in $dSSIM^{L^s}$, and optimal value of ϕ_c can be determined empirically.

The effect of different ϕ_a, ϕ_b, and ϕ_c values on keystroke detection is presented in Section 2.7.1. Appropriately chosen values of ϕ_a, ϕ_b, and ϕ_c would ideally eliminate false positives related to frequently occurring minor body movements that are closer to the mean value ($\mu_{dSSIM^{L^s}}$), and can be otherwise regarded as noise. Similarly, appropriately chosen values of ϕ_a, ϕ_b, and ϕ_c will also eliminate false positives related to infrequently occurring major body movement that are far away from the mean value ($\mu_{dSSIM^{L^s}}$), and can be otherwise regarded as outliers.

Typing Activity Detection. As the target user may type at specific instance(s) in time during the video call, it is necessary for the adversary to detect the time periods (or windows) where typing activity occurred. Typing activity detection is especially needed to effectively eliminate false positives during keystroke detection, which could otherwise result in incorrect word prediction results. We next outline a heuristic-based typing activity detection technique which employs our individual keystroke detection algorithm (Algorithm 2.1).

As outlined earlier, Algorithm 2.1 returns a set of frames where potential keystrokes could have occurred, but these detected potential keystrokes could also include non-typing activities (false positives). We leverage on a few intuitive heuristics in order to distinguish between the (detected) keystrokes that correspond to a typing activity from those that may correspond to non-typing activities similar to typing. The first heuristic, referred as *maximum speed filter*, filters out false positives from the detected keystrokes by observing the maximum rate at which these (potential) keystrokes are detected by Algorithm 2.1. Studies have shown that most users typically type at a rate of about 4 keystrokes per second, and that it is highly unlikely to come across a typing rate of 10 or more keystrokes per second [14]. Thus, the maximum speed filter will filter out (as false positives) from the detected keystroke frames those that correspond to a rate of 10 or more keystrokes per second, per hand.

The second heuristic, referred as *location filter*, filters out false positives by determining if both hands are on or near the keyboard. Here, the basic idea is to first create a set of reference frames (K) where the target user is most likely typing (i.e., hands on/near the keyboard), and then use these reference frames to determine (using optical flow [145]) if their hand(s) are at a significantly different position in the other remaining frames corresponding to potential keystrokes. Specifically, we create the reference set K by including all potential keystroke frames in each 2-second window, if and only if the window contains at least four detected potential keystrokes (both hands combined) with each hand contributing two or more potential keystrokes. This condition represents a very likely case of typing activity, and thus the user's hands being on or near the keyboard. We then use this reference set to filter out as false positive any potential keystroke frame that is not within an empirically evaluated optical flow distance threshold to any frame (of the corresponding hand) in the reference set K.

Similar to the maximum speed filter, the next heuristic we employ is the *minimum speed filter* which filters out detected potential keystrokes as false positives if they occur at a rate of one (or lower) keystroke per second, combined for both hands (i.e., representing highly unlikely typing rate). The fourth and final heuristic, referred as *exclusive hand filter*, attempts to detect one-handed non-typing activities (e.g., mouse clicks) that often get classified by Algorithm 2.1 as potential keystrokes. Specifically, in each 10-second window, the exclusive hand filter filters out 10 or more consecutive potential keystrokes with the same hand as false positives. Figure 2.7 outlines the order in which these heuristics are applied for filtering out false positives.

All potential keystrokes that are not filtered based on the above four heuristics represent typing activity and are used for word predictions. Figures 2.5 and 2.6 further elucidates the working of our heuristic-based approach by means of two real scenarios that we encountered during our experimentation. We present a comprehensive evaluation of its performance later in Section 2.10.

2.5.4 Word Prediction

We now describe how the adversary can infer words that were typed from the detected keystrokes, using two different groups of information. The first group of information is simply the number of keystrokes detected for a word, and the hand (left/right) which was used to conduct individual keystrokes of the word. Let us call this information as *keystroke information*. The second group of information is the magnitude and direction of body displacement, more specifically the arm displacement, between consecutive keystrokes of the word. Assuming that the target user typed on

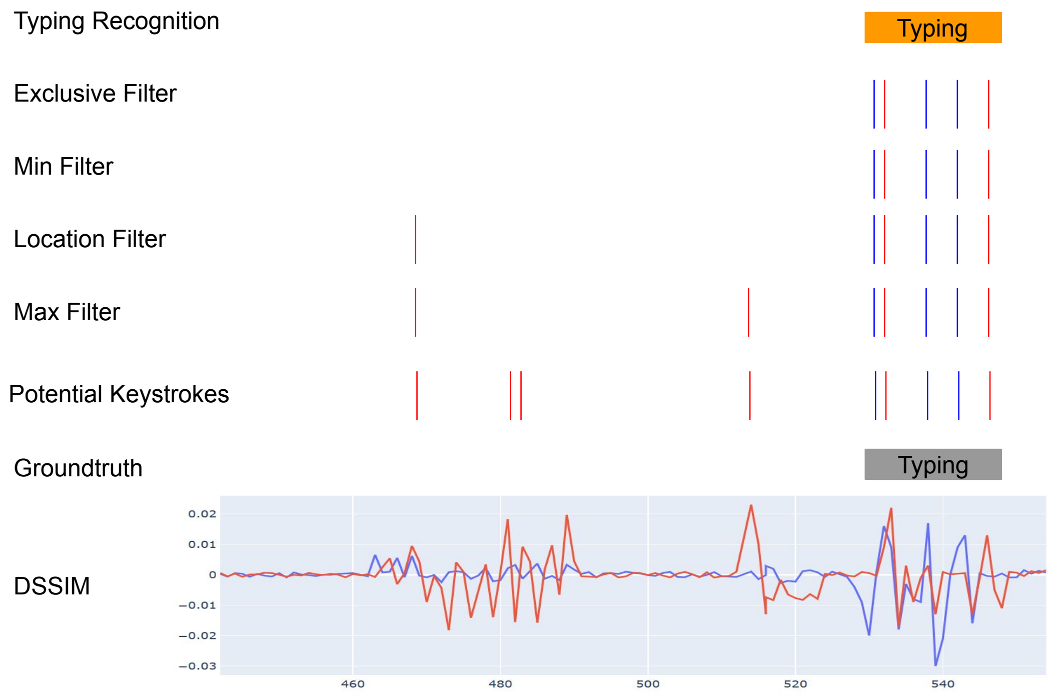


Figure 2.5: An example of our typing activity detection heuristics being applied on potential keystrokes, which resulted in a true positive. Red ticks and lines are for right hand, while blue lines and ticks are for left hand.

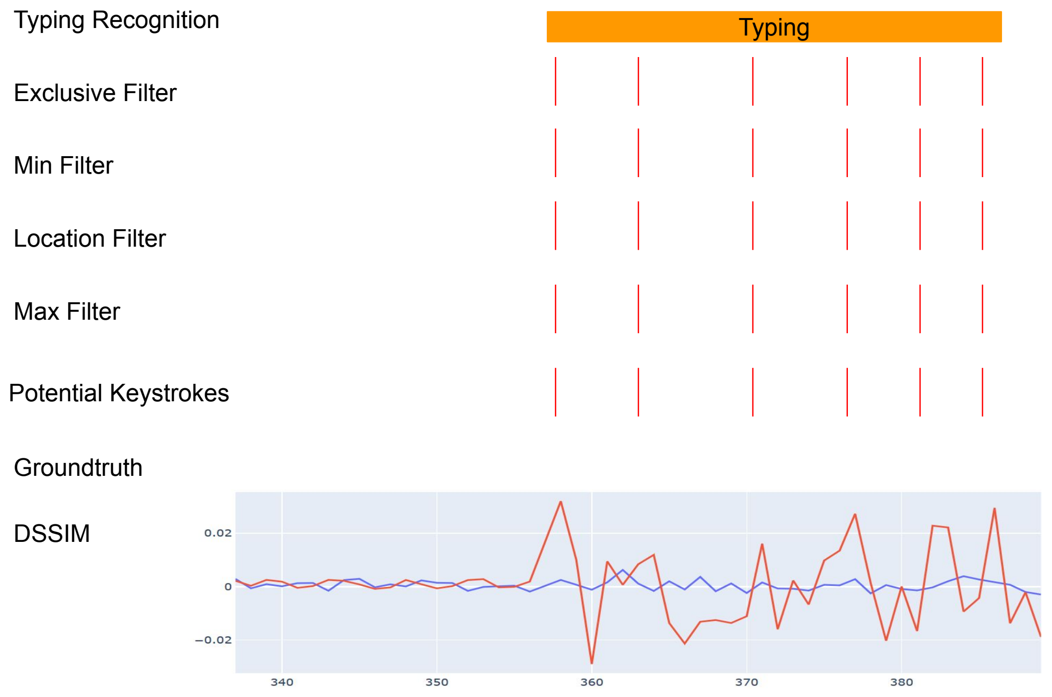


Figure 2.6: An example of our typing activity detection heuristics being applied on potential keystrokes, which resulted in a false positive. Red ticks and lines are for right hand, while blue lines and ticks are for left hand.

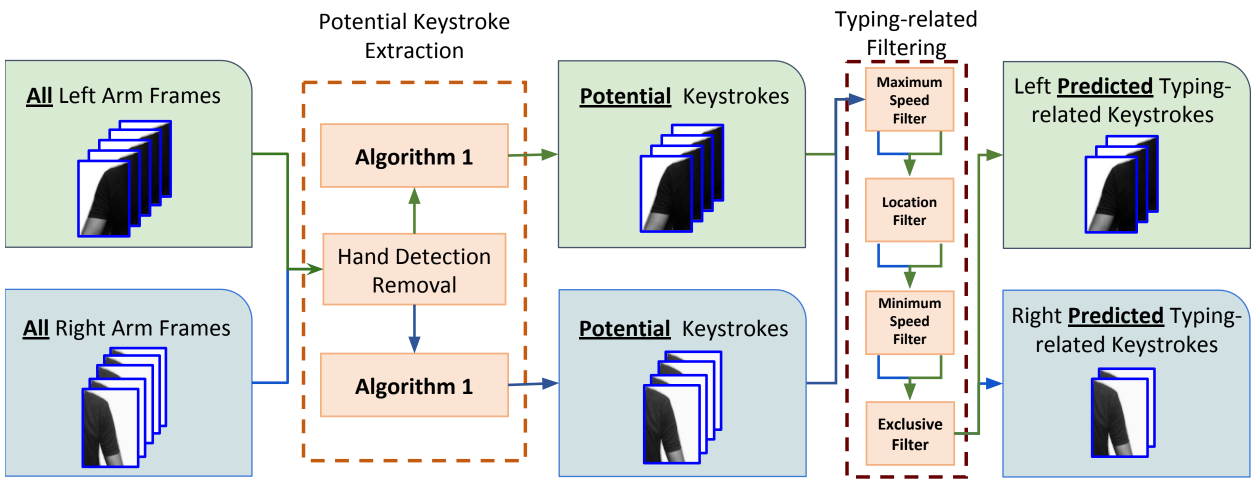


Figure 2.7: Overview of the typing activity detection technique.

a standard QWERTY keyboard, mapping the arm displacement between consecutive keystrokes to relative position of the keys can significantly improve the inference accuracy. Let us call this information as *displacement information*. After executing the typing activity detection (utilizing Algorithm 2.1) the adversary already has knowledge of the *keystroke information* – all the frame segments in which a keystroke was detected, separately for each hand. However, the *displacement information* is not readily available, and will require us to employ advanced computer vision techniques to effectively measure arm displacement between consecutive keystrokes.

Let $keystrokesFS^L$ denote the list of all detected left-hand keystroke frame segments and $keystrokesFS^R$ denote the list of the right-hand keystroke frame segments. $keystrokesFS^L$ and $keystrokesFS^R$ essentially constitute the *keystroke information*. We now describe how to utilize these two lists ($keystrokesFS^L$ and $keystrokesFS^R$) to derive the *displacement information*. In brief, we (i) identify the outer contour of individual arms in each keystroke frame segment, (ii) calculate the displacement of individual arms by tracking change in position of the outer edge of the arms across consecutive keystroke frame segments, and (iii) interpret calculated arm displacements with respect to the QWERTY keyboard layout. After obtaining the *displacement information*, we describe how the adversary can utilize both the *keystroke* and *displacement information* to carry out word predictions using a dictionary or reference database.

Outer Edge Detection of Arms. In order to efficiently measure arm displacement between con-

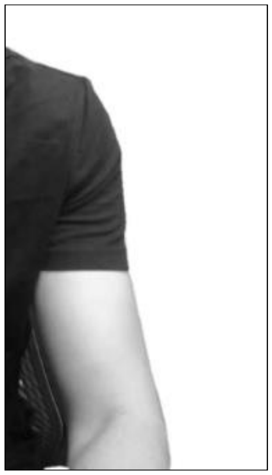


Figure 2.8: A keystroke frame segment.

secutive keystrokes, we focus on specific regions of the keystroke frame segments. Instead of trying to analyze movement of all pixels between two keystroke frame segments, we focus on pixels covering the outer-edge movements of the arms (Figure 2.8). The intuitive reasoning behind this design decision is that the characteristics of outer-edge movements are reflective of the movement of the entire upper arm and shoulder. Let us label the subset of pixels in a keystroke frame segment covering the outer contour/edge of the body as the *outer contour*, or OC. To compute the *outer contour* in each of the keystroke frame segments, we first detect all *edges* in a keystroke frame segment using Canny edge detection technique [85]. As the background in the frame segment is already removed during the keystroke detection step (Section 2.5.3), outer edges of the arm and shoulder are easily captured by the edge detection process. However, there is a possibility that edges within the arm and shoulder areas, such as creases or patterns on a shirt, could also get detected as an edge. To overcome this issue, we device a straightforward approach to remove all additional edges (i.e., all edges except the outer edge of the arm and shoulder), as described below. In case of the left hand, for each row of pixels we keep the rightmost pixel in the edge-detected frame segment that is part of an edge. The intuition is that in the absence of a background, the rightmost pixel in each row has to be part of the *outer contour*. Similarly, in case of the right hand, for each row of pixels we keep the leftmost pixel in the edge-detected frame segment that is part of an edge. An example of *outer contour* can be seen in (Figure 2.9).

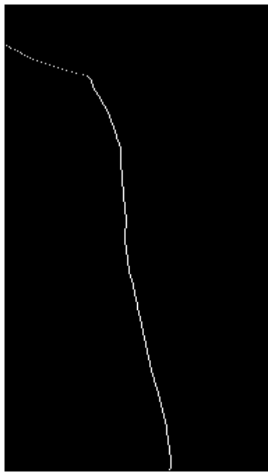


Figure 2.9: Outer contour (OC).

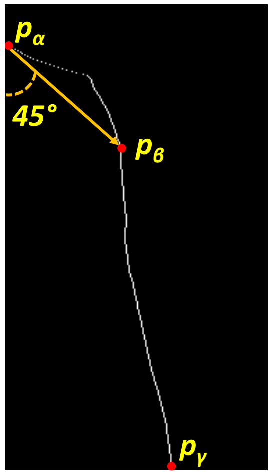


Figure 2.10: 45 degrees projection from p_α that intersects OC at p_β.

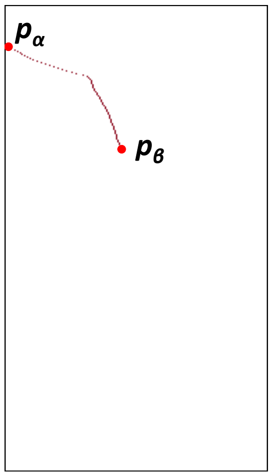


Figure 2.11: Shoulder contour (SC).

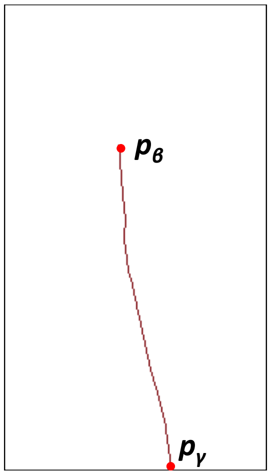


Figure 2.12: Arm contour (AC).

After the *outer contour* is computed for every keystroke frame segment in $keystrokesFS^L$ and $keystrokesFS^R$, we next segment the outer contour into *shoulder contour* (SC) and *arm contour* (AC) based on human physiology (xfig:arms). This physiology-based division is approximated by drawing a projection from the pixel nearest to the neck (p_α) such that the angle between this projection and the vertical boundary of the frame segment is 45 degrees (2.10). Let p_β be the pixel where this projection intersects the *outer contour*, and p_γ be the pixel farthest from the neck in the *outer contour*. Pixels in the *outer contour* between p_α and p_β becomes the *shoulder contour*, and pixels in the *outer contour* between p_β and p_γ becomes the *arm contour*. Obviously, this is just an approximate computation of *shoulder* and *arm contours* as the underlying physiological differences between person to person cannot be accurately modeled using available webcam video data. features-sc and xfeatures-hc shows *shoulder* and *arm contours*, respectively, computed for the *outer contour* example mentioned earlier. While the *arm contours* are directly useful in displacement calculations, explained next, *shoulder contours* are also utilized for calibrating the displacement calculations.

Displacement Calculations. We employ sparse optical flow technique [145] to quantify hand displacements between consecutive keystrokes. Sparse optical flow is a computer vision technique that takes a set of pixels (for example, constituting an object) within an image as input, and outputs a vector set representing the displacement of those pixels (and thus the object) in another image.

Sparse optical flow is especially useful to track object movements across chronological frames of a video. In our framework, we apply sparse optical flow to track the displacement of *shoulder* and *arm contours* across all consecutive keystroke frame segments, individually for each hand. For simplicity, we use the left hand to explain the use of sparse optical flow on two consecutive keystroke frame segments ($\in keystrokesFS^L$), say L_i^s and L_{i+1}^s, with respective *arm contours* AC_i and AC_{i+1}. By applying sparse optical flow between AC_i and AC_{i+1}, we obtain a set of displacement vectors representing the direction and magnitude of how each pixel in AC_i has shifted in AC_{i+1}. We then use this set of displacement vectors to calculate a mean displacement vector for the *arm contour*, and let us call it $\overrightarrow{oa_i}$. Calculated in a similar fashion, let us call the mean displacement vector for the *shoulder contour* as $\overrightarrow{os_i}$. In summary, a $\overrightarrow{os}$ represents the average movement of the shoulder between consecutive keystrokes, whereas a $\overrightarrow{oa}$ represents the average movement of the hand between consecutive keystrokes.

Our objective behind computing $\overrightarrow{os}$ and $\overrightarrow{oa}$ is to use these displacement vectors to determine the relative position of the keys corresponding to keystrokes. We carry out two additional operations using the displacement vectors $\overrightarrow{oa}$ and $\overrightarrow{os}$ in order to make our inference framework more generalizable. First, we observed that in certain scenarios the camera itself may move slightly, in addition to the arm. This can be prominently observed in the case of a laptop webcam, where a press on the laptop keyboard can result in a noticeable motion of the webcam which is generally located on top of the display. We solve this by applying sparse optical flow on the background during the pre-processing (Section 2.5.2), and negating the mean displacement vector of the background ($\overrightarrow{ob}$) from $\overrightarrow{oa}$ and $\overrightarrow{os}$. Second, we observed that in certain instances the typer changes her/his posture in between consecutive keystrokes, for example due to fatigue. To address this, we utilize the shoulder displacement ($\overrightarrow{os}$) as an approximation of posture changes, and subtract it from $\overrightarrow{oa}$. Combining both of these operations, we obtain $\overrightarrow{om} = \overrightarrow{oa} - \overrightarrow{os} - \overrightarrow{ob}$, where $\overrightarrow{om}$ represents the approximate average arm displacement, free of influence from posture or camera movements, that happened between consecutive keystrokes.

Interpreting Calculated Arm Displacements. From our keystroke detection (Section 2.5.3), we

Table 2.1: Classification of left arm displacements.

Conditions	**Direction**
$\overrightarrow{om_i(x)} \geq 0$ and $\overrightarrow{om_i(y)} \geq 0$	NW
$\overrightarrow{om_i(x)} \geq 0$ and $\overrightarrow{om_i(y)} \leq 0$	SW
$\overrightarrow{om_i(x)} \leq 0$ and $\overrightarrow{om_i(y)} \geq 0$	NE
$\overrightarrow{om_i(x)} \leq 0$ and $\overrightarrow{om_i(y)} \leq 0$	SE

Table 2.2: Classification of right arm displacements.

Conditions	**Direction**
$\overrightarrow{om_i(x)} \geq 0$ and $\overrightarrow{om_i(y)} \geq 0$	NE
$\overrightarrow{om_i(x)} \geq 0$ and $\overrightarrow{om_i(y)} \leq 0$	SE
$\overrightarrow{om_i(x)} \leq 0$ and $\overrightarrow{om_i(y)} \geq 0$	NW
$\overrightarrow{om_i(x)} \leq 0$ and $\overrightarrow{om_i(y)} \leq 0$	SW

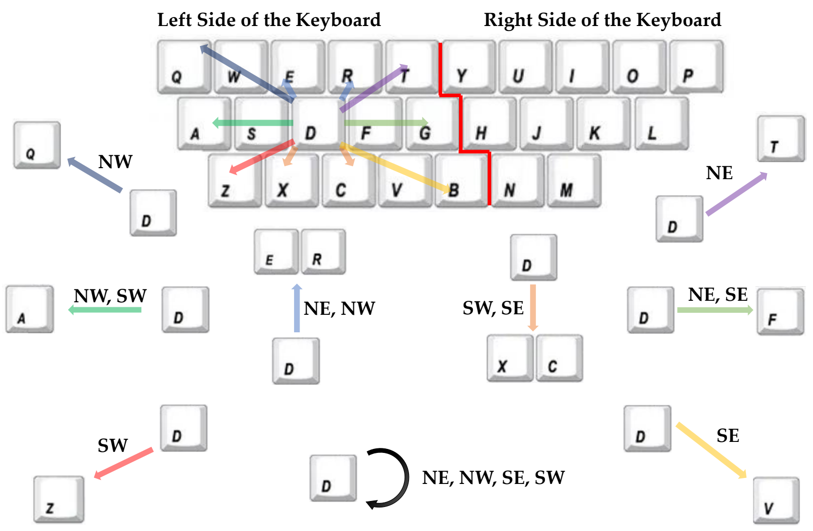


Figure 2.13: Splitting the keyboard into two halves, and the template inter-keystroke directions following the alphabet 'D'.

Table 2.3: Mapping of template inter-keystroke directions.

Relationship Between key_i and key_j	**Template Direction(s)**
key_i is the same key as key_j	NE, SE, NW, SW
key_i is in the same row of key_j and key_j is to the east of key_i	NE, SE
key_i is in the same row of key_j and key_j is to the west of key_i	NW, SW
key_i is in the row above of key_j and key_j vertically overlaps the key key_i	NE, NW
key_i is in the row below of key_j and key_j vertically overlaps the key key_i	SE, SW
key_i is in the row above of key_j, key_j does not vertically overlap the key key_i, and key_j is to the east of key_i	NE
key_i is in the row above of key_j, key_j does not vertically overlap the key key_i, and key_j is to the west of key_i	NW
key_i is in the row below of key_j, key_j does not vertically overlap the key key_i, and key_j is to the east of key_i	SE
key_i is in the row below of key_j, key_j does not vertically overlap the key key_i, and key_j is to the west of key_i	SW

are already aware of which hand was used to type individual letters. While this information alone can be very useful in conducting dictionary-based predictions, we deploy the arm displacement vector ($\overrightarrow{om}$) computed now to further reduce the search space. Reduction in the search space will in turn make our predictions more accurate.

Between any two consecutive keystrokes using the same hand, we classify the corresponding $\overrightarrow{om_i}$ into one of the four intercardinal directions: northeast (NE), northwest (NW), southeast (SE), southwest (SW). The classification of a left hand $\overrightarrow{om_i}$ is conducted as per conditions listed in Table 2.1. In Table 2.1, $\overrightarrow{om_i(x)}$ and $\overrightarrow{om_i(y)}$ are the x-axis and y-axis displacements (i.e., vector components), respectively, measured in pixels. The classification is isomorphic in case of right arm displacements between keystrokes, as listed in Table 2.2.

Template Inter-keystroke Directions. Now, we define template inter-keystroke directions on the standard QWERTY keyboard, which are the *ideal* directions a typer's hand should follow. To define the template inter-keystroke directions, we first divide the QWERTY keyboard into two halves (left and right). The left side of the keyboard contains the letters $\{q, w, e, r, t, a, s, d, f, g, z, x, c, v, b\}$ while the right side of the keyboard contains the letters $\{y, u, i, o, p, h, j, k, l, n, m\}$ as shown in Figure 2.13. Similar to prior works that used an analogous modeling [108, 112], we assume that a typer will predominantly type keys on the left side of the keyboard using her/his left hand, and vice versa. However, every key on the keyboard occupies a rectangular area, and a typer can have some variance in the position within each key where it is pressed. Some keys may be pressed in the center, while others could be pressed around the edges. This naturally occurring variance lead us to model the template inter-keystroke directions more flexibly using nine possible scenarios between any two keys key_i and key_j, as detailed in Table 2.3 and exemplified in Figure 2.13.

Word Inference. Our word prediction is a dictionary-based search for words based on (i) matching the order and number of left and right handed keystrokes, and (ii) matching the calculated direction of arm displacements with the template inter-keystroke directions. To satisfy the first criterion, a $word_i$ in the dictionary is deemed as a candidate for the typed word if $keystrokesFS^L$ and $keystrokesFS^R$ contain a combined number of keystroke frame segments equal to the length of

$word_i$. The keystroke frame segments in $keystrokesFS^L$ and $keystrokesFS^R$ should also be chronologically interleaved according to the alphabets in the left and right sides of the keyboard (Figure 2.13). To satisfy the second criterion, a $word_i$ in the dictionary is deemed as a candidate if the calculated arm displacements $\overrightarrow{om_j}$ between every letter of the $word_i$ satisfies the template mappings outlined in Table 2.3. We also sort the dictionary based on how frequently its words are used in the English literature (in descending order), so as to improve inference accuracy when there exists multiple candidate words that satisfy the above two criteria. In addition to the top prediction (i.e., the candidate word with the most usage in English literature), we also evaluate if the typed word is contained in top-k of such candidate words, as an adversary can run additional semantical and contextual analyses to improve inference of complete sentences. We, however, limit the scope of this work to only word inferences.

We next outline details of the different experimental setups and evaluation experiments that we conduct to evaluate our keystroke detection and word prediction framework. Our first set of evaluation experiments are conducted in a slightly constrained (or "*In-Lab*") setting to analyze the *best-case* performance of our framework. Our second set of experiments are conducted in a fully unrestricted (or "*At-Home*") setting to analyze the *worst-case* performance of our framework. All our participant recruitment and data collection experiments were approved by our university's Institutional Review Board (IRB).

2.6 In-Lab Experimental Setup

Our first set of evaluation experiments were conducted by fixing the video call setup, including, the device(s) used for the calls and participants' sitting position during the call, and text typed by the participants. For this set of experiments, which we refer as In-Lab setup, we recruited a diverse set of 20 human subject participants and collected video call data while they were performing typing tasks, details of which are outlined below.

Participant Demographics. Out of the 20 participants recruited for this setting, 9 are females and 11 are males. Based on a screening-survey, 4 participants conducted hunt-and-peck typing, 5 conducted touch typing, and the remaining 11 participants conducted hybrid typing. One of the participants identified as being left-handed while the remaining 19 participants identified themselves as right-handed.

Participant Tasks. Each participant completed six different sessions across different experimental parameters, which are listed below. Each session was conducted on a different day. Before every session, the experimental parameters were chosen randomly to cover different combinations, and the participant was informed about those parameters beforehand. The data collection sessions were conducted in a controlled setting inside a private office, primarily to limit noise in the audio data collected for equitable comparison (with an audio-only framework by Compagno et al. [77]). The participants were positioned in front of a computer with a display, keyboard, and a webcam directly facing them. Each participant was shown a random word on-screen in large font and the participant was instructed to naturally type the displayed word followed by a blank space. Upon entry of blank space, a new random word replaced the previous word on-screen, and the participant repeated this process for 300 words in each session. The random words were picked from a dictionary of 4000 most frequently used words (of 4 or more letters) in the English literature [40]. In order to minimize the impact of fatigue while typing, each session was divided in to three sub-sessions, each consisting of 100 words. Participants were encouraged to take a break between each of the sub-sessions. It should be noted that, despite the fixed nature of the In-Lab setup, participants were free to change their body posture and position based on their need and comfort-level, both during and in-between the typing sub-sessions.

Data Collected. On the data collection computer, our custom application recorded the webcam video (at $1920 \times 1080\, pixels$), microphone audio (at 44.1 kHz) and time-stamped ground-truth of the keys (characters) typed by the participant. The ground-truth information is used to measure the accuracy of our framework. To obtain realistic results, we later transmitted the recorded video over the Internet through different video calling software and captured it remotely on another computer. Skype [29] was used for majority of the evaluation, but we also compare it with Hangouts [10] and Zoom [43] in Section 2.7.2. The video transmission was achieved using `ManyCam`, a virtual webcam driver that can play pre-recorded videos during a video call. The remote capture of the transmitted video was done using `OBS Studio` [25].

Experimental Parameters. We evaluate our attack framework across a diverse set of experimental parameters to showcase its generalizability and practical impact. Below is a list of the different parameters that were studied:

(1) *Clothing*: Long-sleeves, Short-sleeves, Sleeveless.

(2) *Keyboard*: Logitech K120 (Wired), Anker A7721 (Bluetooth).

(3) *Webcam*: Anivia W8 (1080p), Logitech C920 (1080p).

(4) *Devices*: Lenovo 330-15IGM Laptop, Dell OptiPlex Desktop.

The laptop was evaluated with its built-in keyboard and webcam, whereas on the desktop we collected data using combinations of two different (external) webcams and keyboards. We instructed each participant to wear clothings such that they covered all three types (long-sleeve, short-sleeve, and sleeveless) over the six sessions. Overall, the combination of these parameters resulted in fifteen different experimental settings (within the In-Lab setup), three on the laptop and twelve on the desktop. For evaluating keystroke predictions, we used two different English dictionaries. One is a dictionary of 4K words which was the same dictionary used for data collection, and the other is a more comprehensive dictionary of 65K English words. It should be noted that we do *not* evaluate the typing activity detection technique in the In-Lab experiments, as the participants were not performing any other tasks besides typing.

Different Backgrounds and Removal. As we employ `DeepLabv3` for background removal, which has been extensively evaluated in the literature, we do not evaluate it as an experimental parameter. Nonetheless, in Figure 2.2 we show that `DeepLabv3` was able to remove backgrounds in different indoor and outdoor settings. Moreover, in the case that `DeepLabv3` fails to properly identify and remove a particular background in the recorded video, the adversary can easily substitute it for another background removal technique.

2.7 Video-only In-Lab Evaluation

In this section, we evaluate our prediction framework solely using the video data stream collected during In-Lab experiments. We briefly present results on the performance of our keystroke detection algorithm (Algorithm 2.1), before detailing the various prediction results.

2.7.1 Keystroke Detection Performance

We evaluate keystroke event detection using the *precision* and *recall* metrics, while also studying the effect of different coefficient values ϕ_a, ϕ_b, and ϕ_c used in our keystroke detection algorithm (Algorithm 2.1). As seen in Figure 2.14, recall increased as ϕ_a and ϕ_c were decreased, and when ϕ_b was increased. This is because when ϕ_a and ϕ_c are small and ϕ_b is large, our keystroke detection algorithm will even recognize minute noises as a keystroke event. Corresponding precision values are presented in Figure 2.14. Balancing between precision and recall is always a trade-off, and based on these empirical results we achieved a good precision-recall balance for the coefficient values $\phi_a = 1.5$, $\phi_b = 3$, and $\phi_c = 1.5$. Using these coefficient values, we obtained an average of 93% precision with 92% recall rate. Accordingly, we use keystrokes detected using these coefficient values for the rest of the evaluation, including the false positives and ignoring the false negatives.

2.7.2 Keystroke Prediction Performance

We now present results on word prediction performance for different experimental settings.

Different Webcams. Quality of the video can intuitively make a significant difference in the prediction accuracy, as low quality video frames are more likely to be erroneously processed by our algorithms. Accordingly, we look at the prediction accuracies obtained for the three experimental webcams (two external webcams, one built-in to the laptop). Both the Anivia and Logitech are able to capture videos at 1080p @ 30 fps, but the Anivia webcam features a wide-angle lens when compared to the Logitech webcam. The Lenovo laptop comes with a low-end webcam that can record video only at 720p @ 30 fps. As seen in Figure 2.15, the Lenovo laptop webcam con-

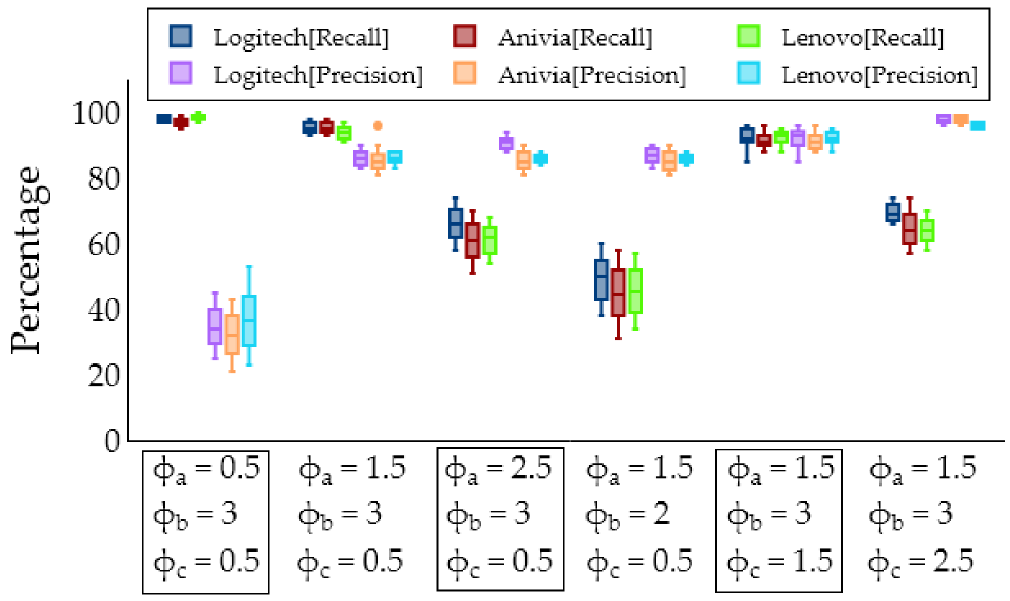


Figure 2.14: Precision and recall of keystroke detection under different ϕ_a, ϕ_b, and ϕ_c.

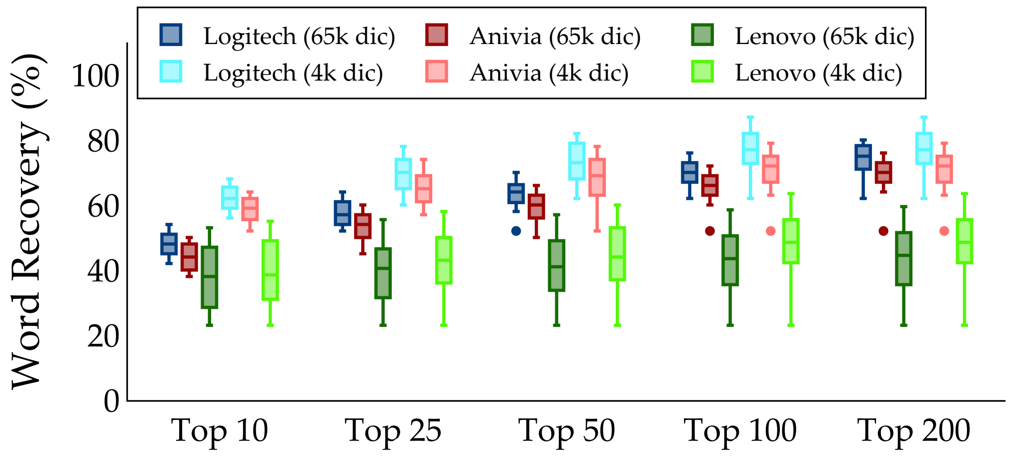


Figure 2.15: Successful word inference within top-k predicted words, for different webcams.

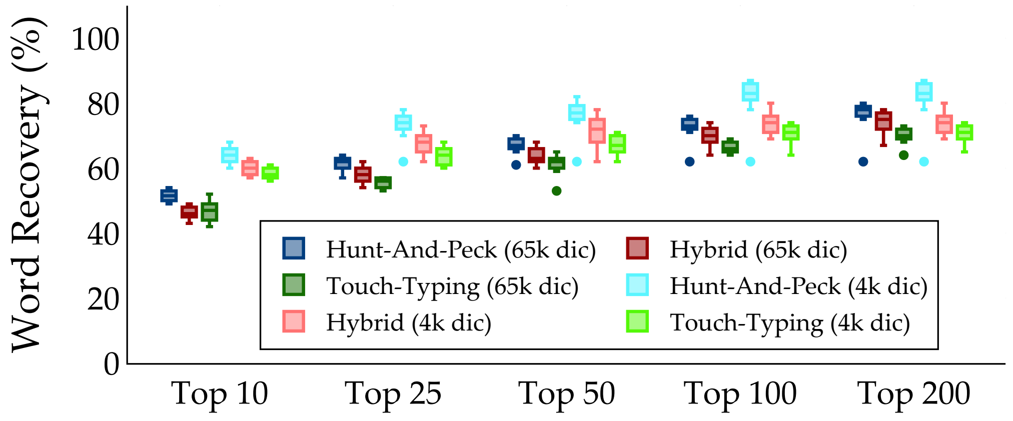


Figure 2.16: Successful word inference within top-k predicted words, for different typing styles (using Logitech webcam).

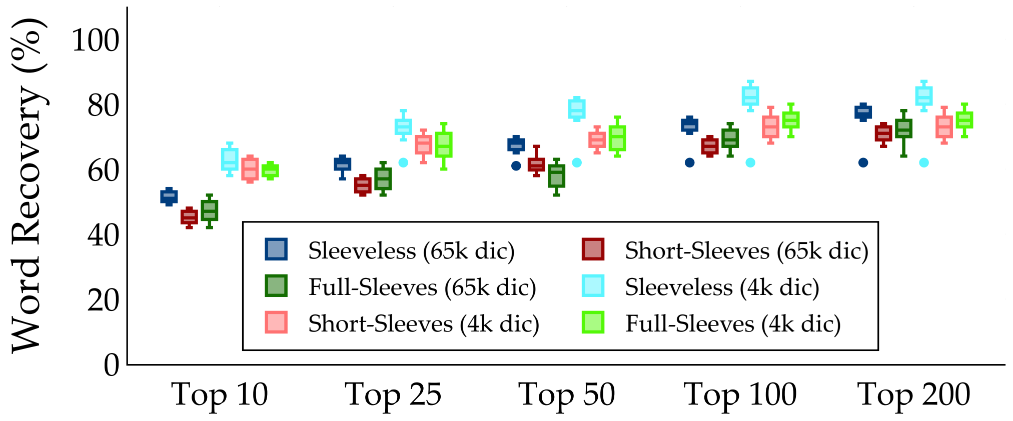


Figure 2.17: Successful word inference within top-k predicted words, for different clothings (using Logitech webcam).

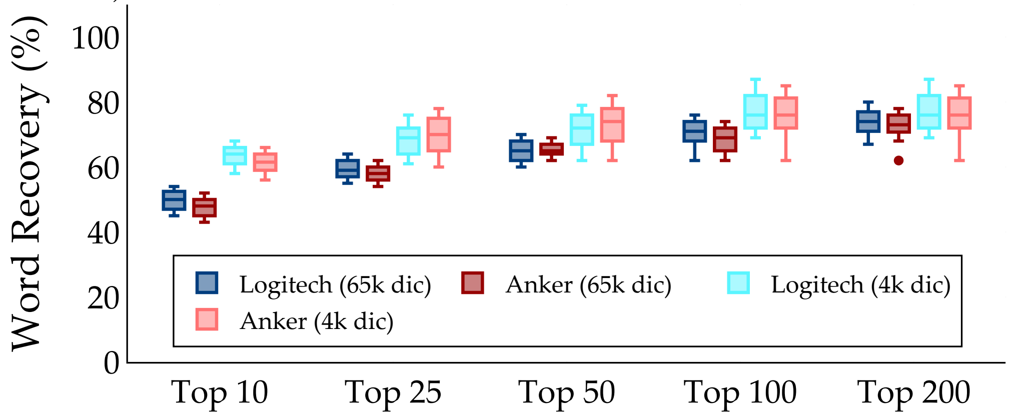


Figure 2.18: Successful word inference within top-k predicted words, for different keyboards (using Logitech webcam).

sistently had the worst performance compared to the Anivia and Logitech webcams. For the 65K dictionary, video from the Lenovo laptop webcam resulted in only 44.3% average word recovery when top-200 words were considered. The Logitech webcam performed slightly, but consistently, better than the Anivia webcam. Using the 4K dictionary, video from the Logitech webcam resulted in 75% average word recovery when top-200 words were considered, whereas video from the Anivia webcam resulted in 70% average word recovery. One of the reasons we speculate why the Anivia webcam did not perform as well as the Logitech webcam is because of its wide-angle lens. A wide-angle view means that the number of pixels capturing the user's body is reduced as more of the background is captured in the fixed video resolution. For many of the following evaluations, we used only the Logitech webcam for better understanding of other parameters.

Different Clothings. We next evaluate if different types of clothing, especially with respect to their

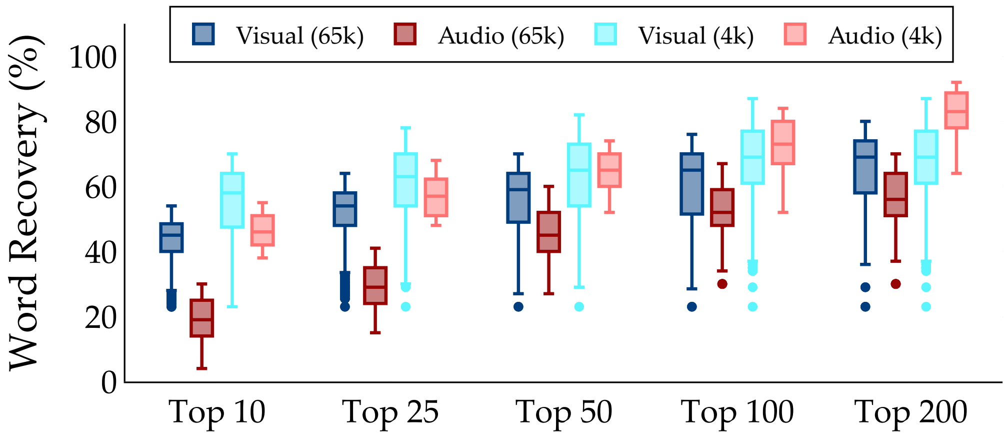


Figure 2.19: Comparison between audio and video inferences.

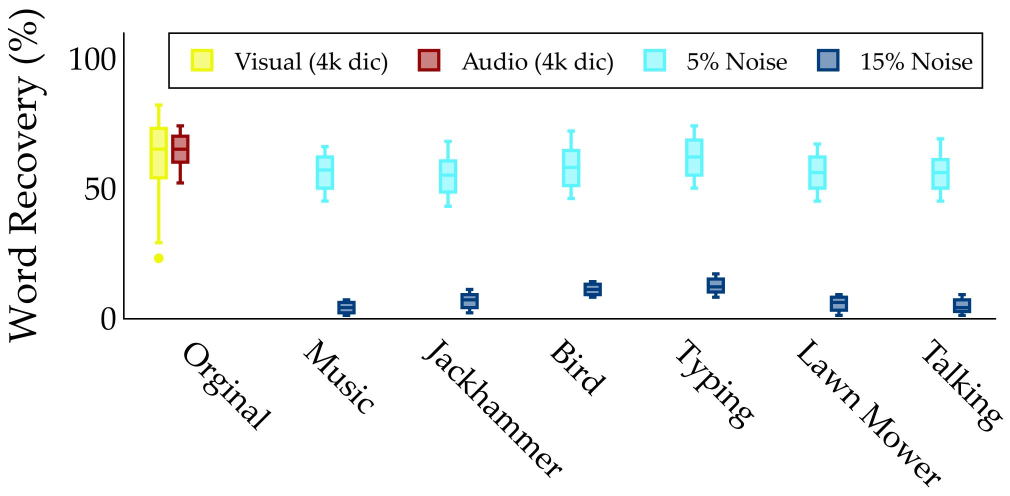


Figure 2.20: Comparison between audio inferences with various acoustic noises (top-50, 4K dictionary).

sleeve design, can affect our word prediction. In Figure 2.17 we observe that sleeveless typers were more susceptible to our attack, with 81.7% mean word recovery (top-200, 4K dictionary), compared to typers who wore either full or short sleeved dresses (74.4% and 73%, respectively). We speculate that both short and full sleeves can mask the extent to which the arms are actually displaced, underneath the clothing. As a result, our displacement vector calculations can get affected, resulting in slightly less accurate word predictions.

Different Keyboards. Size of the keyboard, and thus spacing between the keys, can have a significant influence on the arm displacements observed during typing. However, after evaluating the results from the two external keyboards, we did not find any significant difference between them (Figure 2.18). Even though the Logitech keyboard is significantly larger than the Anker keyboard, the percentage of successful word predictions were almost identical.

Different Video Calling Softwares. We tested our attack using three popular video calling softwares: Skype, Hangouts, and Zoom. Analyzing results using the 65K dictionary and top-50 predictions, we found that our evaluation using Skype was marginally better than using Zoom and Hangouts (+3.4% and +8% mean word recovery, respectively). The same set of videos were used with all the three video calling softwares, therefore the differences are purely due to factors beyond our control, such as the video compression technique used, network bandwidth utilized, and inconsistent latency in the video call.

2.8 Video vs. Audio In-Lab Evaluation

In this section, we compare our prediction framework (which is based on the video data) with Compagno et al.'s work [77], where they utilized the audio stream of a Skype call for keystroke inference. We utilized Compagno et al.'s implementation of an acoustic-based keystrokes inference framework designed to work over audio/video calling applications, and applied our audio data to train and test the inference model.

Audio vs. Visual Performance. With the 65K dictionary, our video-based inference was approximately +15% more successful than Compagno et al.'s audio-based inference (when using top-200 predictions) as shown in Figure 2.19. However, with the 4K dictionary, the audio-based inference was +10% more successful than the video-based inference (using top-200 predictions). The reason why Compagno et al.'s audio-based inference performs poorly for the larger dictionary is because the collision rate significantly increases with the size of the dictionary. Nonetheless, our audio data collection was very controlled, with no one talking and minimum ambient noise levels. A realistic audio/video call will at least have participants talking, which can significantly affect Compagno et al.'s audio-based inference framework. Accordingly, we next evaluate the impact of various types of noise on Compagno et al.'s audio-based inference.

Noisy Audio vs. Visual Performance. We *mixed* six different types of acoustic noises with our audio data: music, typing, lawnmower, bird chirps, jackhammer, and talking. Also, to mimic real-life background noises, we mixed the acoustic noise at only 5% and 15% levels after amplitude normalization. As speculated earlier, after adding only 5% noise the average word recovery dropped from 65% to about 56% (top-50, 4K dictionary) as shown in Figure 2.20. The variance in word recovery across the six different noise types was not very significant. Interestingly, after adding 15% noise the word recovery sharply dropped to about 7%. These results highlight how even minimal noise levels can significantly affect the audio-based keystroke inference framework. In contrast, our video-based inference framework is not at all affected by acoustic noises which is a common occurrence in audio-video calls.

2.9 At-Home Experimental Setup

To understand our inference framework's effectiveness in the wild, we next evaluate it outside of the lab environment. In this setting, participants were asked to use their own device (a laptop or desktop with a webcam) and setup (sitting position, clothing, background, and positioning of devices) for the video call, including location from where the call is done (e.g., their home). This allowed us to collect typing related video data for a diverse combination of devices and setups, already familiar to the participants and not constrained in any way. Using such "At-Home" experiments, we also expand our evaluation beyond just predicting dictated English words. Specifically, we analyze how our framework performs for the inference of user-chosen passwords, websites, and English words. As in a real setting, users are expected to involve themselves in other activities besides typing (e.g., web surfing, playing online games, etc.). This unconstrained At-Home setup allows us to thoroughly evaluate our typing activity detection technique outlined in Section 2.5.3.

Participant Demographics. We collected data from 10 participants for this in-home evaluation, whose ages ranged between 21 and 29 years. Out of the 10 participants, 3 are females, and 7 are males. Based on a screening-survey, 3 participants conducted hunt-and-peck typing, 5 conducted touch typing, and the remaining 2 participants conducted hybrid typing. 9 participants identified themselves as right-handed and 1 as ambidextrous. The average height of the participants is approximately 170 cm, with an average observed typing speed of approximately 3.7 keystrokes per second and typing accuracy (in relation to typographical errors) of approximately 86.7%.

Participant's Task. Participants were invited to join a (maximum) 30 minute Skype video call, using their own device and setup and from their own location of choice, where they had to sporadically (and at their own pace) type 10 email addresses, 10 usernames, 10 passwords, 10 websites, and 10 English words, in no particular order and frequency. The typing was performed in a pre-shared online spreadsheet, which was later used as the ground-truth of the typed text/information. The spreadsheet also automatically recorded edit timestamp for each cell in the spreadsheet, which is useful for evaluating the typing activity detection technique. To ensure participants covered a reasonable amount of time on non-typing activities, we asked participants to take at least three 1-minute breaks doing one of the following three activities: watch a YouTube video, read a Wikipedia article, or play a digital game on their computer that only requires a mouse to play. Participants had the liberty to take additional or longer breaks and/or do any other activity on their computer that does not require keyboard usage. Unlike in the in-lab experiments, participants were allowed to use backspace in case they wanted to rectify a typing error and were allowed to use a larger set of keys/characters on the keyboard for their typing tasks (alphabet keys, number keys directly above the alphabet keys, keys corresponding to “.", “-", and “@" characters, and the enter and backspace keys).

Data Collected. In addition to the ground-truth text and timestamp information contained in the online spreadsheet where participants typed, participant's Skype video was recorded remotely using `OBS Studio` [25]. After each typing experiment was completed, we also collected supplementary information from our participants related to their employed device and setup, as summarized in Table 2.4 and Section 2.10.1.

Webcam Hardware and Positioning. We observed that three of our 10 participants used an external webcam in a similar fashion as in the in-lab setting, placed approximately at eye-level and focused directly on the participant. However, the remaining 7 participants who participated using their laptops, the webcam angle and distance varied noticeably, as shown in Table 2.4 and Section 2.10.1. The native webcam resolution across participants also varied between 720p or 1080p.

2.10 At-Home Evaluation

In this section, we evaluate the performance of our proposed typing activity detection and keystroke (or text/word) prediction techniques using video call data collected from the At-Home experiments. During these experiments, we observed that one of the participant's hair completely obscured his/her shoulder area for the entire experiment's duration, thus making the corresponding video frames unusable within our framework. Due to these At-Home experiments' uncontrolled nature, this participant was not asked to re-position his/her hair or change his/her posture. This points to a limitation of our inference framework. However, this has already been highlighted in our assumed adversary model, where we clearly state that both shoulders and upper arms should be visible (to the adversary) in the recorded video. Thus, our presented evaluation results below are based only on data collected from the remaining 9 participants.

2.10.1 Typing Activity Detection Performance

For evaluation of our typing activity detection technique (Figure 2.7), we employ the same optimal values for parameters ϕ_a, ϕ_b, and ϕ_c as determined earlier in Section 2.7.1. Across all the 9 participants, our typing activity detection technique resulted (Figure 2.21) in an average of 40.22

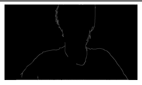
true positives, 12.4 false positives and 9.78 false negatives, for an average precision of 77.6% and recall of 80.4%. This shows that our proposed activity detection technique is fairly accurate enabling the adversary to not only detect a majority of the typing activity during a video call, but also successfully differentiate between typing versus non-typing activities.

In addition to the overall results, let's further highlight some interesting special cases. For participants using a laptop, we observed that the location filter (of the typing activity detection technique) was not very effective due to the proximity of the laptop's touchpad to its keyboard. Also, a few participants (at least, two) claimed to have used both their hands for interacting with the laptop touchpad, making the exclusive filter of the typing activity detection technique ineffective at times and resulted in a higher number of false positives. Significant movement and posture changes (between typing and non-typing activities) also resulted in degradation of detection accuracy, as was observed (Table 2.4 and Section 2.10.1) in the case of at least one participant whose left shoulder was not visible for a significant portion of the video call because of movement/posture changes. This limitation can also be attributed to the fact that while using a laptop, a user's position is a bit constrained (given the webcam's restricted field-of-view) and small movements/posture changes can result in the user's shoulders/upper arms becoming invisible/unavailable to the adversary.

2.10.2 Typing Accuracy

Before presenting our word prediction results, we briefly analyze the rate of typographical errors made by our participants. As our inference framework does not have provisions for handling rectifications made after typographical errors, participants' typing accuracies have a direct correlation to our framework's prediction error. Table 2.4 and Section 2.10.1 lists the typing accuracies of all our participants. As a case in point, participants I and C had the worst typing accuracy (73.1% and 49%, respectively), and our word prediction performance for participants I and C was also the lowest. The following prediction performance results are inclusive of all the typographical errors made by our participants in the At-Home setting.

Table 2.4: Participants (A-E) demographics and experimental settings for At-Home setup.

Participant	**A**	**B**	**C**	**D**	**E**
Gender	M	M	M	F	F
Age	21	24	24	28	29
Dominant Hand	Right	Right	Right	Right	Ambidextrous
Typing Style	Touch	Touch	Hunt-and-Peck	Hunt-and-Peck	Hybrid
Webcam	Dell XPS 15 7590	Mimoday S2	Mac Book Pro A1990	Lenovo Ideapad 5 15IIL05	Mac Book Pro A1398
Keyboard	Dell XPS 15 7590	Blackwidow Chroma RZ03-0122	Mac Book Pro A1990	Lenovo Ideapad 5 15IIL05	Mac Book Pro A1398
Sleeves	Short	Short	Short	Short	Short
Laptop/Desktop	Laptop	Desktop	Laptop	Laptop	Laptop
Height (cm)	180	168	170	168	152
Typing Speed (Keystrokes/Second)	8.83	3.67	2.75	3	3
Typing Accuracy (%)	96.9	93.3	73.1	90.5	98.5
Sample Typing Posture					

Participant	**F**	**G**	**H**	**I**	**J**
Gender	F	M	M	M	M
Age	23	28	21	22	25
Dominant Hand	Right	Right	Right	Right	Right
Typing Style	Touch	Touch	Hybrid	Hunt-and-Peck	Touch
Webcam	Logitech C922x pro stream webcam	Acer N19C3	Lenovo Ideapad 320-15abr	Lenovo Ideapad 320-15abr	Microsoft LifeCam
Keyboard	HP W2M75UA	Dell Wireless WK636p	Acer N19C3	Lenovo Ideapad 320-15abr	Lenovo Ideapad 320-15abr
Sleeves	Short	Long	Short	Short	Short
Laptop/Desktop	Laptop	Desktop	Laptop	Laptop	Laptop
Height (cm)	168	168	173	178	168
Typing Speed (Keystrokes/Second)	3.67	5.25	2.83	1.41	2.58
Typing Accuracy (%)	96.1	93.8	91.2	49	82
Sample Typing Posture					

2.10.3 Keystroke or Text Prediction Performance

Next, we present results for word prediction in the At-Home setting, separated based on the category of typed words. It should be noted that, in contrast to the In-Lab setting, participants in the At-Home setting typed their own words (for each of the five categories), and that four out of the five categories (of typed words) would most likely include words that would not be present in a typical English language dictionary. Thus, rather than using a standard English dictionary for prediction, we first create a ranked reference database of likely words in each category (which could be contextually created based on the target participant) and then employ it for the prediction task. As our framework predicts the possible combinations of typed characters based on the movements, and not individual characters themselves, such a reference database is *required* to complete the prediction task.

Websites. For the prediction of websites typed by target users, we created a reference database of 1 million most-visited websites [31]. In our dataset, all participants typed at least two websites ranked in the top-20 of the reference database, with an overall median rank of 140 and a mean rank of $36,745.3$ (mean is significantly higher than the median due to a few websites that are not popular and thus have very high ranks in the reference database). Our inference framework was successfully able to infer 66.7% of the websites typed by participants, within the top-25 predictions. An adversary may further reduce the search space based on contextual information about the target user.

Passwords. For prediction of passwords typed by target users, we created a reference database of 1 million most commonly used passwords [1]. Only 18.9% of the passwords were successfully recovered within top-50 predictions, which can be attributed to the fact that 74.4% of the passwords typed by our participants were not found in the reference database used for prediction. Considering only the passwords that were present in the reference database, 74% of them were successfully recovered within top-50 predictions.

English Words. For prediction of English words, we use the 65K-words dictionary used earlier for the In-Lab evaluation. Similar to passwords, not all (25.6%) of the words typed by our participants

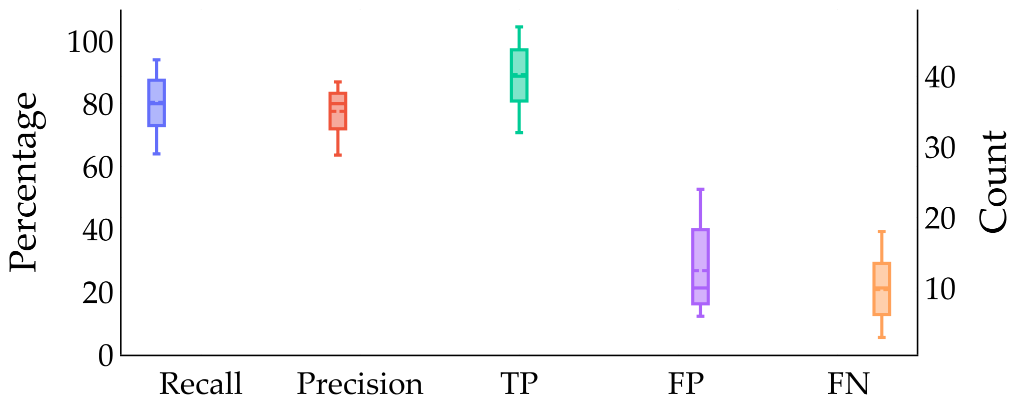


Figure 2.21: Performance of the typing activity detection technique.

existed in the dictionary used for prediction, and 21.1% of the English words were successfully recovered within top-50 predictions. One of the reasons our accuracy is worse than the In-Lab setting is because the reference dictionary's rank sorting is based on word-usage frequency in English language sentences, not based on random words produced by people. In other words, the ranking of words within the reference dictionary is not appropriate for prediction of randomly typed words. If an adversary could produce a more accurate contextual reference dictionary based on the target user, we expect better performance from our framework.

Usernames and Email Addresses. Usernames and email addresses are commonly used as an identifier for authentication, but they are also often publicly known and not sensitive information by themselves. However, knowing *when* a target user typed their username or email address can be valuable to an adversary, as a password is likely to be typed immediately afterwards during an authentication. Therefore, instead of predicting the usernames and email addresses typed by our participants, we try to predict when their known username and email address was typed by them. On average, we were able to correctly predicted when 91.1% of the usernames and 95.6% of the email addresses were typed.

2.11 Threat Mitigation

In this section, we outline and evaluate potential mitigation techniques to the video-based keystroke inference threat. We evaluate these mitigation measures by applying them to the In-Lab video dataset prior to using them in our keystroke inference framework, and then measuring the per-

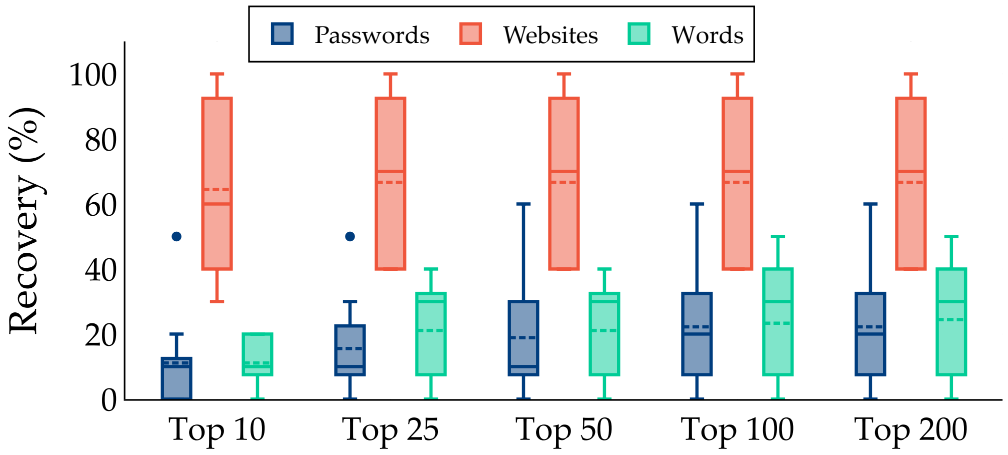


Figure 2.22: Successful inference of different text predictions.

formance of our framework on these modified video data. We evaluate the mitigation techniques using the In-Lab dataset instead of the At-Home dataset, because with its higher inference success, the In-Lab dataset can better illustrate the effectiveness of the proposed mitigation techniques. We measure the performance of our framework under the influence of these mitigation techniques using the metrics of (i) *effectiveness*, (ii) *efficiency*, and (iii) *video quality*, which we describe next followed by a description and evaluation of the mitigation techniques.

(1) **Effectiveness** measures the average reduction in word recovery due to the mitigation technique.

(2) **Efficiency** measures the average time to process each frame.

(3) **Video Quality** measures the image quality in the modified (edited) frames using *SSIM* index [156] as a measure of the structural quality of the frames within the video.

2.11.1 Mitigation Techniques

We now outline three frame manipulation strategies as mitigation techniques against the video-based keystroke inference threat presented earlier, and present performance results for them using the metrics defined above. It must be mentioned that, although these techniques can be applied to all the frames in the entire video call, it makes much more sense to apply them to frames in the vicinity of the target user's actual keystrokes. As keystroke detection for mitigation can be

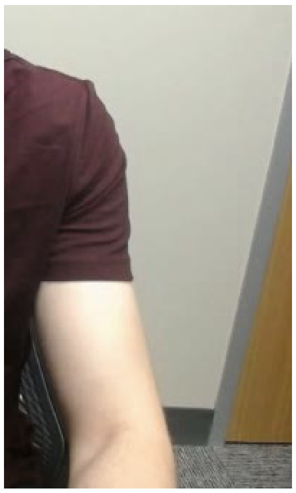


Figure 2.23: A left hand frame segment from a frame

easily accomplished using OS-interrupts on the user side, it should be relatively straightforward to identify frames just before, during and after the keystroke on which the proposed manipulation strategies should be applied. However, to effectively manipulate the frames immediately before a keystroke, we must maintain a buffer of those frames before they are transmitted out. Obviously, a large buffer can introduce significant latency in the video call, which is detrimental to the overall quality. We employ a buffer size of 2 frames (in a 30 fps video) for the first two mitigation techniques, and we use a variable buffer size in the third mitigation technique.

Blurring. The first approach is to manipulate (sensitive) frames using a Box blur approach [145]. This approach produces a *blurring* effect on the original frame by employing an adjustable kernel. The size of the Box blur kernel is chosen as some proportion (z_b) of the original frame size and populated with '1's. Once the kernel is fixed, blurring is done as follows: For each pixel, $p_{i,j}$, of the original image frame, the kernel is centered on that pixel and a new pixel value is computed. This new pixel value is the average of the neighboring pixel values weighted using the kernel. This new pixel value then replaces the original pixel $p_{i,j}$. This process is repeated for all the pixels of the frame. Some visual examples of the impact of blurring on a sample image frame for different values of the kernel parameter z_b are depicted in Figures 2.24 to 2.26. At the press of a keystroke we blur all the buffered frames and four following frames (total 6 frames) for a total duration of about 200 ms, which is the mean duration of keystrokes [72].

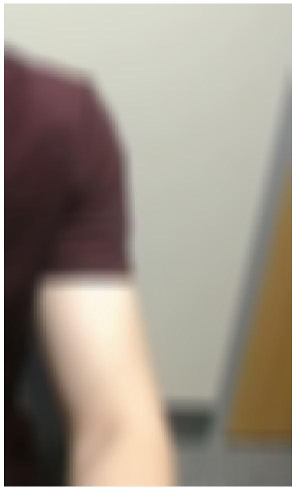


Figure 2.24: A left hand frame segment from a frame after blurring with $z_b = 5\%$

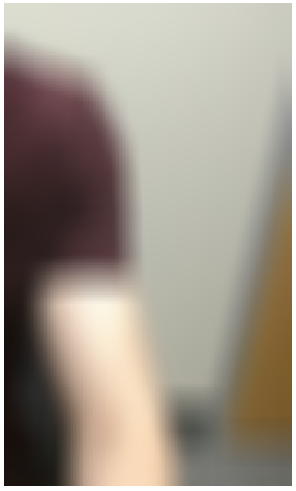


Figure 2.25: A left hand frame segment from a frame after blurring with $z_b = 10\%$

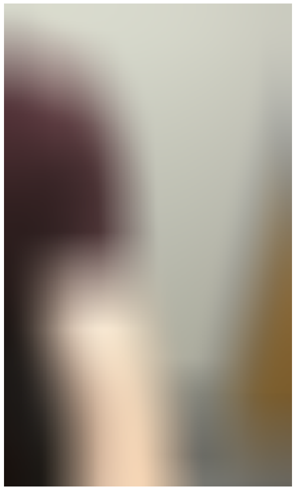


Figure 2.26: A left hand frame segment from a frame after blurring with $z_b = 20\%$

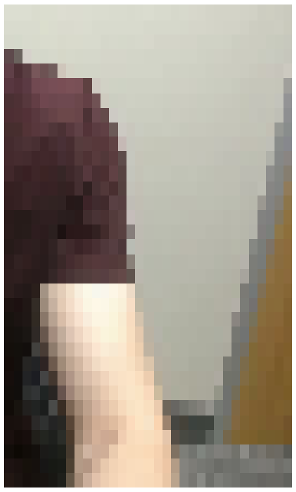


Figure 2.27: A left hand frame segment from a frame after pixelation with z_p = 3%

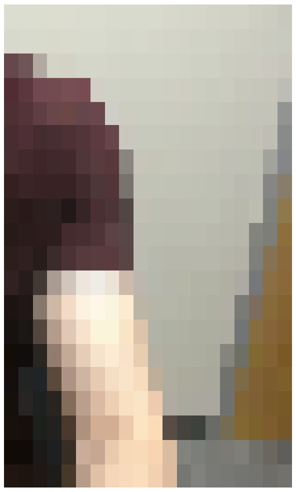


Figure 2.28: A left hand frame segment from a frame after pixelation with z_p = 5%

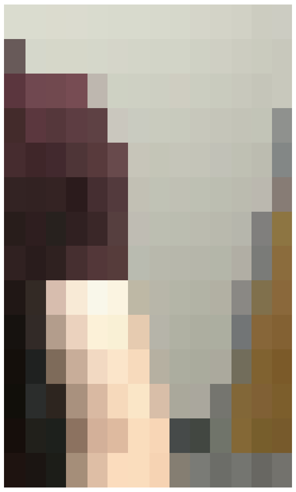


Figure 2.29: A left hand frame segment from a frame after pixelation with z_p = 7%

Our experimentation with using blurring within our inference framework shows that we are able to reduce the average word recovery from 65% to as low as 13% for z_b = 20% (Figure 2.30). In other words, we see a mitigation effectiveness of about -52% in top-50 prediction. We also observed that using higher z_b resulted in less words being recovered, as the frames were more blurry. For 1920×1080 sized frames, we observed that blurring takes around 17 ms per frame with a kernel factor (z_b) of 5%, on a laptop with an Intel i7-7700HQ (2.8 GHz) processor and 32 GB RAM. In terms of image quality, we saw an average *SSIM* index of 78.2% for z_b = 20%. A high *SSIM* index implies that the manipulated frame is similar to the original frame, and vice versa. These results show that blurring is an effective mitigation technique, which imposes little efficiency and quality overheads.

Pixelation. The second approach we analyze is pixelation, where the frame is first pixelated (partitioned) into areas defined by a proportion parameter z_p. In other words, the frame (of size $m \times n$) is partitioned into $1/{z_p}^2$ areas of size $\frac{m}{z_p} \times \frac{n}{z_p}$. Then, for each such area, the average of all pixel values within that area is computed, and each pixel $p_{i,j}$ within that area is reassigned this new average value. Some visual examples of the impact of pixelation on an image frame for different values of the pixelation proportion parameter z_p is shown in Figures 2.27 to 2.29. Similar to blurring, at the hit of a keystroke we pixelate all the buffered frames and four following frames (total 6 frames) for a total duration of about 200 ms.

Our experimentation with using pixelation within our inference framework shows that we are able to reduce the average word recovery from 65% to as low as 4.3% for z_p = 7% (Figure 2.30). In other words, we see a mitigation effectiveness of about -60% in top-50 prediction. We also observed that using higher z_p resulted in less words being recovered, as the frames were more pixelated. For 1920×1080 sized frames, we observed that pixelation takes around 1.41 ms per frame with a z_p of 3%, which is significantly faster than blurring. In terms of image quality, we saw an average *SSIM* index of 74% for z_p = 7%. These results show that pixelation is even more effective than blurring, and it imposes significantly lesser efficiency overhead, with a slight trade-off in quality.

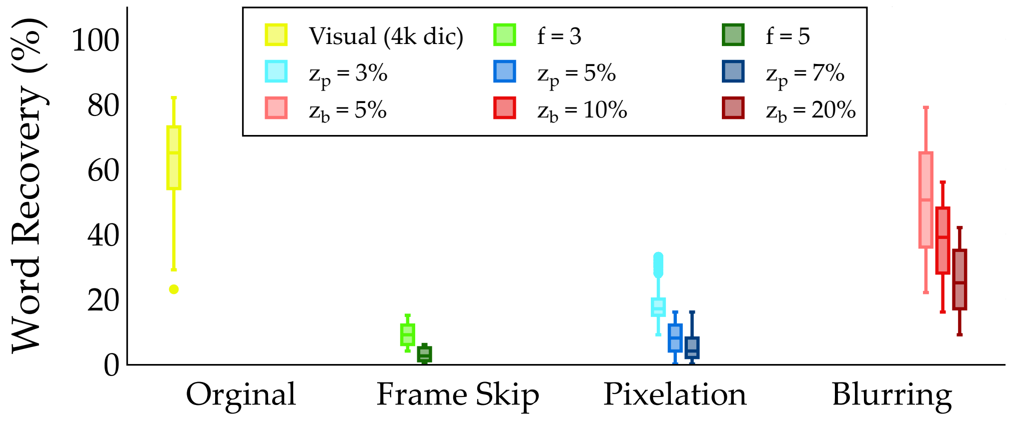


Figure 2.30: Different mitigations techniques and resultant word recovery (top-50, 4K dictionary).

Frame Skipping. The final mitigation approach we analyze is frame skipping, where as the name suggests, not all frames (captured during the video call on the target user side) are sent to the receiver (adversary). More specifically, the approach continuously buffers f frames during the call on the target user side. If typing is detected, all the buffered frames, as well as, an additional f frames after the detected key press, are dropped (i.e., not sent). Our experimentation shows that frame skipping is the most effective method in reducing word recovery rate, with only around 3% of the words recovered on an average (for $f = 5$). In other words, we see a mitigation effectiveness of about -62% in top-50 prediction. Frame skipping successfully eliminates all movement relationship between consecutive key strokes, resulting in such a high mitigation effectiveness. Frame skipping does not impact image quality of individual frames as the original frames are never modified. However, the downside of frame skipping is that user's video will appear to be stuck at a frame just prior to the keystroke, to the other participants in the video call. This can be confusing to the uninformed, but can be remedied with a notice such as *"John Smith is typing"* to other participants in the video call.

2.12 Discussion & Limitations

Generalizability. Let us comment on the generalizability and limitations of our study. First, we believe that our results are very generalizable to real-life scenarios based on the number of participants from which we collected data (more than prior related studies [57, 77]) and the different choices of webcams, keyboards, devices, participant clothings, and video calling software used in our experiments, which we believe are well representative samples. While all our participants were students recruited from a university campus, we observed a huge variety of different typing styles and quirks, which makes us reasonably confident about it being representative of the general population. Moreover, our data collection experiments were designed to reduce all types participant biases, including response bias, and were approved by the university's IRB.

Limitations. In our framework we only employed video feed to detect keystrokes, but video data can be combined with audio data from the call to further improve keystroke detection. The accuracy of our framework also relies significantly on the field of view containing the target user. Obstacles blocking (either completely or partially) the shoulder and arm areas of the target user, such as, microphones, headphone wires, or hair, could adversely affect both keystroke event detection and prediction. Similarly, if a camera's field of view does not fully or partially capture the shoulder and arm areas of the target user, as often observed in laptop webcams as they are generally set at an angle, it could also adversely impact the prediction performance of our framework. Lastly, we have also observed that significant ambient lighting changes (during typing) also disrupts the efficacy of our prediction. Many target user-specific factors can also disrupt the prediction performance of our framework, for instance if there are significant user movements while typing. This is possible especially if the target user is seated on a movable object, such as a rolling chair. As seen in our mitigation techniques, video quality is very impactful. If video frames are dropped, or the frames had some quality issues such as blurring or pixelation, then our framework will have poor inference accuracy.

2.13 Conclusion

We proposed and evaluated a keystroke inference framework which can predict text typed by a user during a video call. Specifically, we modeled and analyzed hand movements observable in the webcam's field of view, in order to detect keystroke events and then carry out a dictionary-based predictions. We evaluated our framework in a variety of controlled and uncontrolled scenarios, and were able to recover up to 75% words in some scenarios. We also proposed and evaluated three mitigation techniques which can effectively deter such keystroke inference attack in video calls.

2.14 Acknowledgement

Research reported in this publication was supported by the Division of Computer and Network Systems (CNS) of the National Science Foundation (NSF) under award number 1943351.

2.15 Publication

Research reported in this chapter has been published as "Mohd Sabra, Anindya Maiti, and Murtuza Jadliwala, "Zoom on the Keystrokes: Exploiting Video Calls for Keystroke Inference Attacks", Network & Distributed System Security Symposium (NDSS), 2021."

CHAPTER 3: PRIVACY OF CONTEXT

**This chapter has previously appeared in publication with IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2022. It was co-authored by Anindya Maiti, Ph.D., Murtuza Jadliwala, Ph.D. and has been reproduced here with minor revisions.*

3.1 Introduction

Next, we study the privacy of context during a video call, more specifically, how secure virtual backgrounds are. We propose a novel background reconstruction framework (Section 3.5) which attempts to reconstruct the real background in a video call that has a virtual background blended in to obscure the real background. Followed by that, we employ the real background (partially) reconstructed by this framework to design and evaluate four different privacy attacks (Section 3.6), namely, location inference, specific object tracking, generic object inference, and text inference attack. Then, by means of video call data collected from real human subject participants (in a variety of different settings and parameters) and pre-recorded videos collected in the wild (Section 3.7), we comprehensively evaluate the performance of the proposed real background reconstruction framework, and the efficacy of the privacy attacks that employ it (Section 3.8).

3.2 Related Work

Next, we briefly summarize some recent privacy threats targeting online video calls, followed by a discussion of more general threats which employ other visual (side)channels and how our proposed privacy attack contrasts with existing efforts in the literature.

Privacy Attacks using Online Video Call Data. As outlined earlier, privacy attacks in online video calls are either platform-specific and can be attributed to shortcomings in the target platform's software design and/or implementation, or platform-agnostic and employ audio/video call data as side-channels for inferring sensitive information. For the former case, we have seen instances of software vulnerabilities and configuration issues in the popular platform Zoom that have resulted in leakage of private information such as stolen account passwords [3] and meeting identifiers [45] resulting in unauthorized individuals joining (“Zoombombing") meetings with the aim of harassment and causing disruption. Similarly, another popular platform Webex has suffered from several vulnerabilities [5] which would allow attackers to covertly join meetings and access private information (e.g., name, email, IP address, device info) on meeting attendees without being admitted to the meeting. These platform-specific vulnerabilities have since been fixed, but new ones have continued to emerge. In the direction of platform-agnostic attacks, there have also been several investigative studies exploiting acoustic data and image frames from video calls as side-channels to infer sensitive information about the meeting/call participants. For instance, Sabra et al. [136] studied the feasibility of inferring a call participant's typed keystrokes by observing the target's fine-grained shoulder movements. Compagno et al. [77] conducted a similar investigation by employing key-press acoustics or audio data, instead of the call's video data. Recently, Nagaraja et al. [122] investigated the feasibility of inferring call locations from echo-reflection characteristics.

Privacy Attacks using other Visual Channels. Beyond the context of online video calls, private data inference attacks that employ other forms of visual channels is a well-researched domain. For instance, visual attack vectors such as a touchscreen's reflection on sunglasses [132], eyeball movements of a hunt-and-peck typer [75], visual tracking of fingers [63], and observation of motion of a tablet's backside [148] have all been utilized to infer typed text, passwords, PINs, and other sensitive information. Similarly, Backes et al. [61, 62] has proposed *Tempest* attack techniques which employ visual reflections to infer sensitive information, for example, reading the contents of a monitor from the reflections off a user's eyes or other seemingly benign objects such as teapots. Some other attacks in this broad direction also include inferring sensitive information from publicly-available images. For example, Shoshitaishvili et al. [139] devised a novel attack framework that, given a large corpus of pictures shared on a social network, automatically determines dating relationships, with reasonable accuracy.

The privacy attack proposed in this chapter lies in the former category, i.e., privacy attacks in online video calls. However, in contrast to earlier efforts, in this work we target a novel application feature (i.e., virtual background or background filter) and the sensitive information leaked by it (i.e., users' obfuscated backgrounds). As backgrounds in online video calls could reveal potentially sensitive information, including user-context, location and preferences, it has resulted in widespread adoption of background hiding/obfuscation features, such as, virtual backgrounds, by users. As a result, it has become all the more important to comprehensively evaluate the efficacy of such popular privacy-preserving features in one of the most widely used web applications (i.e., online video calling). This is what we aim to accomplish in this work.

3.3 Technical Background

Before describing details of our real background reconstruction framework and the related privacy attacks, we first outline the relevant technical concepts that are needed in its understanding.

Representation of Video Data. Any video stream data V is a time-ordered sequence $\{f^1, f^2, f^3, \ldots, f^l\}$ of image frames f^i, where l is the number of frames in the video. l typically depends on the length of the recorded video and the sampling rate of the recording device (hardware and software). Each frame f^i in a video stream V can be represented as an array of (m rows and n columns) of pixels, denoted as follows:

$$\forall f^i \in V = \begin{bmatrix} p^i_{1,1} & p^i_{1,2} & p^i_{1,3} & \cdots & p^i_{1,n} \\ p^i_{2,1} & p^i_{2,2} & p^i_{2,3} & \cdots & p^i_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p^i_{m,1} & p^i_{m,2} & p^i_{m,3} & \cdots & p^i_{m,n} \end{bmatrix}_{m \times n}$$

The size of the array (characterized by m and n) is indicative of the resolution of the image frames within the video, and is generally fixed for all the frames of the video. Each pixel $p^i_{u,w}$ of a frame f^i stores the color information at location (u, w), and the size of the pixel (in terms of number of bits) depends on the bit-depth used to store color information in the image. For example, for an image with a 24-bit depth (also referred as Truecolor), $p^i_{u,w}$ is a 24-bit value, wherein each of the three 8-bit blocks represent the intensity of the red, green, and blue primary colors, respectively.

Virtual Background Feature. Although the exact technique for applying virtual backgrounds to an ongoing video call in commercial video calling/conferencing applications is proprietary and unknown, we have determined the general underlying principles to be as follows. The virtual background feature in video calling applications attempts to replace/overlay the background pixels of each image frame f^i in the video call V with the corresponding pixels of a virtual image, denoted by VI. In this process, the first step is to generate a background mask for each frame, denoted as BM^i, which identifies the regions of the frame where the virtual background should be applied. More formally, a mask BM^i is a bitmap with the same resolution as the target image frame f^i, where non-zero pixel values represent the foreground (such as a human face or body), while zero pixel values represents the background in the video call. Often a *binary mask* is used for this purpose, which for a frame f^i of size $m \times n$ is a $m \times n$ bitmap, where each pixel (u, w) of the mask having a value of either $\langle 255, 255, 255 \rangle$ (indicating that the corresponding pixel $p^i_{u,w}$ of f^i is part of the foreground) or $\langle 0, 0, 0 \rangle$ (indicating that the corresponding pixel $p^i_{u,w}$ of f^i is part of the background). Similarly, a *trimap* mask has three states where an intermediate value of $\langle 128, 128, 128 \rangle$ implies that a pixel $p^i_{u,w}$ can either be part of the foreground or background. Mask generation is usually done by means of custom supervised machine learning models, which are capable of detecting and separating foreground objects (in image frames) from background objects. Moreover, mask generation could either be done independently across frames or based on previously observed frame(s) in order to reduce (classification) errors and abnormalities. Mask generation (to separate the foreground from background) is obviously not a perfect process, and depending on factors such as presence of certain objects in the frame or movement of users/objects across frames, parts of the background could be classified as foreground by the mask and “leaked" in the masked frame. This part of the real background for a frame f^i that is “leaked" or not obscured by the mask is also denoted by LB^i.

After the background mask is generated, a *blend*ing function is used to combine appropriate portions of virtual image VI, called the virtual background image and denoted as VB^i, with the image frame f^i using the background mask (BM^i) generated in the previous step. Typically,

Figure 3.1: A zoomed region of the real background

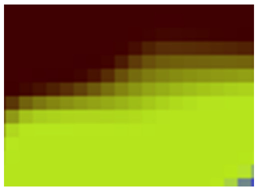


Figure 3.2: The zoomed region (Figure 3.1) after applying virtual background with blending.

bending is used to increase the quality of the video frames by reducing sharp contrasts between the detected foreground objects in the original frame and the virtual background image. Some state-of-the-art blending techniques that could be employed for this purposed include alpha blending, Gaussian blending, and Laplacian pyramid blending [66, 69, 73]. However, the blending function used by popular video calling applications is unknown (to us), and the type of blending function used could also depend on the generated mask. One side-effect of the blending operation is that it creates small regions in the output frames (near the foreground-virtual background edges) such that pixel values in these regions are mixture of both f^i and VB^i, as depicted in Figure 3.3.

Four Conceptual Components of Virtual Frames. Consider a video call V with a virtual image (VI) blended into the frames of V, as described earlier and shown in Figure 3.4. Each (blended) frame f^i of V is composed of four non-overlapping components (represented as bitmaps): the video caller VC^i, the leaked background LB^i, the blended pixels BB^i due to the blending function, and the virtual background VB^i. As such, every pixel in a frame can be defined as a combination

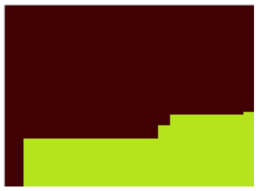


Figure 3.3: The zoomed region (Figure 3.1) after applying virtual background without blending.

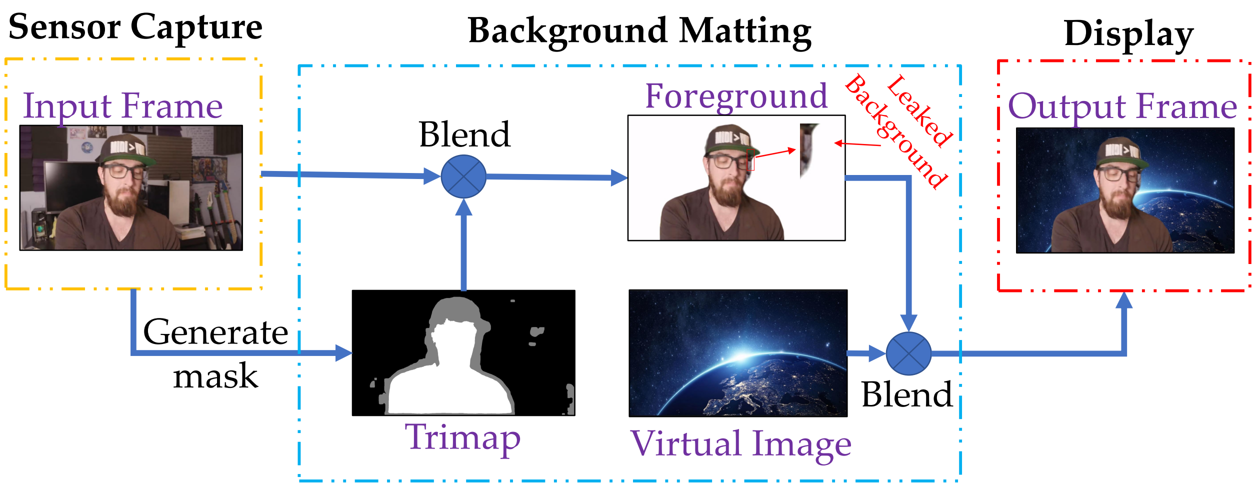


Figure 3.4: An example of virtual background used in a video call.

of these four bitmaps, as shown in Figure 3.10. Later in this chapter, we will attempt to infer the leaked background (LB^i) in a blended frame by reconstructing the other three components. Specifically, reconstructed VB^i, BB^i, and VC^i are removed from the original blended frame f^i, and any residue pixels not removed from f^i are assumed to be the leaked part (LB^i) of the real background.

3.4 Adversary Model

The goal of an adversary in our setup is to infer as much information about the real background as possible from the virtual image blended video call frames, produced by the virtual background feature as discussion in Section 3.3. The leaked (real) background information in this fashion can be used to infer sensitive information such as: (i) location of the target user during the call (for example, if the target user is at home, office, or another location), and (ii) objects present in the background (for example, books or signs/posters indicating political/religious affiliations). This could result in a significant privacy loss for the target user, despite using a virtual background feature to hide such information from being revealed to others on a video call, thus rendering it ineffective and giving the user a false sense of privacy protection.

Specific to our experimental setup, we assume that the webcam employed by the target user is stationary during the video call, and that the adversary is able to save the video stream data (V)

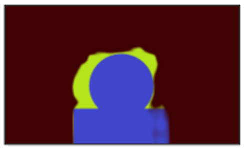


Figure 3.5: f^i

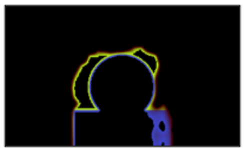


Figure 3.6: BB^i

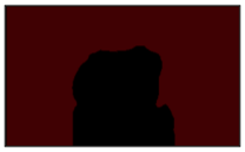


Figure 3.7: VB^i

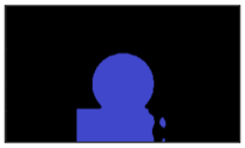


Figure 3.8: VC^i

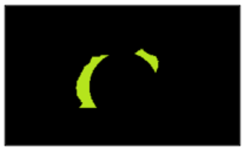


Figure 3.9: LB^i

Figure 3.10: The color green represents the leaked background, brown represents the virtual background, and blue is the video caller.

for post-processing. Additionally, we assume that there is at most only one person in the recorded video stream. The adversary can execute this privacy threat by either participating in a video call as an authorized participant and directly recording the video data corresponding to the target user, or by gaining access to a recording of the call from a private or publicly-available archive [26, 36, 42] after the call. We will discuss additional setting- or scenario-dependent adversarial assumptions as we describe our inference framework in the following sections.

3.5 Real Background Reconstruction Framework

We now present the design of our reconstruction framework, which takes as input a sequence of virtual image blended video frames V, and outputs a reconstruction of the real background by combining leaked background parts LB^i across all the blended frames f^i in the video V.

3.5.1 Overview

Given V, our framework applies the following process on each frame $f^i \in V$: (i) determine the virtual background region (or bitmap) VB^i, (ii) determine the blended region BB^i, and (iii) determine the video caller VC^i. Once these three components are identified, any leaked background regions LB^i in the frame f^i are calculated subtracting all the regions from f^i by VC^i, BB^i, and VB^i . After all frames in $f^i \in V$ are processed in this fashion, our framework combines all LB^is ($\forall i$) to reconstruct (parts of) the real background obfuscated by the virtual background feature. The overall design of our proposed reconstruction framework is shown in Figure 3.11.

3.5.2 Virtual Background Masking

The first step in our framework is to identify the virtual background region VB^i using *virtual background masking*. We consider two different scenarios. First, where the adversary has the virtual image VI employed by the target user, and can use it as a direct mask. This is an effective approach when a built-in and/or popular image is used as a virtual image. The second scenario is where the adversary does not have the virtual image employed by the target user, and must

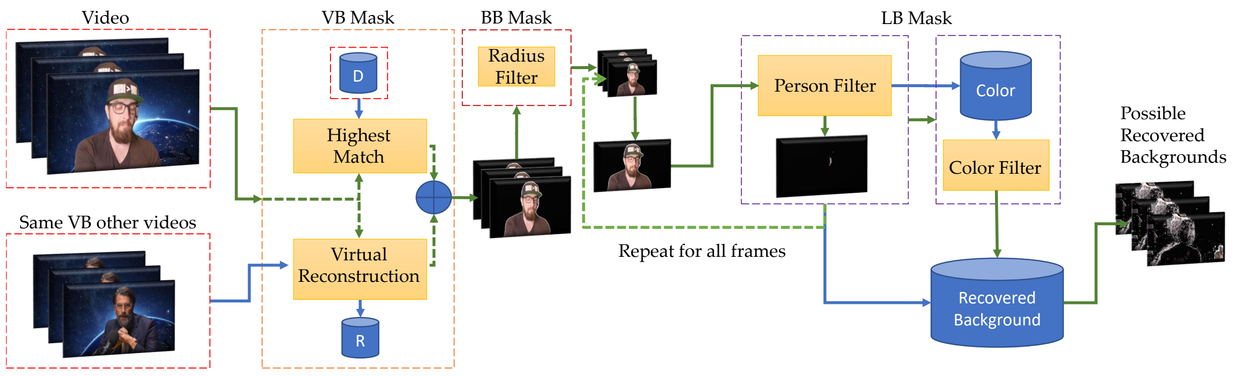


Figure 3.11: Proposed real background recovery framework. Green arrows show the flow.

reconstruct it before applying it as a mask. Similar scenarios should also be considered if the virtual background feature employs virtual videos, instead of static virtual images, for concealing the real background.

Identifying Known Virtual Image. This is the more straightforward scenario for virtual background masking, where the adversary can simply compare pixel-level similarity with a dataset ($D_{img} = \{img_1, img_2, \ldots\}$) of default/popular virtual background images that are potential candidates for the target user's employed virtual background. For this we employ an highest-likelihood estimator that, for each blended video frame f^i, does pixel-level matching with each image in the above dataset, and is defined as follows:

$$img_{actual} = \max_{\forall img \in D_{img}} \sum_{u=1}^{m} \sum_{w=1}^{n} \mu(img_{u,w} \oplus f^i_{u,w})$$

where μ is a matching function,

$$\mu(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases}$$

Identifying Known Virtual Video Frame. In this case, we assume a dataset of known videos, $D_{vid} = \{vid_1, vid_2, \ldots\}$. Please note that as a video is a sequence of frames, the above set is nothing but a set of frames. Now, when processing a blended frame f^i in V, it simply needs to be determined which frame (of which video) in the above dataset is employed as the virtual background. In order to accomplish this, we extend the previous highest likelihood estimator as shown below by considering all frames in all default/popular videos in the dataset:

$$frame_{actual} = \max_{\forall frame \in D_{vid}} \sum_{u=1}^{m} \sum_{w=1}^{n} \mu(frame_{u,w} \oplus f^{i}_{u,w})$$

Using Unknown Virtual Image. In this case, the adversary does not have the raw virtual image used by the video caller. For this we design the framework to recreate the potential image used as the virtual background, from the video V itself. For this we utilize the observation that only pixels belonging to the virtual background would stay the same (static) across all the frames of V, whereas the pixels representing the user (caller) and blended regions around the user would be more dynamic. We employ a threshold-based approach, wherein any pixel with a consistent value across a large number of frames in the video V would be considered as part of the virtual background image. Empirically, we found that for a standard 30 fps video stream, a pixel consistent across 10 or more frames has very high probability of belonging to the virtual background. We employ this threshold in our evaluation. This virtual image derivation approach may work similarly well in case the user is fairly stationary during the call, which will not reveal the regions of the virtual image where the user is positioned. This problem can be mitigated by the adversary by searching for the unknown virtual image in other call videos (used by the same user or other users), and then used them during the virtual image derivation process (Figure 3.11).

Using Unknown Virtual Video Frame. Similar to the previous scenario, in this case the adversary does not have the raw virtual video (frames) used by the video caller as the virtual background. As the unknown video has multiple frames, this makes it even more challenging to identify the exact background video frame that should be used (to compare) with a corresponding frame in the target video call. We utilize the fact that the virtual video loops repeatedly, and use it to derive all the frames of the virtual video using information from every periodic occurrence of each frame (pixels stay the same across every occurrence of a frame). Once the virtual video has been derived, we again employ a pixel-level similarity matching to identify a frame in the video call (V) containing the exact frame from the virtual video. This virtual video derivation may also not be completely accurate, especially if the user is fairly stationary during the call. Similar to the unknown virtual image, we can employ the virtual video used by another user (or in another video call by the same user) for better reconstruction (Figure 3.11).

Generating Virtual Background Mask. After either a known virtual image/frame or a reconstructed image/frame (denoted as M^i) is identified, a binary virtual background mask (VBM) is generated at the pixel-level using the following function:

$$VBM^i_{u,w} = \begin{cases} 1 & \text{if}\, \mu(M^i \oplus f^i_{u,w}) = 0 \\ 0 & \text{if}\, \mu(M^i \oplus f^i_{u,w}) = 1 \end{cases}$$

We can now use $VBM^i_{u,w}$ to obtain VB^i (from f^i) as follows:

$$VB^i_{u,w} = \begin{cases} f^i_{u,w} & \text{if}\, VBM^i_{u,w} = 1 \\ 0 & \text{if}\, VBM^i_{u,w} = 0 \end{cases}$$

Later in the chapter, we visually show VB^i being removed from f^i by depicting the corresponding pixels as black.

3.5.3 Blending Blur Masking

The next step in our framework is to approximately identify the blending blur (BB^i) region $\forall i$, i.e., for all the blended frames $f^i \in V$. The BB^i region resides between virtual background region (VB^i) and foreground pixels (VC^i plus LB^i). And, leaked background pixels (LB^i) are usually found between BB^i and VC^i as the video calling software mistakes parts of the background as being a part of the video caller (user). The separation between the video caller (user) and any leaked real background is not accounted here, and will be recovered in the next phase as part of the foreground.

To recover $BB^i_{u,w}$ we check all pixels within a radius ϕ for every pixel in the $VBM^i = 1$. In other words, $\forall u, w$ where $VBM^i_{u,w} = 1$, we calculate $BBM^i_{p,q} = 1 \oplus VBM^i_{p,q}$, wherein we maintain heuristically derived constraints such as:

$$\sqrt[2]{(p-u)^2 + (q-u)^2} \leq \phi,$$

$$1 \leq p \leq m, \;\; 1 \leq q \leq n$$

We can now use $BBM^i_{u,w}$ to obtain BB^i (from f^i) as follows:

$$BB^i_{u,w} = \begin{cases} f^i_{u,w} & \text{if } BBM^i_{u,w} = 1 \\ 0 & \text{if } BBM^i_{u,w} = 0 \end{cases}$$

3.5.4 Video Caller Masking

We observed a few key characteristics in the video calls pertaining to the video caller (or user), which we take in to account for calculating the video caller mask (VCM^i):

Inaccurate Human Boundaries. It is repeatedly observed that regions under the head, near the hair, between fingers, and other places around the video caller (or user) contains a leakage portion of the real background. This type of leakage contributes significantly towards our real background inference process.

Initial Leakage. We also observed that when a video call starts, the accuracy of a video calling software in concealing the real background is often poor. The accuracy improves after a few frames, when the video calling software is better able to track movements of the video caller, as shown in Figure 3.12. This type of leakage also contributes significantly towards our real background inference process.

Color Analysis. Whenever a pixel from the real background is leaked, it would fairly maintain the same color across different frames when it was leaked. In contrast, the pixels depicting the boundaries of the video caller would have slight variation is color as they relatively move throughout progressing frames. This can also be amplified by factors such as patterns on their cloths. We utilize this observation to refine our video caller mask (VCM^i).

In order to accurately generate VCM^i we apply a state-of-the-art segmentation technique using `DeepLabv3` [74], which employs the *Atrous Convolution* with upsampled filters to extract dense

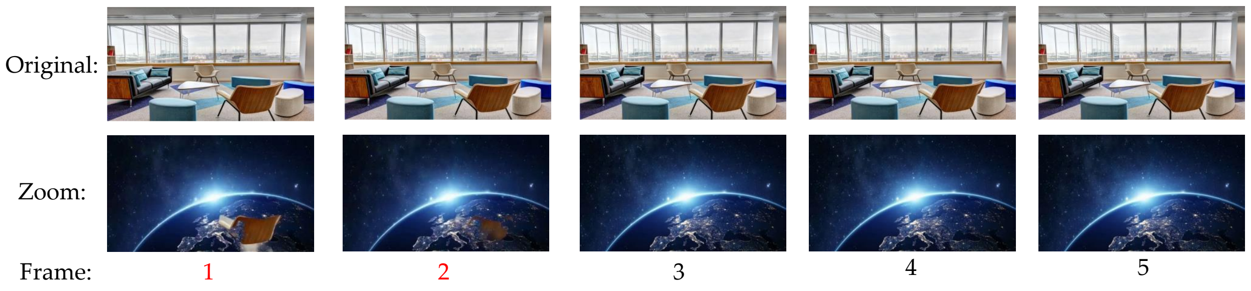


Figure 3.12: Example of leaked background components in the initial frames of a video call.

feature maps and to capture long range context. While this technique cannot be applied in real-time video calls due to computational challenges, an attacker can certainly use it for post-processing of a recorded video call. The output of this process is a video caller mask (VCM^i), which can be used to obtain VC^i from f^i.

Color-based Refinement of VCM^i. Although very accurate, `DeepLabv3` is not perfect, and as a result, the VCM^i it outputs may still contain parts of the leaked background. To address this issue, we further refine the accuracy of the VCM^i obtained as the output of `DeepLabv3` by applying a statistical color-based correction on the VCM^i . Specifically, for every pixel in $VCM^i_{u,w} = 1$, if a color was observed in $f^i_{u,w}$ with a very low frequency (presumably from the real background), we modify $VCM^i_{u,w} = 0$. Ultimately, the color-based refinement would identify some of the leaked background pixels where their color statistically contrasts with the video caller.

3.5.5 Real Background Reconstruction

After the generation of the virtual background mask (VBM^i), blending blur mask (BBM^i) and the video caller mask (VCM^i) for every frame $f^i \in V$, our attack framework attempts to infer the leaked background (LB^i) in each frame $f^i \in V$ by removing the corresponding components (i.e., VB^i, BB^i, and VC^i, respectively) from the original frame f^i. Any residue pixels not removed from f^i are assumed to be the leaked part of the real background. More specifically, VB^i, BB^i, and VC^i are removed from the original frame f^i by applying the corresponding masks VBM^i, BBM^i, and VCM^i, respectively, to f^i. The residual (leaked background) pixels in all frames are then combined to form a (partially) reconstructed real background.

3.6 Inference Attacks

Location Inference. Our first attack, referred as Location Inference attack, is aimed at inferring the location of a victim caller, given that the adversary has auxiliary information about the caller's background at different locations. We had to address two technical challenges in implementing an automated location inference based on matching reconstructed real background with a set of known backgrounds (and therefore their corresponding locations). We also extend our matching to location across different calls, without knowledge of the full real background (auxiliary information). The first challenge is that the real background may have slightly changed (from the background known to the adversary at a prior time) due to changes in ambient lighting, for example, the effect of difference in daytime and nighttime lighting. The second challenge is that the camera view may have slightly rotated and/or shifted from the background known to the adversary, which is likely if the webcam was re-adjusted or if it a laptop webcam which is angled differently based on how the laptop is opened. We address the first challenge by reducing our matching problem to hue matching at individual pixel level, while ignoring their saturation which is significantly affected by ambient light conditions. The second challenge is addressed by incrementally rotating and shifting the reconstructed background, while trying to find the best match with a known background. Due to space constraints, we do not provide full technical details of the above two matching techniques (to overcome both the above challenges).

Specific Object Tracking. Instead of inferring the (whole) background or associated context (e.g., location), the focus of the Specific Object Tracking attack is to look for the presence (or absence) of a specific object (or objects) in the reconstructed (real) background. For this attack, we assume that the adversary has a *template* of the object he/she is searching for in the reconstructed background. A template is nothing but an array of pixels describing the desired object. To search for the object in the reconstructed background, the object template is incrementally rotated, shifted, and scaled while moving across the pixel map of the reconstructed background frame looking for areas (in the reconstructed background frame) that closely represent (or match) the object. For determining a match, both the color (hue) and the relative distance between the pixels being compared are considered, together with the percentage of the template that is matched. Again, we skip some of the specific technical details due to space constraints.

Generic Object Inference. The goal of the Generic Object Inference attack, which we investigate next, is to generically detect (the presence or absence of) objects in the reconstructed background, without using any template (as before). For this, we employ state-of-the-art object detection frameworks such as RetinaNet [104] and YOLOv5 [133], which are trained with the household object data set [115], common objects (COCO dataset) [105], and ImageNet database [12]. For our experimental evaluations, we employ publicly-available implementations of both RetinaNet and YOLO.

Text Inference. The goal of the Text Inference attack is to infer any textual data present in the reconstructed (real) background. For this attack, we employ the TextFuseNet framework [168], which uses a fusion of Mask R-CNN and Mask TextSpooter models to detect bounding boxes around the text in the image frames. Followed by that, the framework employs a fusion of Residual Neural Networks (RESNet) and Region Proposal Networks (RPN) models to infer the text within the detected bounding boxes.

3.7 Experimental Setup and Data Collection

Next, we present details of the different experimental settings and scenarios under which video call data from human subject participants is collected, and eventually used for evaluating the proposed attack framework. In order to collect video call data that captures a wide-range of target user movements, backgrounds and interactions, we designed three different experimental setups or data collection strategies. The first setup, referred by us as E1 and described in Section 3.7.1, is designed to comprehensively evaluate the vulnerability of the virtual background feature under a variety of target user related parameters (such as actions, backgrounds and clothing/apparel), by carefully controlling these parameters. In this setup, the collected videos are short in nature and there is no active interaction (by the author collecting the data) with the participant while recording the videos. The second setup, referred by us as E2 and described in Section 3.7.2, is designed to mainly evaluate the vulnerability of the virtual background feature in two contrasting settings with no control on the participants' action or movements. The first setting is where the participant is passively watching/listening to some online content, while the second setting is where the participant is actively interacting with another user (for example, presenting some content). In this setup, the collected videos are slightly longer in nature and there is active interaction (by the author collecting the data) with the participant while recording the videos. While the first two setups collected video call data from actual human subject participants, in the third setup (referred by us as E3 and described in Section 3.7.3) videos are captured from the wild (say, from YouTube). Videos from this third setup is used to simulate a setting where the adversary is not an active participant in the call and attempts to compromise the background privacy of arbitrary pre-recorded videos with the virtual background feature. We would like to highlight that all data from human subject participants in our experiments were collected in an ethical fashion and after obtaining the necessary approvals from our university's Institutional Review Board (IRB).

3.7.1 Experimental Setup E1

For this setup, we recruited 5 human subject participants (from on-campus) and asked them to record short two-minute videos performing ten unique actions/movements under varying background and other conditions. Participants used their own personal laptop or desktop computers (with a webcam) to complete all the recording sessions in a remote fashion at a location of their choice. The participants recorded the videos without the researchers present or interacting with them. They were given a fixed time-frame to complete all the video recordings remotely at their own convenience, and then send it back to the researchers (all together) once completed. The ten unique actions/movements included: leaning forward, leaning backward, arm waving, rotating, clapping, stretching, typing, drinking and exiting/entering room. In addition, participants were asked to repeat the above actions under different backgrounds, different lighting conditions (lights on/off), different apparels (similar/contrasting to the background), and with different accessories (e.g., headphones, hats, etc.). Participants were only given high level experimental parameters, with the specifics (exact background, apparels, accessories, etc.) left at the participants' discretion.

For each two-minute video, participants received \$1, with a maximum of \$20 per participant for this phase. At the end of this experimental data collection phase, we ended up collecting a total of 163 videos, which we use in our evaluations.

3.7.2 Experimental Setup E2

For this setup, we recruited 5 participants as well, out of which four were from the E1 data collection phase while the remaining were newly recruited participants. Here, we first asked each participant to initiate a video call with one of the authors (who was collecting the data) and record 4 ten-minute videos where they are passively watching some online content (e.g., YouTube video). Then, they record an additional ten-minute video where they are actively engaging (by means of a presentation) with the author. For each recording, we asked the participants to pick a different background. For each ten-minute video, participants received \$4, with a maximum of \$20 per participant for this phase. At the end of this experimental data collection phase, we ended up collecting a total of 25 videos, which we use in our evaluations.

3.7.3 Experimental Setup E3

For this setup, instead of recruiting human subject participants, we searched for pre-recorded videos in the wild, for example, videos uploaded on classical video sharing services such as YouTube. Our goal here is to find videos of users participating in real video conference calls, or as close to a conference call setting as possible. To accomplish this, we searched on YouTube using a variety of keywords, including terms like “vlog", “podcast", “talk", “review", etc. At the end of this data collection phase, we collected a total of 50 videos with varying lengths, which we use in our evaluations. During our search for videos in the wild, we observed that some of the videos had post-recording edits, overlays, and other elements which were were not very desirable. In those cases, we either completely discarded the video (from our dataset), or only used a segment of the video without the post-recording edits.

3.7.4 Post-processing

All the videos in the three different setups above are collected without a blended virtual background (or background filter). This is done to preserve the true background, which will be later used as ground truth in our evaluation. In order to blend virtual backgrounds post-recording, we replay the collected videos on a video calling software, such as Zoom, via the Open Broadcaster Software (OBS) VirtualCam plugin [25]. OBS VirtualCam simulates a webcam and allows us to pass the pre-recorded videos to the video calling software as if it was directly coming from the webcam. From there, we applied a virtual background to each of the pre-recorded videos using Zoom's virtual background feature. We then use Zoom's record feature to produce recorded versions of our videos (collected from E1-E3) with the virtual background applied. After creating a video dataset using Zoom, as described above, we repeat the same process using the Skype video calling software instead of Zoom, in order to comparatively evaluate the performance of our attack framework on both these popular video calling applications.

3.8 Evaluation

We now present evaluation results of our proposed framework by means of the video call data collected in various settings, as described above. The following results are for videos with virtual background rendered by Zoom, except in Section 3.8.5 where we contrast the results using Skype.

3.8.1 Performance Metrics.

As we propose a framework to exploit a new type of vulnerability, we first define a custom set of metrics that can appropriately capture its performance and effectiveness. We specifically employ only these metrics in our evaluation.

Virtual Background Masking Rate. We define *Virtual Background Masking Rate (VBMR)* for a given frame as the percentage of the virtual background that was masked after applying blending blur to that frame (Section V-C). A 100% VBMR indicates that the virtual background is completely masked from the frame, preventing any virtual background pixels to be mistakenly labeled as a component of the leaked background. Similarly, a 0% VBMR means that all pixels in the virtual background could be classified as part of the leaked background. VBMR is the difference between the pixels from the ground truth data (video before applying virtual background) and the corresponding pixels after the blending blur is applied.

Reconstructed Background Recovery Rate. We define *Reconstructed Background Recovery Rate (RBRR)* as the percentage of the original video frame that is recovered by the proposed real background reconstruction framework (Section 3.5). To compute RBRR for a target video (with the virtual background applied), we count all the pixels of the original video without the virtual background applied that are leaked in one or more frames of the target video, divided by the frame/video resolution.

Action Speed. We define *Action Speed* of a particular action event (conducted by the user or participant in the call) as the number of frames from the start of the action event until the end of the event, divided by the frame rate.

Displacement. For a particular action event, *Displacement* is defined as the percentage of unique pixel changes across all the frames from the start of the action event until the end of the action event.

3.8.2 Virtual Background Masking Rates

We measure VBMR of our framework using three different virtual images and two virtual videos. When the ground-truth virtual backgrounds are included as possible virtual backgrounds, we observed an average VBMR of approximately 98.7%. Alternatively, when the ground-truth backgrounds are not included, we observed a slightly worse average VBMR of approximately 92.6% (when 10 minute video call footages were used).

3.8.3 Background Recovery Rates

We now discuss the reconstructed background recovery rate (RBRR) results for the proposed framework, evaluated under a variety of experimental parameters as outlined in Section 3.7.

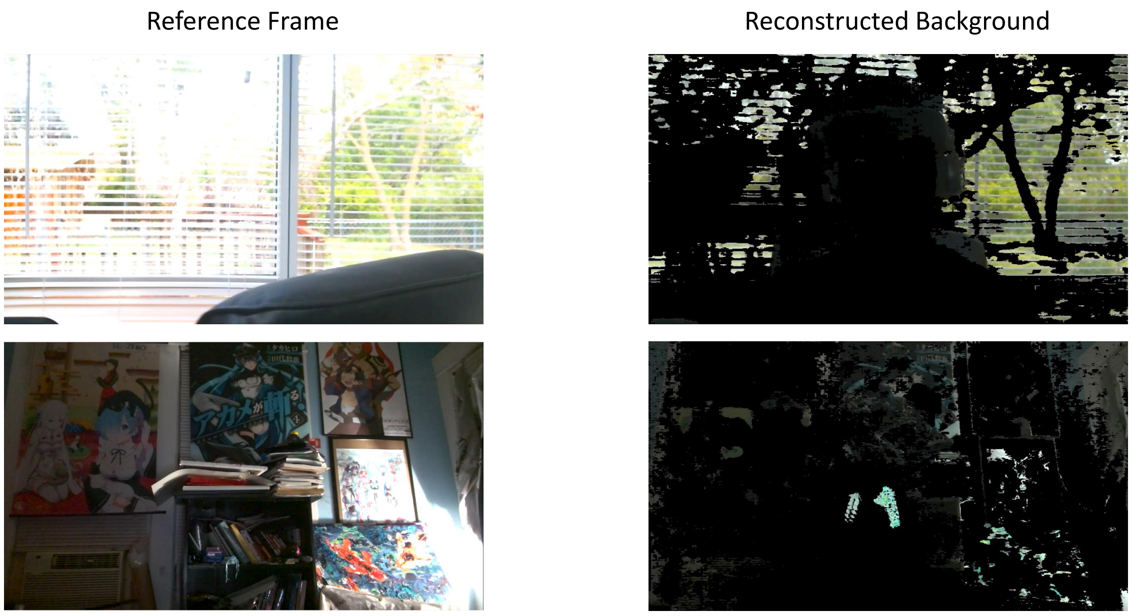


Figure 3.13: Two examples of reconstructed background images using two separate videos from E1.

Impact of Different Framework Parameters. One of the parameters we introduced in our framework is the radius (ϕ) that determines which portion of a frame is part of the blur. If $\phi = 0$, then naturally our obtained RBRR will increase, but at the cost of precision as some of those pixels would be blurred. However on the other extreme, increasing ϕ to a very high value is also not advisable as there will be nothing to recover then. In order to determine a suitable value for ϕ, an adversary could apply a virtual background on a set of static images using the target software, after which the adversary can calculate the average depth of the blur (ϕ) by comparing the virtual image, real backgrounds, and the output images. Following this process, we obtained $\phi = 20$ as the average depth of blur, and use it for the rest of our evaluation.

Effect of Different Actions. An interesting pattern that we observed in our experiments is that action events which result in higher displacements cause more of the real background to be leaked, compared to events with lower displacements. For instance, we can see from Figure 3.14 that on average, entering and exiting (a room) events resulted in a RBRR of about 38.6%, while typing resulted in 4.4% RBRR. Intuitively, this is because entering and exiting events cause significant movement across the real environment, which eventually leads to significant leakage of the real background.

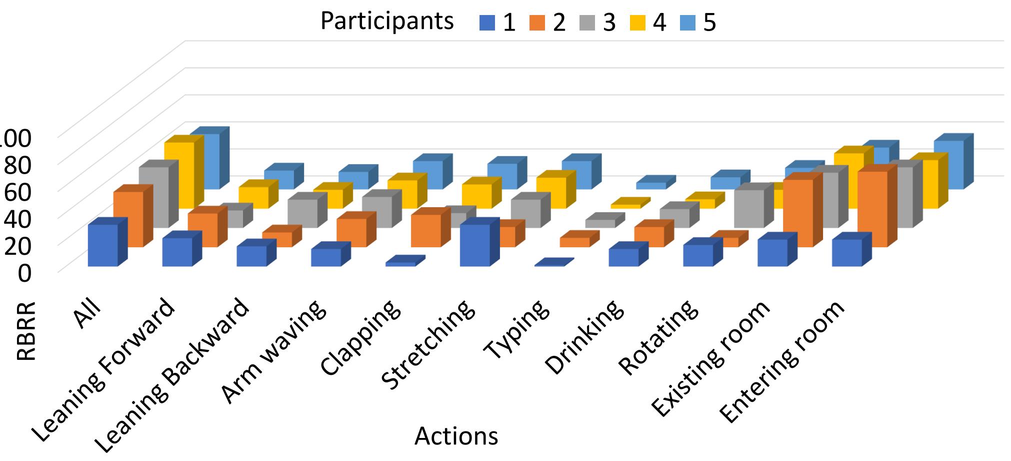


Figure 3.14: Background recovery under various actions.

Effect of Movement. We next analyze how action speed and displacement impacts the background recovery. In our first experiment (E1), we asked the participants to vary the arm waving and clapping action speeds as "slow", "average", and "fast". The interpretation of these subjective scales were left to individual participants. The average [action speed, displacement] for the "slow", "average", and "fast" clapping speeds were [0.9s, 7.2%], [0.26s, 5.1%], and [0.11s, 4.4%], respectively. While for arm-waving they were [2.3s, 28.2%], [0.9s, 24.1%], and [0.7s, 23.4%], respectively. We observed that both for clapping and arm waving, slower action speeds by participants typically resulted in greater displacements. Conversely, faster action speeds resulted in the lesser displacement. From our background recovery results Figure 3.15 in this setup, we observe that action events with the slowest speed returned the highest RBRR (35.9% in "slow", 30.3% in "average", and 33.7% in "fast" arm waving). A faster arm waving, on average, was leaking more background pixels than an "average" arm waving, most likely because of the motion blur produced by the waving action, which made the foreground blend with the background. However, we also saw that too much motion blur could be less revealing if the hand is masked as part of the background. This was observed in case of the "fast" arm clapping event which resulted in a RBRR of 20.8%, while "average" clapping event resulted in a RBRR of only 22.6%.

Effect of Different Accessories. In the next set of experiments, we analyzed the effect of par-

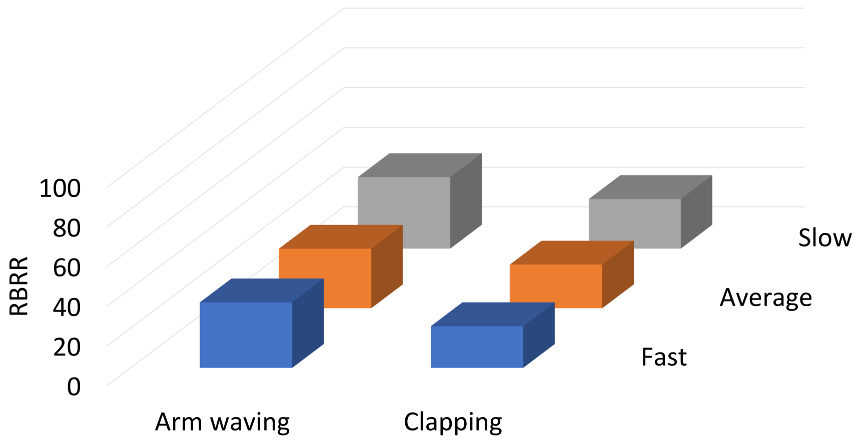


Figure 3.15: Effect of action speed on background recovery.

ticipant accessories like headphones and hats on the real background recovery process. Overall, we did not find any significant difference between the participants' choice of different accessories worn during the call. For instance, Figure 3.16 exemplifies the indifference in background recovery for one particular participant with four different combinations of accessories.

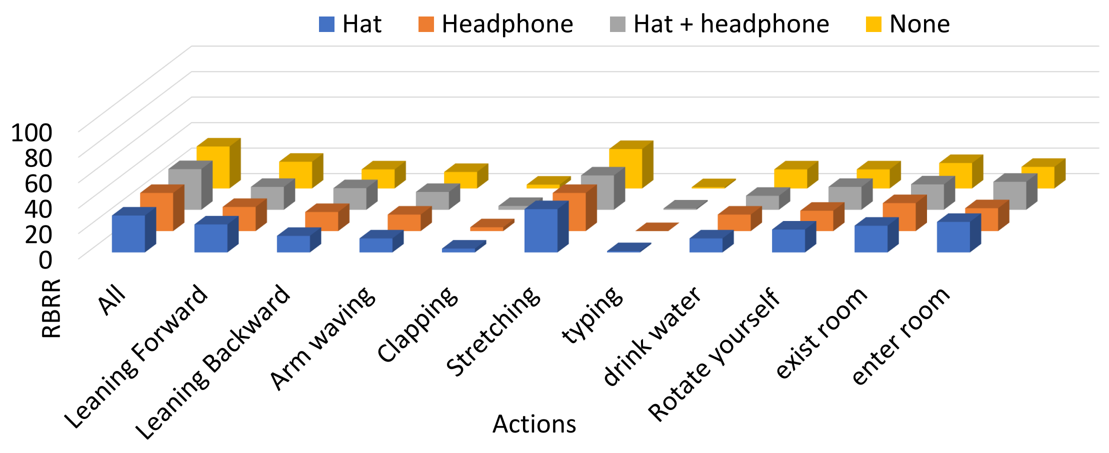


Figure 3.16: RBRR for a participant in E1.

Effect of Different Background Lighting. During this set of experiments, we asked participants to repeat the same video call setup under two different lighting conditions, once with background light OFF and then with the background lights ON. Based on the data collected from these experiments, we observed that, in general, there was more background leakage in low lighting conditions than under high lighting conditions (41.6% RBRR light OFF vs. 39.6% RBRR light ON), as shown in Figures 3.17 and 3.18. While the 2% difference in RBRR is not significant, interestingly, we

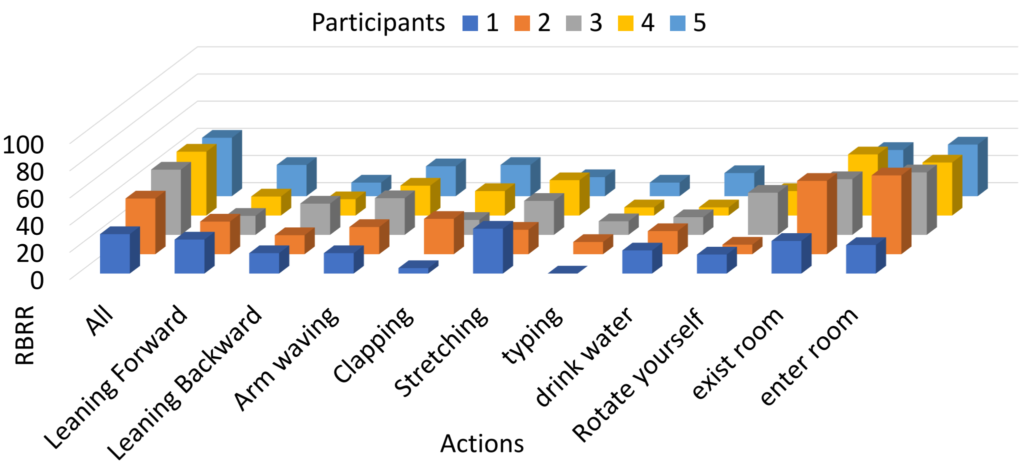


Figure 3.17: Background recovery with background lights off.

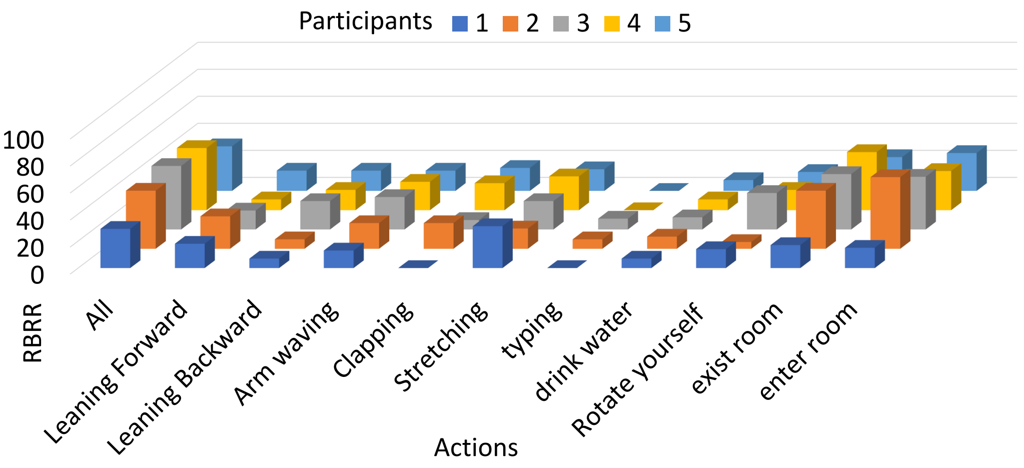


Figure 3.18: Background recovery with background lights off.

observed that the regions of the background reconstructed under the different lighting conditions varied significantly. This suggests that the difference in the RBRR is not merely due to noise.

Being Passive vs. Active in Video Calls. Next, we present our background recovery results for the experimental setups E2 and E3 (outlined in Section 3.7). From the data collected in the setup E2, we observed that passive video callers, i.e., video callers who are not talking during the call, are less likely to leak significant portions of their real background compared to those who are active, i.e., speaking or presenting, during the call. As outlined in Figure 3.19, for passive callers in E2 the obtained RBRR is 9.8%, while for active callers in E2 the RBRR obtained is 30%. For the data collected in the wild from YouTube (setup E3), we were able to obtain a RBRR of 23.9% as shown in Figure 3.19. This is significantly higher than the 9.8% RBRR obtained for passive callers in E2

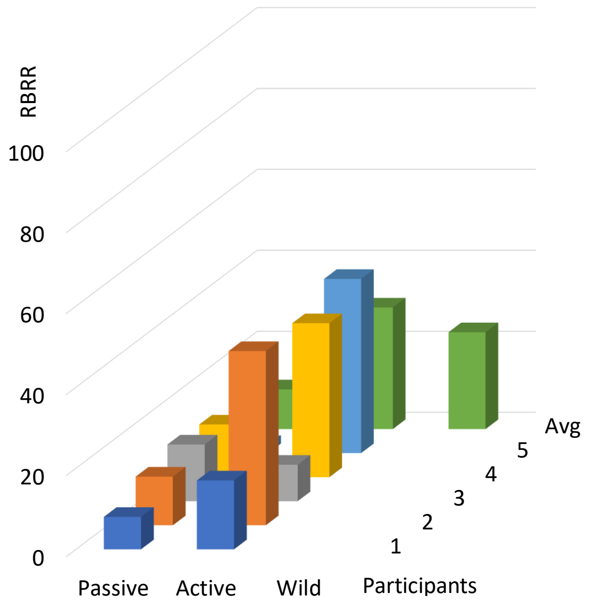


Figure 3.19: Background recovery in E2 and E3 experiments.

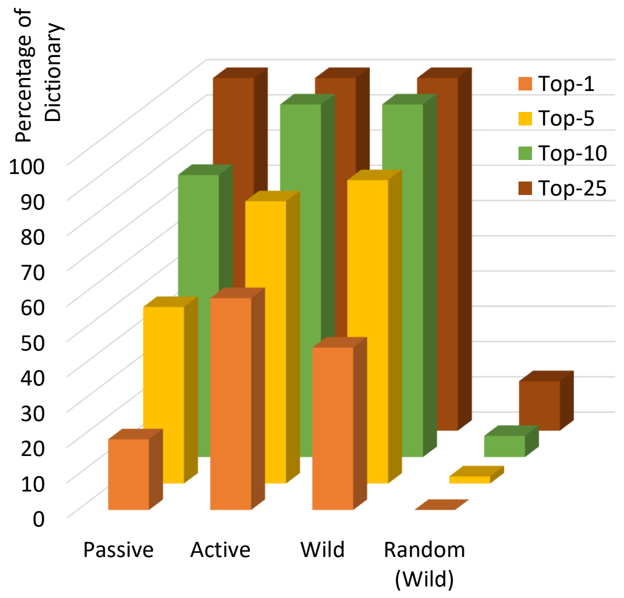


Figure 3.20: Location inference in E2 and E3. Shows the percentage of videos where the background (location) is correctly classified within top-k images from the dictionary.

because users in E3 were actively speaking and presenting (similar to the active users/callers in E2). Another interesting observation is that the RBRR obtained for data collected in E3 is slightly worse than the RBRR obtained for active video callers in E2 (23.9% for E3 versus 30% for active callers in E2). This is most likely because of the high-quality lighting and cameras employed for producing YouTube videos, resulting in the video calling software being able to better differentiate the background from the foreground.

3.8.4 Privacy Attacks

Location Inference. As outlined earlier, in this attack experiment our objective is to infer a target video caller's location by matching his/her partially reconstructed (real) background to a pre-populated dictionary of background images (with known locations and where the target caller may have been in earlier videos). For this experiment, we populated our image dictionary with 200 unique (real) backgrounds from the video calls in E1, E2, and E3. Now, for a particular target video, we rank all background images in the dictionary by computing their similarity to the partially reconstructed (real) background in the target video, with the top rank (rank 1) for image which is most similar and the last rank (rank 200) for the image in the dictionary that is the least similar. This similarity is calculated by comparing the hue changes and distances between all pixels in the reconstructed real background (of the target video call) and background images in the dictionary, and finding the best possible match. We evaluate all videos in datasets E2 and E3 separately by employing the metric top-k ($k = \{1, 5, 10, 25\}$), where k determines how close to the best matching image in the ranked dictionary the real background in the target video call is. So for a top-1 match, the best matching image in the ranked dictionary (i.e., image with rank 1) is the actual real background, while for a top-10 match the actual real background is any image in the dictionary with rank ≤ 10. In addition to the above, we also consider a baseline metric (for evaluation of videos in E3), where k images are randomly chosen from the dictionary. If the real background is present within those k randomly chosen images, then the attack is successful, otherwise it is not. Our experimental results show that higher the RBRR for a target video, better is the

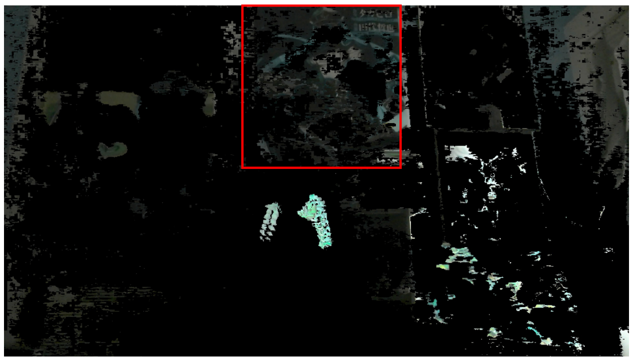
.49 

Figure 3.21: Example of items detected using specific object tracking: a poster.

location inference given an appropriately created dictionary. For instance, our results (Figure 3.20) show that 20% of passive video calls in E2, 60% of active video calls in E2, and 46% of videos in E3 where recovered as top-1 images (from the ranked dictionary). Moreover, regardless of whether the video caller is passive or active, we observed that our proposed scheme is generally effective compared to random guessing.

Specific Object Tracking. Given that we have a template image of an object in a video caller's background, we attempt to track if the object exists in the reconstructed background. We do so by applying the same matching technique in our location matching. If the resulting matching score is high, we consider that the object is present in the background. In our experiment, we were able to accurately infer multiple objects such as shirt, poster, painting, toy, bookshelf, and book. However, a limitation of our object tracking methodology is that it produces a number of false positives when matching our template image with very small areas of the reconstructed background. It produces more false positives when being matched to an area of the reconstructed background with fewer pixels successfully recovered. Therefore, we enforced a minimum matching window size of 5% of the total pixels in the frame and the window must have at least 50% of the pixels successfully recovered. Overall, we were able to track 90 individual objects across different participants' background with 96.7% accuracy.

Generic Object and Text Detection. As our experiments did not regulate the background, we had no control on the objects that may exist in the background. Using our RetinaNet and YOLO object detction models (Section 3.6), pre-trained with COCO and ImageNet datasets, we were

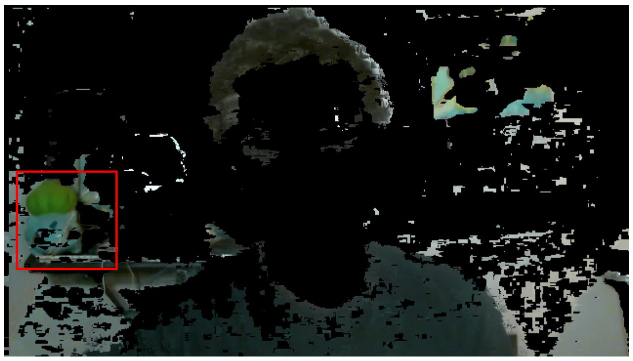
.49 

Figure 3.22: Example of items detected using specific object tracking: a Pokemon figure.

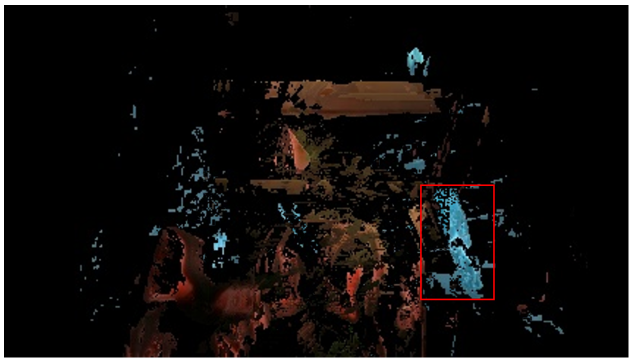


Figure 3.23: A book detected using COCO

able to detect books in four different reconstructed backgrounds (Figure 3.23), a Television in two reconstructed backgrounds, shirts in one reconstructed background, display monitors in three reconstructed backgrounds, and a clock in one reconstructed background. A significant number of videos had a blank wall, bricked wall, window, or door in the background. Text was recovered (using TextFuseNet) from only one video, which was written on a sticky note present in the background (Figure 3.24).

3.8.5 Different Video Calling Software

In addition to Zoom, we also tested our attack on Skype. We observed multiple visual differences between Skype and Zoom virtual background rendering, confirming that they likely use different virtual background masking techniques. Skype was more accurate in its virtual background rendering, resulting in an average RBRR of 19.4% for the E3 dataset, compared to an average RBRR of 23.9% for Zoom on the same dataset. As expected, some privacy attacks in Skype are less ef-

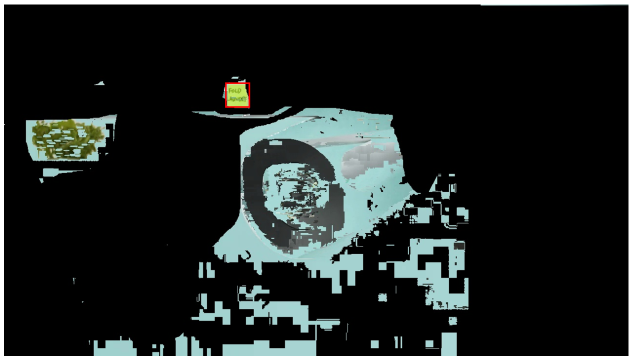


Figure 3.24: Text on a sticky note was detected using TextFuseNet.

fective due to the lower RBRR. For instance, the sticky note shown in Figure 3.24 was leaked from a Zoom call but was not leaked during a Skype call. Likewise, the location inference attack also suffered slightly as for Skype the location of only 76% of passive video calls where ranked within the top-10, compared to Zoom's 80% (as shown in Figure 3.20).

3.9 Mitigation

Next, we briefly outline (and evaluate) a few mitigation techniques against the proposed background reconstruction and privacy attacks.

3.9.1 Dynamic Virtual Background

The main intuition behind the first mitigation technique is to reduce the difference between the virtual background and leaked background pixels as much as possible, by dynamically adapting the pixel properties of the virtual background based on the properties of the real background pixels. To accomplish this, we employ a Gaussian kernel to modify the brightness and saturation of the virtual background pixels for each frame depending on the brightness and saturation of the corresponding real background frame pixels. Further, the hue value of each modified virtual background pixel is forced to randomly fluctuate over multiple hue values (closer to the modified hue value) across different frames, making it further difficult (for the adversary) to differentiate the leaked real background pixels from the virtual background pixels in each frame.

Results. When the dynamic virtual background is applied, our obtained average RBRR increased

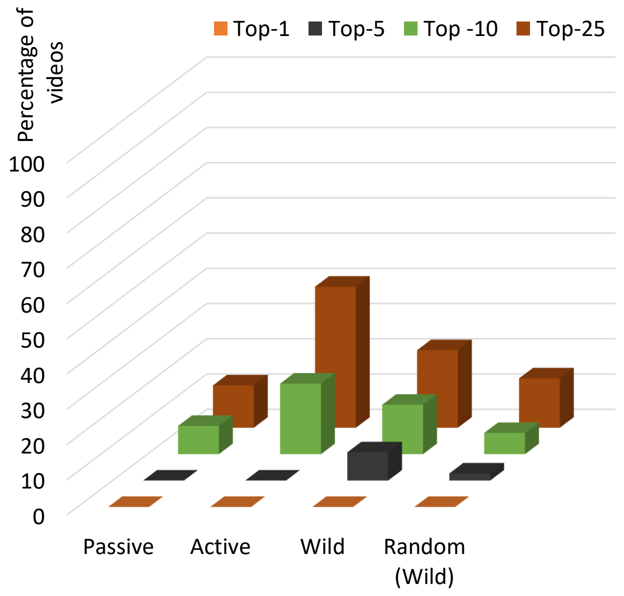


Figure 3.25: RBRR after applying dynamic virtual background.

to 65.8%, 74%, and 86.2%, respectively, for the passive video caller in E2, active video caller in E2, and the E3 datasets (Figure 3.25). While normally a higher RBRR would generally imply a higher leakage of the real background, in this case the recovered real background not only contain pixels of the real background, but it also detects pixels of the virtual background as real background. We re-analyze the location inference attack in Figure 3.26 using the proposed dynamic virtual background mitigation techniques and observe that the overall top-k inference accuracy of the location inference attack reduced considerably. With dynamic virtual backgrounds, a successful location inference within the top-25 was possible for only 40% of the active video call data in E2 and 22% of the video data in E3.

3.9.2 Other Heuristics

We discuss a few other heuristics which could also be employed to mitigate the proposed attack. One heuristic that could be employed is to generate and use a new random virtual background image for every call. The intuition behind this heuristic is that an adversary may not have a prior copy of this newly generated (never-seen-before) virtual background image, which would make the virtual background masking process (Section 3.5.2), and thus the real background reconstruction, a

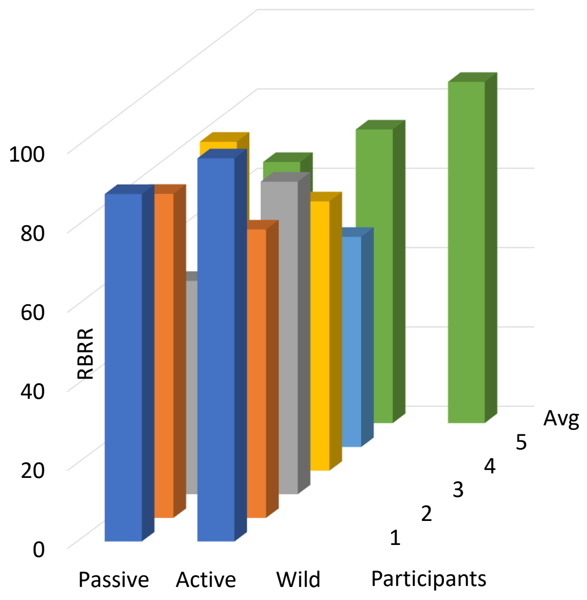


Figure 3.26: Location inference results with dynamic virtual background.

bit more challenging. Another heuristic could be to reduce the number of video call frames shared with the adversary, which will significantly reduce the extent of real background reconstruction, but at the expense of reducing the quality of the video call. Lastly, a more extreme measure could be to not send the actual video call frames (to the adversary) at all. Other than the first frame, all other (following) frames can be replaced with fake video call frames. This can be accomplished by applying a real-time deepfake technique such as the First Order Motion model [140], which will take as input the initial frame of the user's video call (with the blended virtual background) and continue to automatically animate the user for the remaining frames. These animation frames (generated by the First Order Motion model) will be the ones sent in the video call, rather than the actual video call frames. As the real frames are never sent, the real background is never leaked, while the animated frames continue to provide an illusion of the call being live.

3.10 Discussion

The attacks and mitigations we propose and evaluate are based on specific practicality assumptions. For instance, a critical requirement for the location inference attack is the availability of ground-truth background images. Without the ground-truth background images, the adversary will not be able to form a tractable dictionary for efficient location inference. Also, if the (reconstructed) background does not contain any sensitive information, users may not be motivated to apply the proposed mitigations. It is also challenging to assess the overall impact of the proposed attacks and mitigations, as it is difficult to accurately discern users who primarily use virtual backgrounds for preserving their background privacy. Lastly, it is possible that our attacks and mitigations may perform differently if tested on a larger set of more diverse participants. Nevertheless, we validated that at present virtual backgrounds are not a perfect solution for attaining background privacy.

3.11 Conclusion

We proposed and evaluated a real background inference framework leveraging on the imperfections of a virtual background feature in modern video calling software. Based on the reconstructed background, we also proposed a set of different privacy attacks. By means of a comprehensive evaluation using pre-recorded videos in the wild and video call data from human subject participants in a variety of scenarios, we validated the feasibility and implications of such an attack framework.

3.12 Acknowledgment

Research reported in this publication was supported by the US NSF under award number 1943351.

different manuscripts, are mandatory. Where the student is not the sole author of a manuscript, the student is required to make an explicit statement in the introductory material to that manuscript describing the student's contribution to the work and acknowledging the contribution of the other author(s). The approvals of the Supervising Committee which precede all other material in the Master's Thesis/Recital Document or Doctoral Dissertation attest to the accuracy of this statement.

3.13 Publication

Research reported in this chapter has been published as "Mohd Sabra, Anindya Maiti, and Murtuza Jadliwala, "Background Buster: Peeking through Virtual Backgrounds in Online Video Calls ", IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2022."

CHAPTER 4: PRIVACY OF IDENTITY

**As of date, this chapter has been submitted to Network & Distributed System Security Symposium (NDSS), 2023. It was co-authored by Anindya Maiti, Ph.D., Murtuza Jadliwala, Ph.D, Nisha Vinayaga Sureshkanth, Ari Sharma and has been reproduced here with minor revisions.*

4.1 Introduction

Next, we study the privacy of identity while a user in a virtual reality, more specifically. We propose a novel correlation framework to carry out the de-anonymization in such VR services, and comprehensively evaluate parameters such as effect of different actions/movements and the effect of using having their mobile device in different bodily locations (such as different pockets).

Our work advances research investigation of how data from the physical world can be used to compromise the privacy of users in the virtual worlds. We believe that this is the first research effort which investigates this issue. Identity protection is key to VR innovation as otherwise users will be hesitant to participate in the ecosystem [53] in order to protect their privacy, reputation and security. To protect users from the potential de-anonymization attack, we also propose novel countermeasures that users can adopt while participating in VR applications. In summary, we make the following main contributions in this chapter:

1. Contribution 1: A framework that transmutes both motion sensor data and avatar's visual movement data into a comparable *activity-vector*.

2. Contribution 2: A correlation model that filters mismatching activity-vectors, and ranks matching activity vectors from best to worst.

3. Contribution 3: Test data collection for real-world human participants, and a comprehensive empirical evaluation of the correlation framework under various settings.

4. Contribution 4: Improvements and optimizations of the correlation framework for a *large-scale* attack.

4.2 Related Work

Research efforts in the literature related to our work can be categorized into those that focus on inferring private information by means of mobile device motion sensors which we discuss first, followed by those that propose new sensitive information inference vectors for VR systems and applications.

Information leakage through mobile device motion sensors: Mobile and wearable device motion sensors such as accelerometers and gyroscopes have been heavily scrutinized in the research literature for their potential to be employed as a side-channel for leaking users' private information. For instance, motion sensor data on smartphones and smartwatches have been utilized to infer keystrokes and passwords [70,106–108,127,137], identify lock screen patterns [167], deduce travel routes and location [90,121,124], infer speeches [89,91,116], infer handwritten text [158], reconstruct 3D models from printer vibrations [144], and estimate demographic information [81,143]. Application of on-body motion sensors onboard consumer mobile devices such as smartphones and smartwatches for user authentication [102,159,165] is another closely related research space that has received significant attention in the literature. However, such biometric authentication systems require training data from individual users, which is not available in our adversarial setting.

Information leakage in VR systems and applications: Albeit relatively new as a consumer technology, VR has garnered a host of privacy and security concerns. As mentioned earlier, attacks such as password inference from finger movements (using motion sensors) when typing a password in the virtual world can become a security problem if the same password is reused by the user in real world [80]. Some VR headsets include eye-tracking, which can reveal valuable personal information [99]. VR, when used in conjunction with Deepfakes [157], can also become a serious threat as an adversary can potentially utilize personal gait and movement data collected from a VR headset to create a very authentic-looking fake video. These type of attacks can be used to damage personal reputation [20, 138], conduct social engineering attacks [27, 160], and spread misinformation (fake news) [84,97].

Authentication in VR is a closely related research topic [146], wherein authorized sensors on the VR headset or paired on-body controllers are used to authenticate individual users. Synchronization among multiple (on-body) sensors [76, 94] for utility focused applications is another closely related research topic. Unlike these prior research, in our attack we focus on out-of-band motion sensor data, which are not natively paired with the VR system. De-anonymization solely using movements observed in the virtual world is difficult, especially when the confusion set size is large. In this work, we carry out de-anonymization of users of a VR platform by correlating visually observed movements of the user's avatar in the virtual world with out-of-band motion data available from users.

Use of anonymous avatars and identity transformation inside a virtual reality experience [86,87, 111] is a significant factor contributing to the technology's popularity. Therefore, identity protection is key to VR innovation as otherwise users will be hesitant to participate in the ecosystem [53] in order to protect their privacy, reputation and security. Previous works on de-anonymization of VR users utilized *in-band* data (such as sensors on the VR systems and/or movement characteristics of virtual avatars) to infer users' identity [117,118], anthropometrics [123], environment [123], device information [123, 151], and demographics [123]. To the best of our knowledge, our proposed de-anonymization attack using out-of-band motion data has thus far not been analyzed or publicly presented. We also evaluate the scope of the proposed de-anonymization attack within a small set of users and at a larger scale. To protect users from the proposed attack, we also suggest countermeasures that users can adopt while immersing in a VR experience.

4.3 Threat Model

We consider an adversary whose goal is to de-anonymize users of a VR ecosystem by correlating visual movements of anonymous virtual world avatars with *out-of-band identifiable* mobile/wearable motion sensor data from target users. The *size* of the labeled motion dataset of users in the possession of the adversary, representing the *confusion set* of the target VR user or avatar, may vary between a *large-scale* where the cardinality (of the dataset) may be very high, to a significantly smaller *small-scale* (e.g., employees of a company or participants of an event). Similarly, the recordings of VR users or avatars will result in a visual movement dataset, which can also range between a *large-scale* where its cardinality may be very high, to a significantly smaller *small-scale* such as avatars present within a (targeted) virtual room or playing a (targeted) virtual game. As depicted in Figure 4.1, the goal of the adversary is to de-anonymize a target user (i.e., its avatar) in the VR space by matching an element in the labeled motion dataset to the element (corresponding to the target user or avatar) in the visual movement dataset by utilizing some efficient correlation mechanism, similar to the one we propose in this chapter. This adversarial goal can be easily extended to include de-anonymization of multiple VR users or avatars.

In order to compile the visual movement dataset (denoted by $V = \{v_1, v_2, \ldots, v_p\}$, with cardinality p), the adversary has to join the virtual world, observe and record each avatar for a baseline duration of time within which a series of movements are likely observed. In case of the VR service provider being the adversary, this process can scale easily. In order to compile the labeled motion dataset (denoted by $M = \{m_1, m_2, \ldots, m_q\}$, with cardinality q), the adversary installs a malicious data collection app on the mobile/wearable devices of a targeted set of users, which records motion (accelerometer and gyroscope) sensor data and reports it back to the adversary. As explained earlier, this targeted set of users can be at a small or large scale. Typically, this can be achieved by means of a malicious SDK and/or a trojan app that offers some utility to the users on the front-end (e.g., a game or a social networking service), while surreptitiously recording the motion data on the back-end. We also assume that both datasets (V and M) contains timestamps which are fairly in sync with the standard global time.

For popular apps/services that also offer a VR platform, for example, Meta, such an attack can potentially be scaled globally for both the motion and visual datasets. Nonetheless, such an attack is easier to be carried out at a *small-scale*, implying that the malicious mobile/wearable app has to be popular within a small group of users and/or the VR ecosystem has to be popular within the group. When both p and q are large, the correlation process to de-anonymize all users grows to be computationally challenging for the adversary. Later in Section 4.8, we propose optimization techniques that can significantly reduce the computational complexity, and thus the average runtime, of the proposed correlation framework in the above case. Below we present two different realistic scenarios representing our threat model.

Scenario 1. A large organization (such as Meta) that operates both a popular VR platform (such as Metaverse) and a popular mobile app (such as Facebook, WhatsApp, and Instagram) can collect both the visual movement dataset and motion sensor dataset for respective platforms. Users who do not want to be identified across both of these platforms are susceptible to the proposed de-anonymization attack, even when using anonymous identity and avatar on the VR platform. This scenario represents a *large-scale* attack where the confusion set is large.

Scenario 2. A criminal group uses VRChat [38] to anonymously meetup. An undercover police officer present in the meetups is able to record the visual movements of individual (anonymous) avatars of the criminal group. With the help of a popular smartphone app company (such as Google), the police is also able to collect identified motion sensor data from a list of known criminals and suspects. Thereafter, the proposed correlation framework can be used by the police to de-anonymize members of the group on VRChat. This scenario represents a *small-scale* attack where the confusion set is small.

4.4 Correlation Framework

Our proposed correlation framework (Figure 4.2) is composed of two key components. The first component converts both the (out-of-band) motion sensor data and the visual movement data into a comparable format, which we refer to as *activity-vector series*. The activity-vector series enables us to directly compare and match elements from the two datasets (V and M) using a *matching heuristic*. The second component in our framework *ranks* the closest matches across the elements of either dataset, in a fashion such that the high ranked matches are likely associated with the target user (identifiable from M).

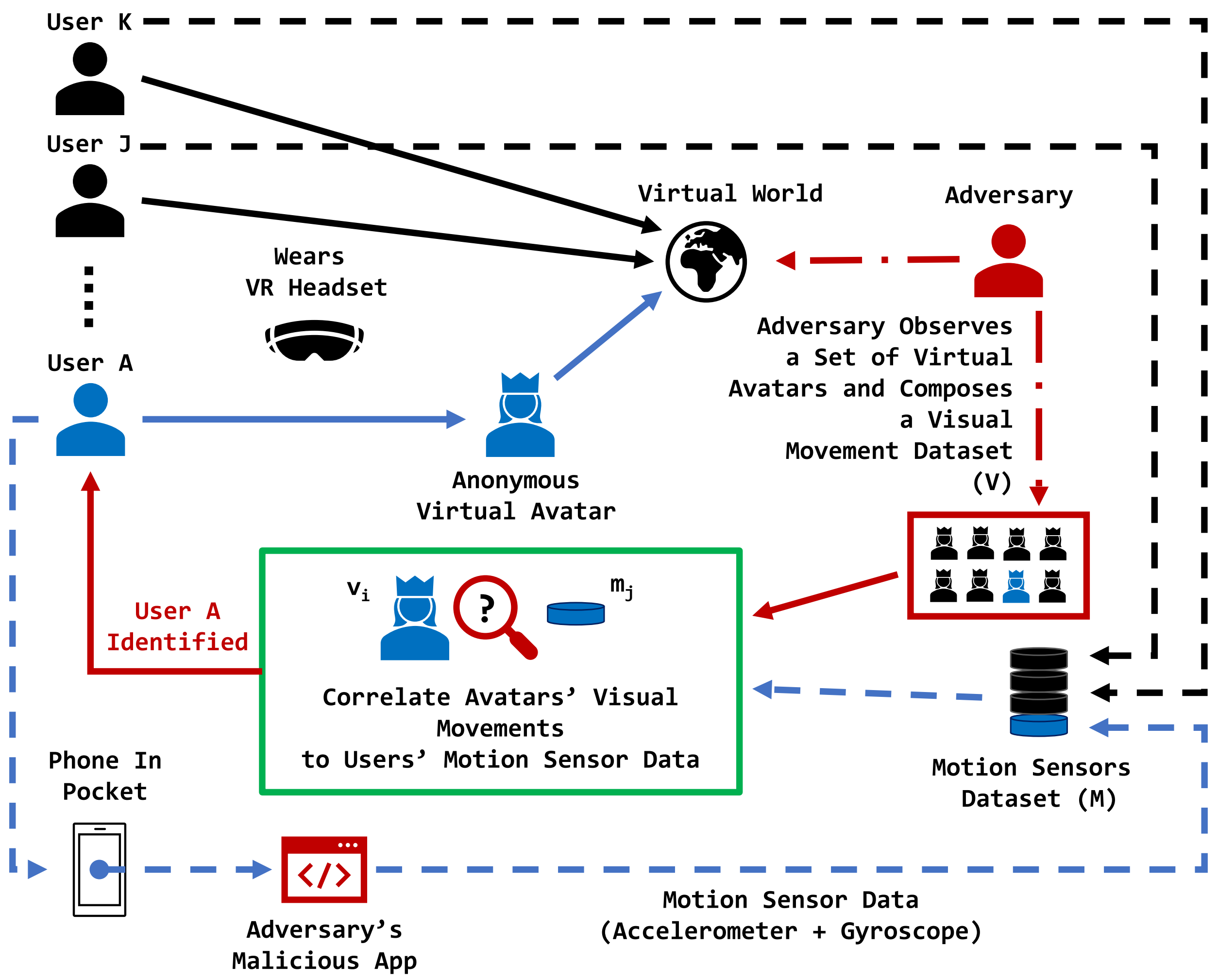

User K
User J
User A
Wears
VR Headset
Virtual World
Adversary
Adversary Observes a Set of Virtual Avatars and Composes a Visual Movement Dataset (V)
Anonymous Virtual Avatar
User A Identified
v_i
m_j
Correlate Avatars' Visual Movements to Users' Motion Sensor Data
Motion Sensors Dataset (M)
Phone In Pocket
Adversary's Malicious App
Motion Sensor Data (Accelerometer + Gyroscope)

Figure 4.1: Threat model.

4.4.1 Activity-Vector Series

Our motivation behind defining an *activity-vector series* stems from the fact that the two datasets (motion sensor data from the mobile/wearable device and visual movement dataset from the VR app) are not directly comparable to each other. The motion sensor data comprises of samples measuring linear acceleration and orientation changes of a user's body, whereas the visual movement data consists of a video wherein an anonymous avatar's movements are recorded as changes in pixels across its frames. Consequently, we define an *activity-vector series* as a sequence of activities observed (classified by some machine learning or ML model as discussed later), combined with a pairwise sequence of "magnitudes" for each observed activity from each of the data sources (visual movements and motion sensor). Our magnitude quantification of an observed activity is approximate, but serves as a critical attribute in our correlation framework as detailed in Section 4.4.5.

More precisely, our activity-vector series is composed of the following commonly observed activities: *idle*, *body rotation*, *head rotation*, *hand movements*, *walking*, *bending*, *jumping*, and "*other*". These were the common movements observed in over 2000 hours of activity data collected inside VRChat [38] by us (more details on data collection can be found in Section 4.5). These activity classifications combined with magnitude calculations form a vector-like representation where each observed activity has a corresponding magnitude information (similar to a vector which consists of direction and magnitude). An activity-vector series from either sources can be depicted as follows:

Left-front Hip Pocket (Motion Sensor)

Activity	*walking*	*walking*	*idle*	*bending*	*walking*	*walking*	*jumping*	*idle*	*walking*	*jumping*
Magnitude	a_4	a_3	a_1	a_7	a_6	a_{10}	a_8	a_2	a_5	a_9

where $a_i \in \mathbb{R}^+$ is the positive real magnitude of an activity time window, such that $a_1 > a_2 > \ldots > a_{10}$. In order to generate this activity-vector series, we next detail the steps taken to pre-process and utilize supervised machine learning models to classify the activities observed in individual sequences.

4.4.2 Pre-Processing

We first segment both the physical motion data (obtained from the mobile device motion sensors) and the visual movement data (obtained from the VR apps) into small time windows (of w seconds each) and classify each window as one of the eight aforementioned actions. We empirically evaluate the effect of the size of w on correlation accuracy in Section 4.7.1 and use the optimal value for rest of the evaluation. For the visual movement data, we further separate individual user's avatar from the background, so as to better classify the movements of the avatar without any background noise. PaddleSeg [78], an open-source toolkit that applies image segmentation using different techniques, was used to segment out the individual avatars. More specifically, we used a pre-trained ORCNet model with HRNet backbone that was trained using the Cityscapes dataset [23]. For the motion sensor data, we apply a Savitzky-Golay filter [131] to smooth the

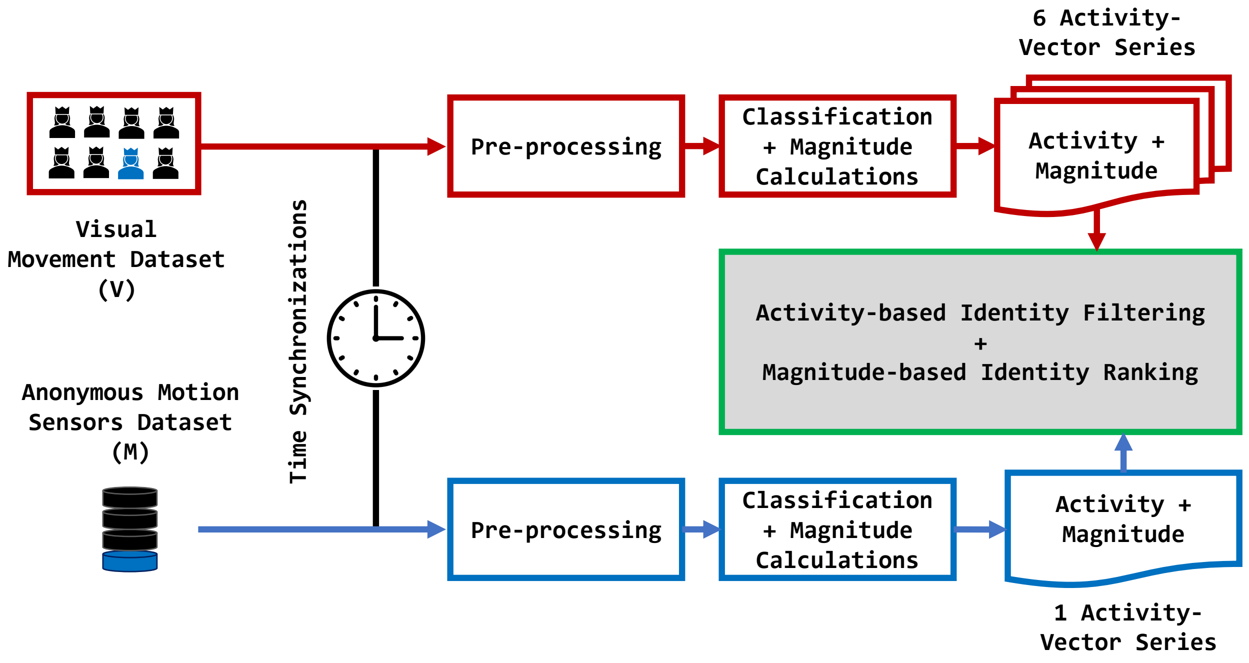

6 Activity-Vector Series
Visual Movement Dataset (V)
Time Synchronizations
Pre-processing
Classification + Magnitude Calculations
Activity + Magnitude
Activity-based Identity Filtering + Magnitude-based Identity Ranking
Anonymous Motion Sensors Dataset (M)
Pre-processing
Classification + Magnitude Calculations
Activity + Magnitude
1 Activity-Vector Series

Figure 4.2: Overview of our correlation framework.

signals for noise reduction before classification.

4.4.3 Training Data Generation

In order to generalize and scale our activity classification for a *large-scale* attack, we generate training data as an amalgamation of a well-known dataset in the literature and add synthetically generated variations to capture a wide range of bodily variances and anomalies (often caused by imperfections in the VR systems) all of which are otherwise impractical for collection from real human subjects. Specifically, we generate the training data of our visual movement classifier using the 3D game engine Unity [33] (Figure 4.3), utilizing the Carnegie Mellon University (CMU MoCap) [6] dataset and synthetically generated variations of motions captured in the CMU MoCap dataset. The CMU MoCap dataset was created using a motion capture system where the subjects wore 41 markers and performed various activities. It is a well-known dataset for evaluation of activity recognition frameworks [64, 120, 141], and can be applied to reproduce avatar movements inside Unity using corresponding body keypoints.

Our synthetically generated movement variations randomized the speed between $0.25\times$ and $2\times$ of CMU MoCap speeds, and rotation angle between $-10 and$+10 of CMU MoCap rotation angles.

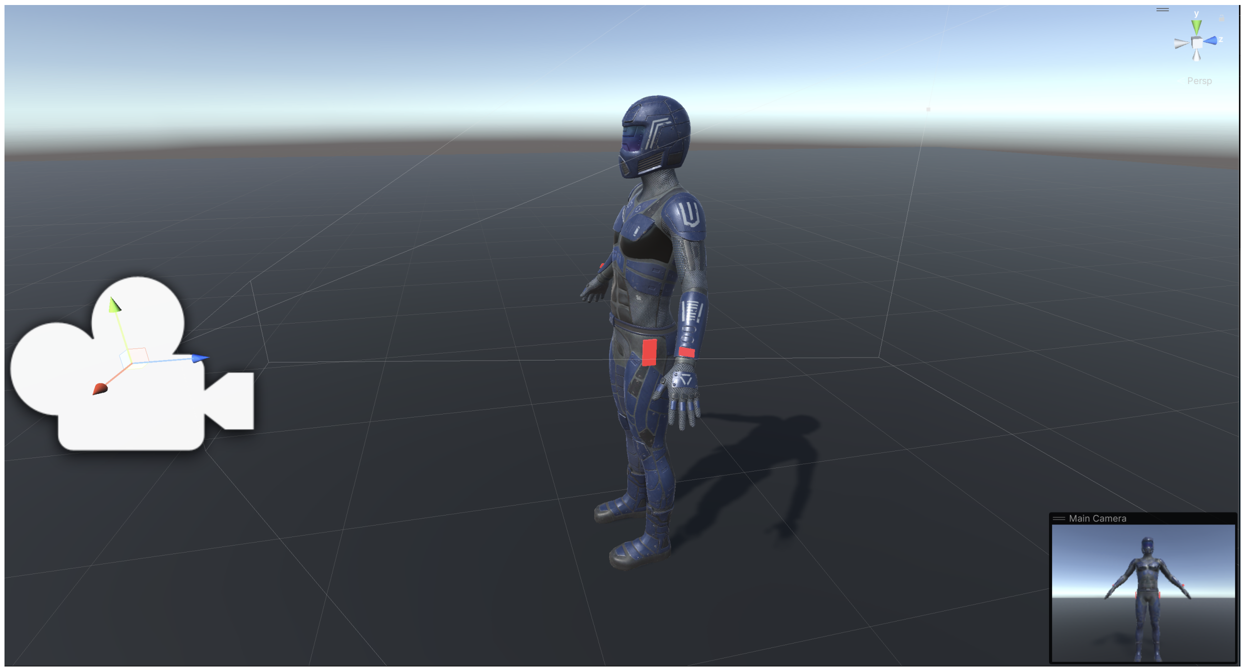


Figure 4.3: The training data generation setup inside Unity, depicting only one camera viewpoint and virtual motion sensors attached to the avatar (in red).

In addition to the CMU MoCap model avatar, we also train using another freely available avatar, namely the *Futuristic soldier - Scifi character*[1]. As the video movement data is dependent on the viewpoint of the adversary, we also capture varying camera positions around the virtual avatar in Unity (Figure 4.3). Specifically, the camera position was randomized around the avatar (across all angles for which the avatar is visible), enabling a different visual perspective and thus improving our classifier training. The visual movements of the avatars were recorded using OBS Studio [22].

Additionally, in Unity we attached a custom-made *virtual* motion sensor to the avatar (Figure 4.3), which is able to capture acceleration and orientation changes of the avatar. This virtual motion sensor closely captures the kinematic forces experienced by the avatar in the same way a smartphone or smartwatch motion sensor on a real person would experience, and it allows us to collectively train a classifier for the motion sensor data alongside the visual movement classifier. The key advantages of using such a virtual sensor for training are the elimination of synchronization errors, and not requiring real human subject participants for data collection (except for the human subject participants who helped in the development of the CMU MoCap dataset). Note that

[1]https://assetstore.unity.com/packages/3d/characters/humanoids/sci-fi/futuristic-soldier-scifi-character-202085

for our experimental evaluations (Section 4.7) with an adversarial standpoint, we compose a realistic *test* dataset with the help of real human subject participants and also address synchronization errors between the motion sensor and visual movements data (Section 4.7.3).

4.4.4 Activity Classification

We collectively utilize Apple's Core ML [2] and Create ML[3] libraries to generate two classification models (each trained separately), one for the video movement training data and another for the motion sensor training data. The Core ML model is already trained by Apple for generic action and activity classification, and can be further customized using transfer learning [113] using the training data generated in Section 4.4.3. Prior research has already demonstrated the feasibility of such activity recognition using Core ML [101]. Moreover, Apple's Vision framework[4] is already pre-trained for keypoint detection on humans, which can also be utilized with Core ML on humanoid avatars. *Applying these trained classification models on test visual movement and motion sensor data split into w second windows will result in a sequence of activities observed on the two data sources, which is one of the two sequences in the activity-vector series defined earlier.*

4.4.5 Activity Magnitude

Intuitively, when the same classified activity is observed in both data sources (in a given time window), we can improve our identity correlation by *ranking* smaller magnitude differences above larger magnitude difference. For example, if an anonymous avatar is observed to be walking fast in the virtual world (high magnitude), it is likely that their activity magnitude will also be high on the motion sensor data. As mentioned earlier, our magnitude quantification of an observed activity is approximate. For the motion sensor, we calculate magnitude of each w second activity window as the average magnitude of acceleration vectors in the motion sensor data. For the visual movement data, we utilize optical flow to compute the average acceleration of areas on the avatar's

[2]https://developer.apple.com/documentation/Core ML
[3]https://developer.apple.com/documentation/createml
[4]https://developer.apple.com/documentation/vision

body where the motion sensor may be attached. Optical flow estimates the motion of objects between consecutive frames in a video, caused by the relative movement between the object and camera [83, 92].

However, as some activities tend to generate disproportionate levels of motion in various parts of the body, it may result in different magnitudes of movements for the same activity. Furthermore, as the adversary may not have knowledge of the motion sensor's positioning for each user's data, the visually observed magnitude of movement experienced by an avatar's different body keypoints is another attribute that should be factored in to improve our correlation model. We consider six usual body positions where the motion sensor is likely to be attached, such as a smartphone in pant pocket or a smartwatch on the wrist: left-front hip pocket, right-front hip pocket, left-back hip pocket, right-back hip pocket, left wrist, and right wrist. As a result, the activity-vector series calculated from the visual movement dataset will consists of six different magnitude sequences (for the same activity sequence) as follows:

Left-front Hip (Visual)

Activity	*walking*	*walking*	*idle*	*bending*	*walking*	*walking*	*jumping*	*idle*	*idle*	*jumping*
Magnitude	a_4	a_3	a_1	a_7	a_6	–	a_8	a_2	a_5	a_9

Right-front Hip (Visual)

Activity	*walking*	*walking*	*idle*	*bending*	*walking*	*walking*	*jumping*	*idle*	*idle*	*jumping*
Magnitude	a_4	a_3	a_2	a_7	a_6	a_{10}	a_8	a_1	a_5	a_9

Left-back Hip (Visual)

Activity	*walking*	*walking*	*idle*	*bending*	*walking*	*walking*	*jumping*	*idle*	*idle*	*jumping*
Magnitude	a_4	a_2	a_3	a_7	a_6	a_{10}	a_8	a_1	a_5	a_9

Right-back Hip (Visual)

Activity	*walking*	*walking*	*idle*	*bending*	*walking*	*walking*	*jumping*	*idle*	*idle*	*jumping*
Magnitude	a_4	a_3	a_1	a_7	a_6	a_9	a_8	a_2	a_5	–

Left Wrist (Visual)

Activity	*walking*	*walking*	*idle*	*bending*	*walking*	*walking*	*jumping*	*idle*	*idle*	*jumping*
Magnitude	a_3	a_4	a_1	a_6	a_7	a_8	–	a_2	a_5	a_9

Right Wrist (Visual)

Activity	*walking*	*walking*	*idle*	*bending*	*walking*	*walking*	*jumping*	*idle*	*idle*	*jumping*
Magnitude	a_1	a_3	a_4	a_6	a_7	a_9	a_8	a_2	a_5	a_{10}

where "–" implies unobservable position for optical flow calculations, all a_i in red depict mismatched magnitude rank with the left-front hip pocket motion sensor activity-vector series shown in Section 4.4.1, and all green a_i imply matching magnitude rank. Moreover, there is an activity misclassification in this example at the ninth window, highlighted as *idle* in red. All of these seven magnitude sequences (one from motion sensor data and six from visual movement data) are utilized in the correlation and identity ranking processes described next.

4.4.6 Correlation and Identity Ranking

The first intuitive assumption in our correlation framework is that the order of activities conducted by an user (and their avatar) will be unique when observed for a long enough duration. Intuitively, this observation duration can be shorter in a *small-scale* attack where the confusion set is smaller.

In a *large-scale* attack, the observation duration has to be longer because with a large confusion set the occurrence of more than one anonymous user conducting the same sequence of activities within a short observation duration is more probable, thus creating confusion between them. We use this first assumption to filter out unlikely matches from our identity ranking calculations, using the activity sequences in the activity-vector series.

Our second intuitive assumption is that varying activity magnitudes caused by disproportional levels of motion in various parts of the body can be utilized to identify closely correlated visual movement and motion sensor sequences. Accordingly, we utilize magnitude correlation rankings to rank known identities (from dataset M) such that users with motion sensor magnitude sequence closely matching to a visual movement magnitude sequence (best of the six visual positions) are ranked closer to 1.

Activity-based Filtering

As the activity classification is not perfect, we cannot reliably use the sequence of activities for correlation. Instead, we use a high degree of mismatch between sequences of activities (across visual movement and motion sensor data) to filter out identities whose motion sensor data are objectively different from an anonymous avatar being observed. More specifically, we calculate the hamming distance between the motion sensor activity sequence and the visual movement activity sequence (which is the same for all six activity-vector series generated from the visual movement data). Thereafter, we eliminate pairs with distance threshold $> t$ from further magnitude-based identity rankings. We empirically evaluate threshold t in Section 4.7.1 as part of our framework parameter optimizations. For example, between the pair of activity-vector series illustrated in Section 4.4.1 and Section 4.4.5, this hamming distance is 1 (or 10%) due to the activity mismatch in the ninth time window.

Magnitude-based Ranking

After filtering, we are left with identities whose motion sensor activity sequences closely matched at least one of the six visual movement activity sequences. We utilize Spearman's rank correlation coefficient [170] to correlate and rank potential identities based on magnitude sequences, which is computed as follows:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

where n is the number of observations (of w second windows) in the activity-vector series, and d_i is the difference in the paired ranks of the two magnitudes (across the visual movement and motion sensor data sequences) at the i_{th} time window.

The higher the Spearman's rank correlation coefficient, the more likely the two sequences correlate to each other, and thus the corresponding identity from M would be ranked closer to 1 out of the q (minus the identities that did not pass the activity-based filtering). As the adversary does not have positioning information of the motion sensor on the users' body, we compute Spearman's correlation coefficient for the six likely positioning of the motion sensors (Section 4.4.5), and consider only the maximum for identity ranking. Between the examples shown in Section 4.4.1 and Section 4.4.5, magnitude from the visual data sequence of the left-front hip will have the highest Spearman's correlation coefficient with the left-front hip pocket motion sensor magnitudes.

When activity-based filtering threshold t is set very low (i.e., only tolerance for very minor or no mismatches in the activity sequences), it is also possible that all identities are eliminated from this magnitude-based raking, thus resulting in no identity ranking. The entire correlation procedure is described in Algorithm 4.1.

4.5 Experimental Setup

To evaluate our proposed correlation framework and training methodology, we collect test (visual and motion sensor) data from human subject participants using a real VR application. In this section, we outline the details of our data collection procedure.

Algorithm 4.1 Correlation Algorithm.

```
1: Input:
2:   $video[]$                                   ▷ Video's activity-vectors series
3:   $motion[]$                                  ▷ Motion's activity-vectors series
4:   $t$                                         ▷ Filtering Threshold
5: Output:
6:   $ranked[]$        ▷ Ranked list of correlated motion/video indexes with maximum Spearman's rank
   correlation coefficient
7: procedure CORRELATE
8:     $correlated[]$                        ▷ Maps motion indexes to correlated video indexes
9:     $unranked[]$    ▷ Unranked list of correlated motion/video indexes with maximum Spearman's rank
   correlation coefficient
10:    for $i$ in range($video.size() - 1$) do
11:        for $j$ in range($motion.size() - 1$) do
12:            if $HammingDistance(video[i], motion[j]) < t$ then
13:                $correlated[i].append(j)$
14:            end if
15:        end for
16:    end for
17:    for $i$ in range($video.size() - 1$) do
18:        for $j$ in range($correlated[i].size() - 1$) do
19:            $m_{idx} = correlated[i][j]$                    ▷ Motion index
20:            $maxSpearman = max(Spearman(video[i], motion[m_{idx}]))$
21:            $unranked[i].append(\{maxSpearman, m_{idx}\})$
22:        end for
23:        $ranked[i] = unranked[i].sort()$    ▷ Sorted based on Spearman's rank correlation coefficient
24:    end for
25: end procedure
```

4.6 Controlled Activity Sets in Data Collection

Table 4.1: List of controlled actions performed by participants in the real and virtual reality worlds.

Action Types	**Action Description**
Head-based	Looking [left, right, up, down]
	Rotating the head in [clockwise, anti-clockwise] directions
Arm-based	Raising [left, right, both] arms in [forward, upward, sideward] directions
	Rotating [left, right, both] arms in [clockwise, anti-clockwise] directions
	Stretching arms [forward, upward, sideward]
Palm-based	Handshaking with [left, right, both] arms
	Waving with [left, right, both] arms in [forward, upward] directions
	Thumbs up and down with [left, right, both] arms forward
	Clapping with hands forward
Leg-based	Stepping along [left, right, forward, backward] directions
	Walking diagonally towards [left, right, forward, backward] directions
	Raising [left, right] knee
Combination-based	[Twisting hip, turning body around] in [clockwise, anti-clockwise] directions
	Crouching or squatting, Jumping up and down
	Sitting on the [floor, chair]
	Exploring [public, private] instances
	[Walking, running] in [straight, zig-zag] paths
	[Talking, browsing] smartphone in [portrait, landscape] modes
	Fiddling with an object
	Picking up objects placed on the [floor, table]

4.6.1 Participants' Task

Our participants (details in Section 4.6.3) carry out a set of representative activities in a virtual reality app while carrying a smartphone and smartwatch on their body (details in Section 4.6.4). Table 4.1 details all the different types of activities that participants were instructed to perform, in addition to other uncontrolled activities that they may perform while navigating inside the virtual world. The controlled actions include movement of the head, arms, palms, legs, and also actions that require combinations of them. These different actions were chosen to generate a wide variety of different movements within the limited time we had with the participants. During the uncontrolled activity phases, participants were free to interact with the VR app on their own volition, not limited by the aforementioned activities. The average time our participants spent on the virtual reality app, in order to provide us data for our study, was 1 hours and 8 minutes.

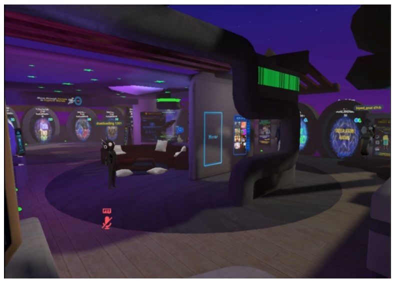


Figure 4.4: HC1

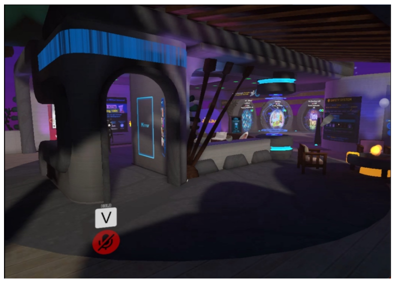


Figure 4.5: HC2

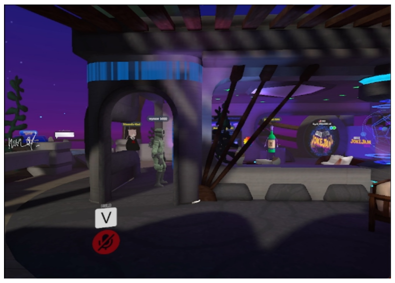


Figure 4.6: HC3

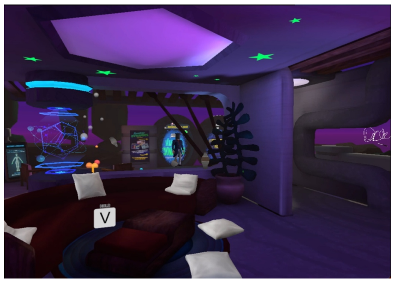


Figure 4.7: HC4

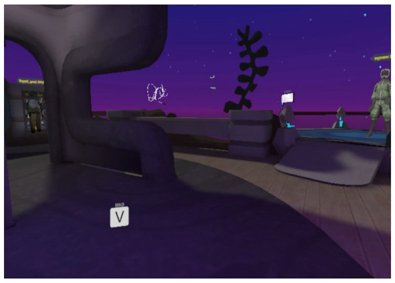


Figure 4.8: HC5

Figure 4.9: Adversarial viewpoints.

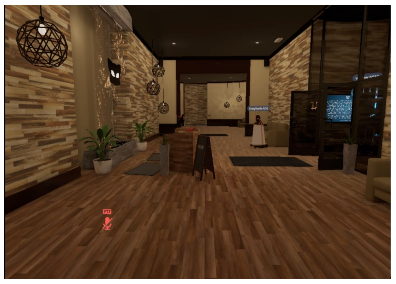


Figure 4.10: BC1

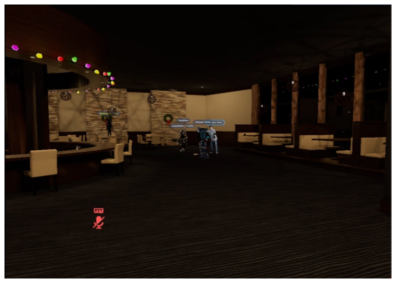


Figure 4.11: BC2

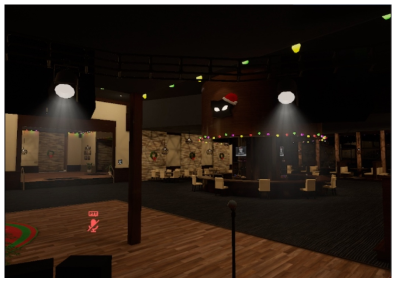


Figure 4.12: BC3

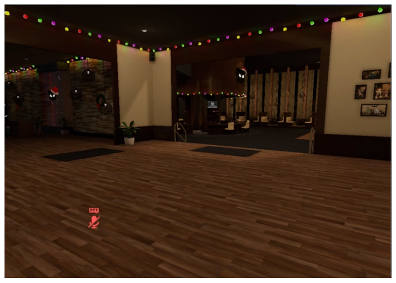


Figure 4.13: BC4

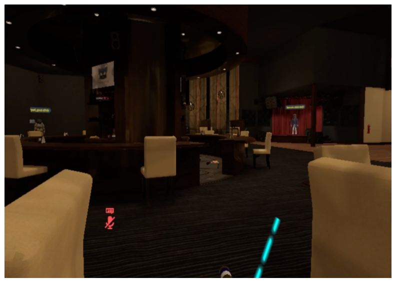


Figure 4.14: BC5

Figure 4.15: Adversarial viewpoints.

Table 4.2: Legend of camera viewpoints used in Section 4.7.

Home	Legend	Black Cat	Legend
Static Camera 1	HC1	Static Camera 1	BC1
Static Camera 2	HC2	Static Camera 2	BC2
Static Camera 3	HC3	Static Camera 3	BC3
Static Camera 4	HC4	Static Camera 4	BC4
Mobile Camera	HC5	Mobile Camera	BC5
Combined	HCC	Combined	BCC

Table 4.3: Background details of the 35 participants.

Gender	
14 Female	21 Male
Dominant Hand	
2 Left	33 Right
VR Familiarity	
11 Slightly	24 Moderately-Extremely
Prior VR Experience	
5 Never Used VR Before	30 Used VR Before

4.6.2 Adversarial Viewpoint

We continuously observe and record the participants' avatar (Figure 4.9) in the virtual world by means of five different virtual camera positions, where each camera position represents a different adversarial viewpoint. Four of these camera positions are static and positioned at different corners of the virtual room (Figure 4.9), each of which represents the fixed (or static) position of an adversarial avatar observing the target participant from that position. The fifth camera is mobile, and represents the view of an adversarial avatar moving and navigating in the proximity of the (target) participant's avatar. We carried out our experiments in two different virtual worlds – one in a public world (called Black Cat) where other real users' avatars may be present, and second in a private world (called Home) where access is restricted to a select group of users. We refer to these five adversarial viewpoints in these two worlds by means of a legend outlined in Table 4.2. In our evaluation (Section 4.7), we will also analyze the effect of combining these five viewpoints on the accuracy of activity classification (where the viewpoints are referred to as HCC and BCC for Home and Black Cat, respectively).

4.6.3 Participants

Between August and December of 2022, we recruited 64 participants for test data collection. However, due to various personal, technical, and medical factors, only 35 of them completed the study and whose data is included in our evaluation. The participants' ages were between 18 and 48, with a median age of 19. Additional demographic and other details about our participants are listed in Table 4.3. All participants were appropriately compensated for their time and our study procedure was approved by our university's Institutional Review Board (IRB).

4.6.4 Data Collection Apparatus

VR Device and App. We utilize the Meta Quest 2 VR device[5] and the popular VRChat [38] app (installed on the Quest 2) for generating and collecting test data from the participants in our study. As of July 2022, VRChat had more than 200,000 daily active users and more than 7 million registered users [52]. Moreover, VRChat was one of the few VR apps which supported full-body avatars (instead of only the upper body) at the time we started our experiments. Although other popular apps later added integration of full-body avatars [48], the fundamental nature of data generation (and collection) does not significantly differ across a majority of the VR apps.

Motion Sensors. Participants' body motion was captured at 20 ms sampling interval on a smartwatch (TicWatch 2) worn by the participants on their wrist and on a smartphone (Moto G7 Play) placed in their pocket. 10 participants chose to wear the smartwatch on their right wrist, while the rest chose to wear it on their left wrist. 23 participants placed the smartphone in one of their front pockets, while the rest place it in one of their back pockets.

Data Logging. The VRChat app was installed on five different desktops to record the viewpoints/perspective of an adversary as described in Section 4.3, and OBS Studio [22] was used to record the each adversarial perspective into individual video files with timestamps. The motion sensors were logged in respective devices with timestamps, and later transferred to another desktop for analysis.

[5]https://www.meta.com/quest/products/quest-2

Analysis Computer. A 2021 MacBook Pro was used to train and classify activities, and also for the activity-based filtering and magnitude-based rankings. It is equipped with 10-Core M1 CPU, 16-Core GPU, 16GB memory, 1TB SSD storage, and 16-core Neural Engine. For our *large-scale* analysis in Section 4.8, we also used a desktop with Ryzen 5 3600 6-Core 3.6GHz CPU, RTX 3060 12GB GPU, 1TB SSD storage, and 16GB memory, to train and generate large datasets using CTGAN [47, 163].

4.7 Evaluation

We next evaluate the proposed correlation framework utilizing the test data collected from participants, which represents a *small-scale* attack with confusion set size of 271 (accumulating different motion sensor locations from individual participants). We start with identifying suitable framework parameter values such as the activity window size (w) and activity-based filtering threshold (t). After extensively evaluating the correlation framework in the *small-scale* setting, we also generate and evaluate a representative dataset for a *large-scale* correlation in Section 4.8.

4.7.1 Framework Parameters

Our correlation framework has two key parameters that are critical for the rest of our empirical evaluation. The first parameter is the activity window size (w), which is the time duration used to classify an action. The second parameter is the Hamming distance used as the activity-based filtering threshold (t), which is the minimum requirement for an activity-vector to be considered in the identity ranking. As the total observation time, and thus the number of observed activity windows, will vary between different target users, the activity-based filtering threshold (t) is normalized with respect to the number of observed activity windows. No filtering occurs when the filtering threshold is set at 100%, whereas at 0% even one mismatch in the activity sequence will result in that activity-vector being filtered out.

Figures 4.22 and 4.29 show the correlation accuracy, where "None Correlated" occurs when the activity-based filtering filters all candidate activity-vectors, "Incorrectly Correlated" occurs

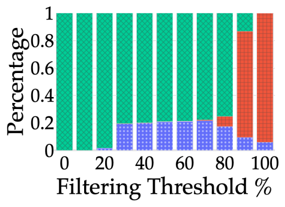


Figure 4.16: $w = 0.5s$

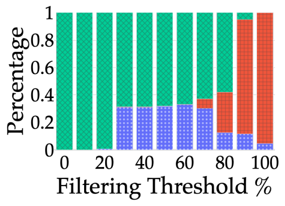


Figure 4.17: $w = 1s$

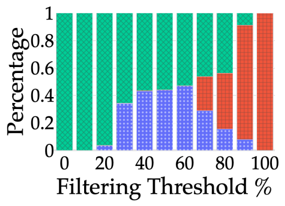


Figure 4.18: $w = 2s$

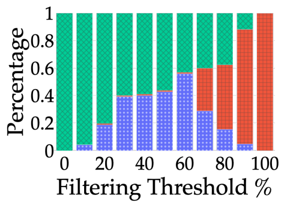


Figure 4.19: $w = 3$

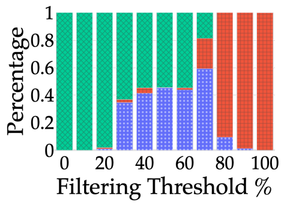


Figure 4.20: $w = 5s$

🟩 None Correlated 🟥 Incorrectly Correlated 🟦 Correctly Correlated

Figure 4.21: Legend

Figure 4.22: Right smartwatch motion sensor and visual movement data correlated with different w and normalized t parameters. Accuracy based on top-1 identity in the rankings.

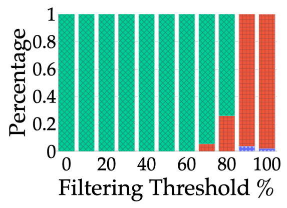


Figure 4.23: $w = 0.5s$

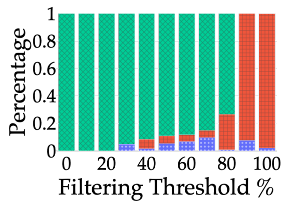


Figure 4.24: $w = 1s$

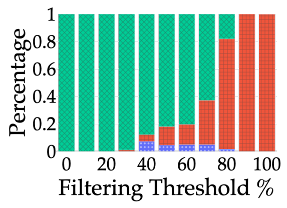


Figure 4.25: $w = 2s$

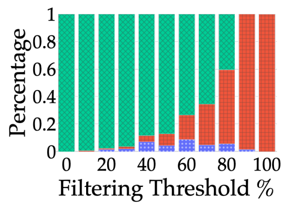


Figure 4.26: $w = 3s$

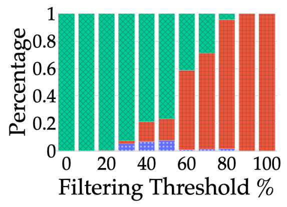


Figure 4.27: $w = 5s$

🟩 None Correlated 🟥 Incorrectly Correlated 🟦 Correctly Correlated

Figure 4.28: Legend

Figure 4.29: Front right pocket smartphone motion sensor and visual movement data correlated with different w and normalized t parameters. Accuracy based on top-1 identity in the rankings.

when the top ranked identity is incorrect, and "Correctly Correlated" occurs when the top ranked identity is correct. From these figures we can see an overall trend that as we increase w, the percentage of identities that passes the activity-based filtering and then used for identity ranking also grows. Conversely, the percentage of “None Correlated” is diminished as w is increased. This can primarily be attributed to (i) the size of activity sequence in the activity-vector is inversely proportional to w for a constant observation time period thereby reducing the number of probable mismatches, and (ii) the activity inference tends to perform more accurately for larger w.

While the above observation should compel us to select a larger w, in Figures 4.22 and 4.29 we also observe that there exists a trade-off between w and correctly correlated identities for different activity-based filtering thresholds. For instance, when $w = 5s$ we observe that the percentage of correctly correlated identities starts to decrease beyond the filtering threshold of 70% in Figure 4.20. This is most likely because as the size of activity-vector is reduced with increasing w, the probability of confusion with another person’s activity magnitudes is increased. This trend was consistent across other experimental variables, such as different adversarial viewpoints, different motion sensors, and different motion sensor positions on the body.

Based on empirical observations across different experimental variables, we set $w = 1s$ *and* $t = 30\%$ *for the rest of our analyses.* On average, these selected values are best suited for maximizing the percentage of correctly correlated identities. The average correctly correlated identities using these parameter values within top-1 of the ranking was 16.3%, and 17.0% of the identities were within top-3. In an alternate adversarial model where the motion sensor positions on the body is known to the adversary, more specific (i.e., per target user) w and t values can be selected to further improve the percentage of correctly correlated identities.

4.7.2 Activity Confusions

The accuracy of the activity classification models plays an important role in the correlation framework’s overall success rate. Activity classification between visual and motion sensor data differs significantly due to the modality (of input signal), and is potentially subject to different types of

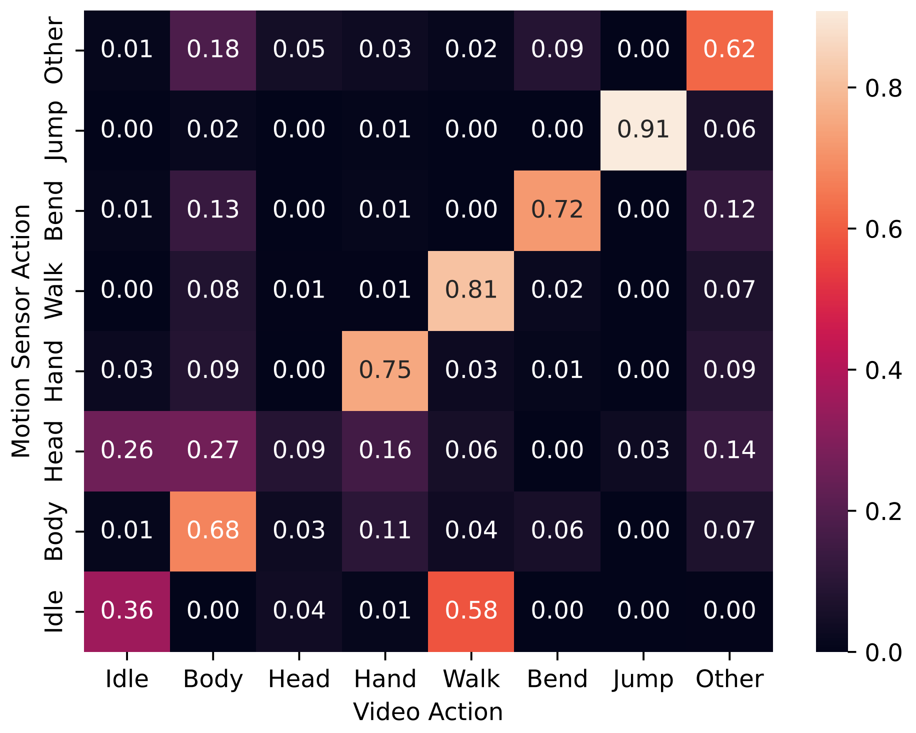


Figure 4.30: Using right wrist smartwatch.

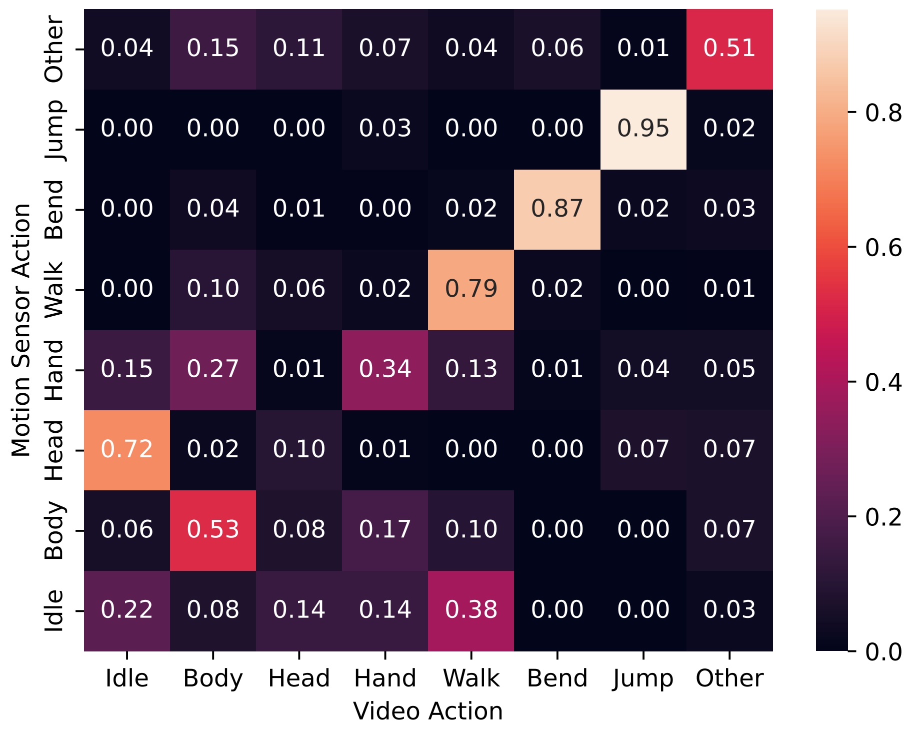


Figure 4.31: Using front right pocket smartphone.

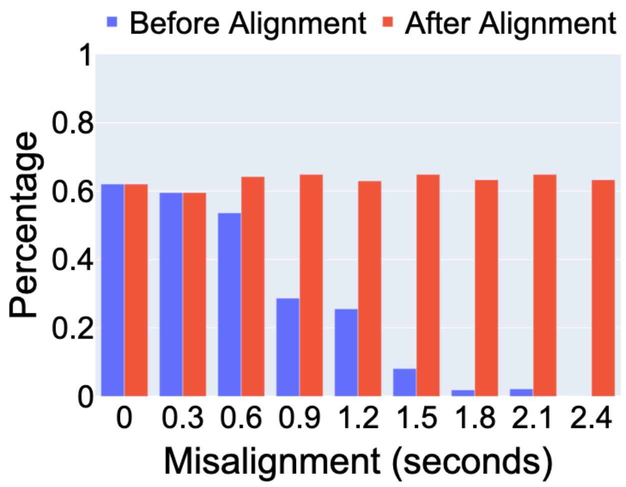


Figure 4.32: Correctly correlated accuracy (top-1 rank) with artificially introduced misalignment, shown for data from the right wrist.

noises and interference signals. Different adversarial viewpoint angles, distances, and occlusion levels affect the visual data classification. For instance, if only half of the avatar is visible due to being behind a coach or another avatar is in front of the target avatar, the chance of a misclassification is significantly increased. On the other hand, the positioning and orientation of the device used to collect motion sensor data also imposes certain limitations on the activity classification accuracy, especially as we assume that the adversary is unaware of the exact position of the motion sensor. For instance, if the motion sensor data is from a smartwatch worn on the right hand, it is very useful to classify activities involving the right hand, but may result in high misclassification of activities not involving the right arm.

Due to these apparent limitations, we analyze the direct consequence of misclassifications, i. e., the confusion of activities between the visual and motion sensor data. In Section 4.7.2, we observe that the *idle* activity has noticeably low accuracy (36% and 22% for right wrist smartwatch and front right pocket smartphone, respectively), and is often confused with other activities. An unexpected, yet clearly discernible, confusion exists between motion sensor *idle* and visual *walking*. One possible factor behind this observation is that VR users may be using the VR joystick to walk in the virtual world. As a result, the target user appears idle in the motion sensor data, while their virtual avatar is visually walking. Another noteworthy observation is that head movements had high confusion due to the fact that placement of motion sensors around hip and wrist areas is not suitable for capturing the target user's head movements, where as a head-mounted VR device is accurately able to capture head movements and apply them to the avatar in the virtual world.

In light of these insights, we further optimize our framework as follows. Rather than considering all the classified actions, we only utilize activities with less than 60% of confusion – body, hand, walk, bend, jump, and others – for our activity-based filtering. Remaining activities in the activity-vector are ignored from the Hamming distance calculations. The average correctly correlated identities after this optimization within top-1 of the ranking was 37.3%, while 38.7% of the identities were within top-3.

4.7.3 Time Alignment

Both the visual and motion sensor data are collected with device timestamps for synchronization. Although most modern smartphones and smartwatches are by default periodically updated against Internet-based time servers, motion sensor data collection in the wild may contain time drift errors and thus misaligned with the visual movements. Misaligned data sources will likely cause confusion between classified activities, resulting in a high failure rate in satisfying the activity-based filter threshold. As shown in Figure 4.32, misaligned data can drop a 62.1% correctly correlated result down to 0% in the presence of only 2.4 seconds (of artificially introduced) misalignment. The adversary can potentially detect and overcome such misalignments by offsetting the (motion sensor) data in increments, and selecting a time offset ($\pm\delta$) that results in the minimum Hamming distance in the activity-based filtering. Realistic assumptions must be made on the bounds of δ in order to keep the computational time practical.

4.7.4 Different Motion Sensors and Camera Locations

We next detail how different positions of the motion sensor on the (human) body and different adversarial viewpoints affect the correct correlation of our proposed framework. Overall, smartwatch (motion sensor) on left or right wrist performed better than the smartphone in the hip pockets (Fig-

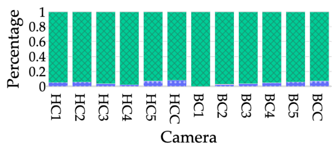


Figure 4.33: Motion sensor in back left pocket

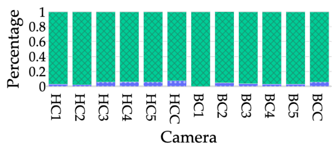


Figure 4.34: Motion sensor in back right pocket.

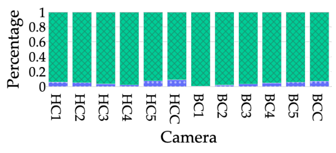


Figure 4.35: Motion sensor in front left pocket.

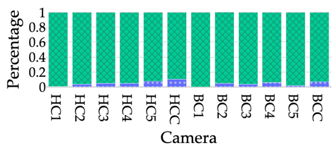


Figure 4.36: Motion sensor in front right pocket.

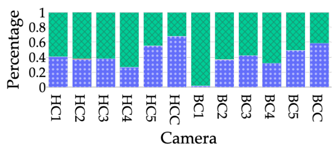


Figure 4.37: Motion sensor on right wrist.

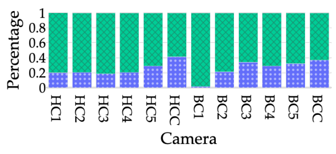


Figure 4.38: Motion sensor on left wrist.

🟩 None Correlated 🟥 Incorrectly Correlated 🟦 Correctly Correlated

Figure 4.39: Legend

Figure 4.40: Accuracy for different cameras positions and motion sensors locations of devices during the free-movement phase. Accuracy based on top-1 identity in the rankings.

ure 4.40). For example, for the *Home* world, the smartwatch yielded about 41% and 68% correct correlations (top-1 rank), for left and right wrists, respectively. In contrast, the front left-front pocket smartphone data resulted in about 9.1% correct correlations, while other smartphone locations are in a similar range. Intuitively, one of the main factors behind this observation is the inability of smartphone motion sensors to pick up hand movements when they are located in the hip area pockets. This causes higher confusion between activities (Section 4.7.2), resulting in the activity-vector of the target user being filtered out with high likelihood.

As far as the impact of different adversarial viewpoints on the correlation accuracy of our framework is concerned, we can see from Figure 4.40 that, except for BC1, all other camera locations (or adversarial viewpoints) yielded comparable results within each of the motion sensor locations. The reason behind BC1 performing particularly poor is that its location was near the entrance point of the *Black Cat* world and most participants eventually moved away from the field-of-view of this camera during the data collection experiments. In summary, combining multiple viewpoints and the availability of wrist-based motion sensor data are the most favorable conditions for the adversary.

4.7.5 Conflicting Activity Sequences

There can be cases, especially in a *large-scale* attack, where multiple target users perform a similar or even an identical sequence of activities. In such cases, the magnitude-based ranking should ideally still rank the real identity higher than others. In this part of our analysis, we study the extent to which our magnitude-based ranking is able to do so, by comparing correlation accuracy when participants (and their avatars) performed the same sequence of activities. In Figure 4.42, we observe 16.5% correct correlation for motion data from the right wrist in top-1 of identity rankings and 50.1% correct correlation within the top-3 ranks. This demonstrates that, to an extent, the magnitude-based ranking is in fact able to discern the difference between identities based on the magnitude of movements.

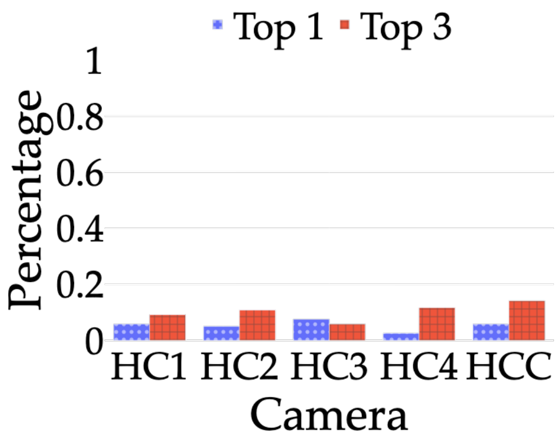


Figure 4.41: Front left pocket motion data

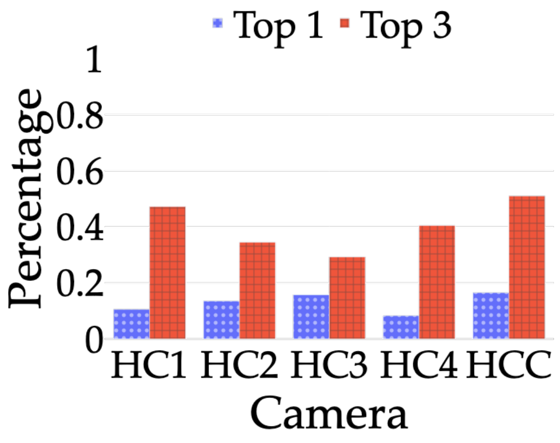


Figure 4.42: Right wrist motion data

Figure 4.43: Identity correlation for conflicting activity sequences.

4.8 Optimizing for Large-scale Attacks

An adversary trying to correlate thousands or millions of anonymous avatars with identified motion sensors data is presented with a very significant computational task. In this section, we analyze the computational complexity of this task and propose related optimizations to our correlation framework.

Synthetic Data Generation To test the scalability of our framework, we must first generate a very large synthetic dataset *utilizing* real participant data collected in Section 4.5. While it was not feasible for us to collect real-world data from thousands or millions of participants, due to the time and resources required for systematic data collection per participant, we still want to test using a

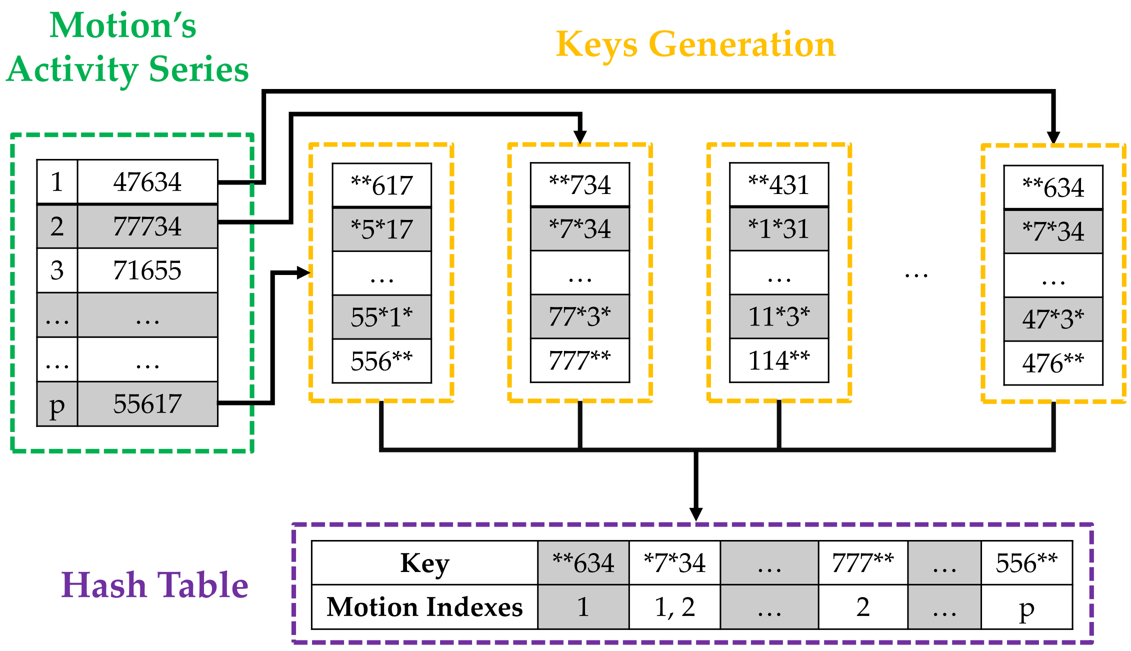


Figure 4.44: Generating a hash table from the motion activity series.

dataset that has resemblance to the *small-scale* dataset instead of generating completely random activity-vectors. The activity classification and magnitude calculation tasks take constant time, and will grow linearly with the size of each dataset (p and q, for visual movement and motion sensor datasets, respectively). For large p and q, the more complex task is that of calculating the correlation of all q identities against all p anonymous avatars. However, as seen in Section 4.7, the activity-based filtering is very effective in reducing the complexity of the magnitude-based identity rankings. Therefore, for large p and q the most computationally complex task in the entire framework comes down to the activity-based filtering. Accordingly, we generate our *large-scale* dataset to test the scalability of our activity-based filtering, which only requires activity sequences as input. Our first *large-scale* dataset was generated using a modern tabular Generative Adversarial Network (GAN) technique [47], called CTGAN [163], which is trained using activity sequences from real participants, as outlined in Section 4.5. Our second *large-scale* dataset was generated using random permutations of our activity sequences from Section 4.5. Each of these *large-scale* datasets contained 1 million activity sequences for the motion sensor and 1 million activity sequences for the visual movements.

Activity-based Filtering Without Optimizations. Without any optimizations, the activity-based filtering has a time complexity of $O(pqk^2)$, where p is the number of unique avatars from the visual movement data, q is the number of different identities from the motion sensor data, and k is the size of the activity sequences. As such, we can further assume that increasing the size of k would have diminishing returns (computationally), making it less attractive for an adversary to record each target for too long. Therefore, we assume k would not be scaled, unlike p and q, and treat k as constant, thus resulting with a complexity of $O(pq)$. As shown in Table 4.5, our setup takes $2.2 * 10^1$ ms to finish activity-based filtering when $p = q = 100$. However, when we scale up to $p = q = 10^5$, it requires $3.15 * 10^7$ ms (or about 8 hours) to finish activity-based filtering. We estimate that for $p = q = 10^6$, it will take approximately 30 days to finish, and about 3000 days when $p = q = 10^7$, which is not very scalable.

Optimization. We propose the use of a hash table to store our activity sequence data in order to reduce the time complexity of activity matching and filtering. However, as even a single mismatch between two activity sequences will result in completely different hash values (i.e., the keys in a hash table), we design a larger hash table that allows for some degree of mismatch. Specifically, we populate a hash table with keys based on permutations of the q activity sequences in M (each of length k) from the motion sensors data, accounting for possible errors allowable within the Hamming distance threshold (t). Let us assume that the numbers 0 to 7 denotes each of the eight activities we classify. If $k = 5$, an example of the activity string would be $\langle 47634 \rangle$. If our hamming distance threshold is $t = 2$, then any two activities can be mismatched and still pass the threshold. Now, assume the character $*$ as a wildcard activity that may or may not be a match. To populate the hash table exhaustively, we compute every possible permutation of each activity sequence in M including up to two $*$. For our previous example, $\langle 47634 \rangle$, some of the permutations generated would be $\langle * * 634 \rangle$, $\langle 4 * 6 * 4 \rangle$, and $\langle 47 * 3 * \rangle$. All these permutations are then used as the key in our hash table, while the corresponding value is the identity of users from the motion sensor data (M) (an example in Figure 4.44). Thereafter, during the correlation process, each activity sequence from the video dataset also undergoes permutations with up to two $*$, and then queried

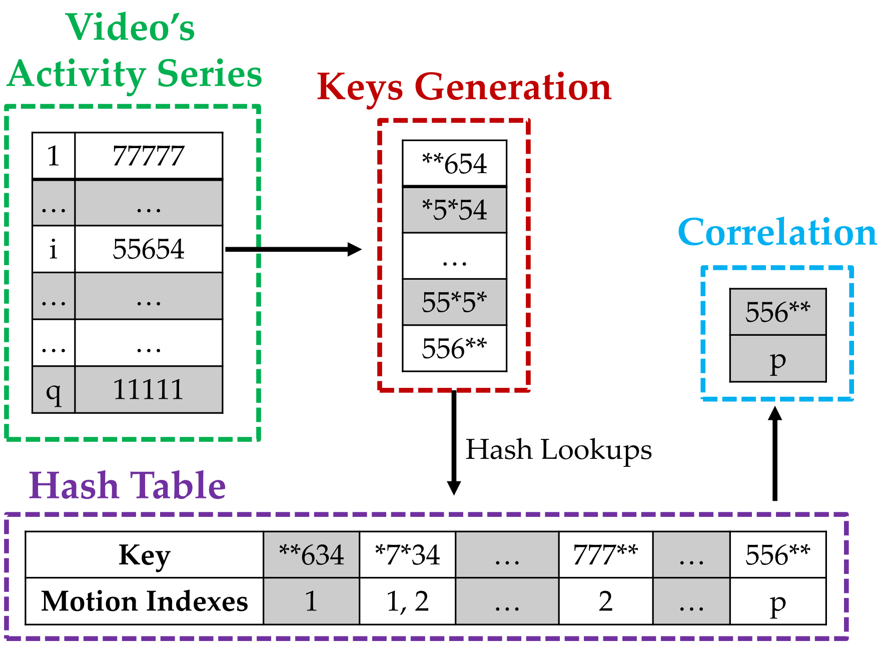


Figure 4.45: Matching video activity series to a potential motion activity series using the hash table generated from Figure 4.44.

against the above hash table for a match. If a matching key exists, the corresponding identity and activity-vector has satisfied the activity-based filtering and is included in the identity ranking, as depicted with an example in Figure 4.45).

Optimized Performance Analysis. The number of permutations per activity-vector does not scale with the size of datasets and thus can be treated as $O(1)$ time complexity. Similarly, hash table search and insertion is $O(1)$ time complexity. Therefore, with the use of our hash table, the new time complexity becomes $O(p+q)$, where $O(q)$ time is required to create the hash table, and $O(p)$ time is require to iterate through V for filtering.

Table 4.4: Computational analysis for permutation generated datasets, for default and optimized activity-based filtering. Tested for $k = 5$ with $t = 2$, and $k = 10$ with $t = 3$. Entries marked as "–" did not finish.

Permutation Generated	Average Collision (#)			Average Correctly Correlated (%)		
Motion × Video	Default	Hash 3/5	Hash 7/10	Default	Hash 3/5	Hash 7/10
100 × 100	0.5	0.5	0.5	54.0	49.0	49.0
500 × 500	0.6	1.0	0.9	55.0	42.0	40.0
1000 × 1000	1.3	17.1	1.3	48.9	41.1	43.1
10000 × 10000	4.5	242.6	7.6	53.2	40.9	36.2
100000 × 100000	7.2	2314.1	12.6	51.7	36.6	29.8
1000000 × 1000000	-	22100.0	23.9	-	38.7	24.5

Our empirical results (Tables 4.4 and 4.5) show that with this optimization, the activity-filtering is significantly faster. For instance with $p = q = 100000$, $k = 10$, and $t = 3$, using the optimization technique was 575 times faster than the default activity-based filter. Another important aspect we also evaluate in the empirical results is the number of collisions, which occur when multiple unique data sources satisfies the activity-matching threshold. For $k = 5$, we observe 1594.87 average number of collisions, and for $k = 10$, we observe only 4.26 average collisions (with $p = q = 100000$). This implies that if an adversary observes a very high number of collisions, the adversary should increase the value of k.

4.9 Discussion

Next, we highlight some limitations and interesting observations that we made during our experiments, which may need to be considered by an adversary carrying out the above de-anonymization attack. Further, we also list some additional adversarial optimizations that could be applied to the proposed framework and identify potential mitigation strategies against this threat.

Dataset Limitations and Improvements. Our framework and data collection assumes time synchronization across devices (VR headset and smartphone/smartwatch). While most smart devices are periodically synchronized with Internet-based time servers, they can potentially accumulate minor drift errors between time synchronizations. The adversary may have to use a larger alignment window to account for such time drift errors. Although our test dataset from the 35 participants is limited and does not include all the population demographics, our framework utilizes components such as the CMU MoCap dataset, which has been studied comprehensively in the literature [64, 134] and has been validated for its generalizability.

Other Limitations and Improvements. During out experiments, we observed random objects, for example, a tent (Figure 4.47) and a meteoroid (Figure 4.46), being spawned arbitrarily and at random locations within the VRChat worlds. While the reason for these arbitrary objects appearing was unclear, depending on their location, they could interfere with the adversary's viewpoint by blocking his (visual) line-of-sight to the target. In addition to randomly appearing stationary ob-

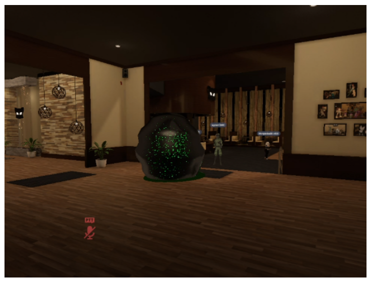


Figure 4.46: Meteoroid

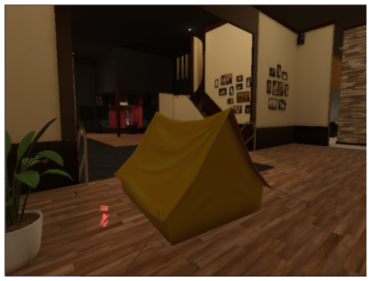


Figure 4.47: Tent

Figure 4.48: Examples of object spawning.

jects, we have also sometimes observed arbitrary appearances of moving non-playable characters which can also impact the adversary's view of the target. In summary, an adversary should plan for such arbitrary obstructions during visual data collection, and perhaps employ multiple viewpoints (or perspectives) to the target user in order to overcome this issue, similar to what we do in our experiments.

Image & Link Injections. Another challenge an adversary could face in the virtual world (especially, public worlds) while collecting visual data (corresponding to the target) is random and uninitiated interactions with other VR users. During our experiments, we observed random users positioning themselves in front of our adversary (and its view), thus blocking his line-of-sight (to the target) and impacting the attack. While a mobile adversary may be able to adjust his position (within the virtual room) to regain view of the target, a stationary adversary may be unable to do it and thus unable to record useful visual data for the attack. Other forms of interactions (by other users with our adversary) could include sharing of images and links, which could also disrupt the visual data recording by the adversary. For instance, during our experiments we observed that when an image is shared (see Figure 4.49) by a VRChat user (with our adversary), it overlays a

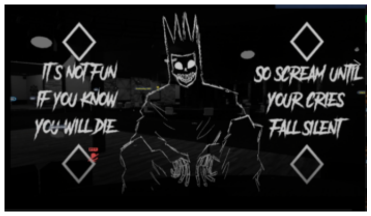
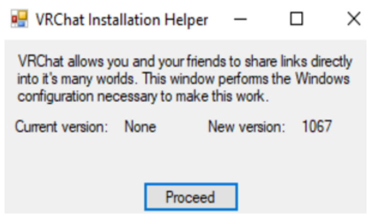

IT'S NOT FUN IF YOU KNOW YOU WILL DIE
SO SCREAM UNTIL YOUR CRIES FALL SILENT

Figure 4.49: Injected image.


VRChat Installation Helper

VRChat allows you and your friends to share links directly into it's many worlds. This window performs the Windows configuration necessary to make this work.

Current version: None New version: 1067

Proceed

Figure 4.50: Injected link.

Figure 4.51: Examples of image/link injection.

transparent image on top of the adversary's viewpoint, rendering the visual data collected by him ineffective during that period. Similarly, we also observed that sharing of links can also have undesirable effects on the adversary's avatar (Figure 4.50), rendering it ineffective in collecting useful visual data.

Table 4.5: Computational analysis for GAN generated datasets, for default and optimized activity-based filtering. Tested for $k = 5$ with $t = 2$, and $k = 10$ with $t = 3$. Entries marked as "–" did not finish.

GAN Generated	Time (ms)			Average Collision (#)			Average Correctly Correlated (%)		
Motion $\times$ Video	Default	Hash 3/5	Hash 7/10	Default	Hash 3/5	Hash 7/10	Default	Hash 3/5	Hash 7/10
100 $\times$ 100	22.0	7.0	29.0	0.5	0.5	0.5	47.0	41.0	44.0
500 $\times$ 500	586.0	36.0	177.0	0.7	0.9	0.7	44.0	43.0	41.0
1000 $\times$ 1000	2415.0	80.0	386.0	1.1	19.5	1.9	45.8	39.4	40.1
10000 $\times$ 10000	267293.0	1020.0	4714.0	2.8	200.2	2.1	51.3	42.6	34.7
100000 $\times$ 100000	31481325.0	9617.0	54736.0	4.9	1594.9	4.3	39.1	31.1	21.5
1000000 $\times$ 1000000	-	99304.0	644596.0	-	16011.0	17.3	-	30.9	21.9

Detecting and Ousting Suspicious Avatars. The VRChat service employs an anti-cheat software which attempts to detect bots, inactive avatars, and avatars who misuse VRChat terms of services and kicks them out or bans them from the service. During our experiments, we did observe that some of our adversarial avatars, especially stationary avatars, were kicked out of the room (being monitored) or even banned altogether from VRChat. Although the main reasons (could be the anti-cheat software or other users reporting our adversarial avatars) behind such kick-outs or bans are unclear to us, we believe this could present a significant obstacle to an adversary attempting to accomplish the proposed attack. In order to continue collecting visual data in the presence of such room kick-outs and bans, an adversary would need to find ways to circumvent such “anti-cheat" measures or be ready to deploy backup avatars, similar to what we did during our experiments.

Additional Optimization. In addition to the optimizations we presented earlier in Section 4.8, an adversary can carry out additional optimizations as part of the framework to improve the overall accuracy by further reducing the number of incorrect correlations. For example, suppose that an adversary has collected visual and motion data over multiple sessions/days. It is highly unlikely that a correlation between motion data of two (or more) unique people/users to a target avatar will repeat over a span of multiple independent observed virtual reality sessions. To utilize this factor, the adversary has to first increase the activity-based filtering threshold (t) for all the observed sessions/days. With a higher allowable mismatch between the activity sequences, the adversary is more likely to include the target user’s identity in the rankings across all of the sessions. Thereafter, with elimination of identities not present in rankings of all the sessions, the combined ranking/search set will reduce drastically, increasing the probability of correct correlation.

Active Mitigation Measures. The best mitigation for the de-anonymization attack presented in this work is to fully decouple the visual and motion sensor data by not making the motion data available to the adversary when users are in virtual environments. This can be accomplished through various means such as increasing user awareness of such threats, not wearing/carrying smart mobile devices (with in-built motion sensors) while using VR services or through appropriate user-notifications at the beginning of VR sessions. If the smart mobile device(s) is synced with the VR device, access to the mobile device motion sensor could also be automatically and appropriately regulated while the user is in a virtual reality session. Alternatively, another option to protect against such attacks would be to use non-humanoid avatars or a humanoid avatar with adversarial patches [150]. Adversarial patches typically overlay an image patch on a target image object (in our case, an avatar), causing some pre-trained machine learning or deep learning classifier into misclassifying the object. An appropriate adversarial patch on the user’s chosen avatar would prevent recognition of the humanoid character in our framework, thus preventing accurate generation of the activity-vector series required for correlation.

4.10 Conclusion

We proposed a novel framework to correlate anonymous avatars in virtual worlds with identified out-of-band motion sensor data. Our work highlights a newfound privacy risk to users of the growing VR ecosystem. Specifically, VR users can be vulnerable to de-anonymization attack if they carry a smartphone or wear a smartwatch while using a VR system. Our evaluation of the proposed framework is a step towards demonstrating the feasibility of such an attack, utilizing real-world data from human participants. Through our empirical analyses, we were able to optimize framework parameters, improve scalability, and identified current limitations and potential for further improvements.

4.11 Acknowledgment

Research reported in this publication was supported by the US NSF under award number 1943351 and by UTSA VPREDKE's Grant for Research Advancement and Transformation (GREAT).

4.12 Publication

As of the writing of this dissertation, this chapterâs research is currently undergoing peer-review at a security conference.

CHAPTER 5: CONCLUSION

The first part of this dissertation focused on investigating the privacy implications of activity during video calls, specifically with regards to the exposure of shoulders and arms through the user's camera. A novel attack framework was proposed, which mapped arm movements to predict words from a dictionary, and was comprehensively evaluated. The findings demonstrated the feasibility of this framework under different conditions, including video-calling software, clothing, type of words, keyboards, and webcams. These results highlight the need for users to be cautious about what can be seen during a video call to safeguard their privacy.

The second part of this dissertation aimed to explore the privacy of context during video calls. It was revealed that virtual background software, which is often used as a privacy measure, may not effectively protect a person's privacy during a video call. The research demonstrated that adversaries can exploit vulnerabilities in virtual backgrounds to reconstruct the real background, detect objects in the background, and even infer the user's location.

The third part of this dissertation focused on studying the privacy of identity in virtual reality. It was found that wearing out-of-band motion sensor devices, such as smartwatches, while in a virtual world can compromise a user's identity. To address this issue, a novel framework that correlates visual data from the virtual world with real-world motion data was proposed. The feasibility of this attack highlights the importance of users being cautious about carrying motion sensors while in a virtual world to prevent potential leaks of their identity.

BIBLIOGRAPHY

[1] 10 million password list top 1000000. https://gitlab.com/kalilinux/packages/seclists/-/blob/kali/master/Passwords/Common-Credentials/10-million-password-list-top-1000000.txt. [Online; accessed 9-Sep-2020].

[2] 20 Astonishing Video Conferencing Statistics for 2021. https://digitalintheround.com/video-conferencing-statistics/. [Online; accessed 28-Nov-2021].

[3] 500,000 Hacked Zoom Accounts Given Away For Free On The Dark Web. https://www.forbes.com/sites/leemathews/2020/04/13/500000-hacked-zoom-accounts-given-away-for-free-on-the-dark-web/. [Online; accessed 22-May-2020].

[4] AudioSet - A large-scale dataset of manually annotated audio events. https://research.google.com/audioset/. [Online; accessed 22-May-2020].

[5] Cisco Webex vulnerabilities may enable attackers to covertly join meetings. https://www.helpnetsecurity.com/2020/11/19/cisco-webex-vulnerabilities-attackers-covertly-join-meetings/. [Online; accessed 29-Nov-2021].

[6] Cmu graphics lab motion capture database. http://mocap.cs.cmu.edu. [Online; accessed 01-Nov-2022].

[7] Coronavirus Disease 2019 (COVID-19) - Cases, Data, and Surveillance. https://www.cdc.gov/coronavirus/2019-ncov/cases-updates/index.html. [Online; accessed 22-May-2020].

[8] Coronavirus prompts increased use of video chat platforms for work, connection. https://www.ipsos.com/en-us/news-polls/coronavirus-prompts-increased-use-of-video-chat. [Online; accessed 11-March-2021].

[9] Google Blocks. https://arvr.google.com/blocks. [Online; accessed 01-Nov-2022].

[10] Google Hangouts. https://tools.google.com/dlpage/hangoutplugin. [Online; accessed 22-May-2020].

[11] Google Meet. https://meet.google.com/. [Online; accessed 28-Nov-2021].

[12] ImageNet dataset. https://image-net.org/index.php. [Online; accessed 28-Nov-2021].

[13] Introducing tiled view, and other top-requested features in Google Meet. https://cloud.google.com/blog/products/productivity-collaboration/introducing-tiled-view-and-other-top-requested-features-in-google-meet. [Online; accessed 22-May-2020].

[14] livechat. https://www.livechat.com/typing-speed-test//global-scores. [Online; accessed 9-Sep-2020].

[15] ManyCam. https://manycam.com/. [Online; accessed 22-May-2020].

[16] MeetinVR. https://www.meetinvr.com. [Online; accessed 01-Nov-2022].

[17] Metaverse. https://about.meta.com/metaverse. [Online; accessed 01-Nov-2022].

[18] Microsoft Teams. https://teams.microsoft.com. [Online; accessed 28-Nov-2021].

[19] Mozilla Hubs. https://hubs.mozilla.com. [Online; accessed 01-Nov-2022].

[20] MrDeepFakes. https://mrdeepfakes.com. [Online; accessed 01-Nov-2022].

[21] New National Survey: Video Conferencing's Role in Enterprise Now and in a Post-COVID World. https://mediasite.com/blog/new-national-survey-video-conferencings-role-in-enterprise-now-and-in-a-post-covid-world/. [Online; accessed 28-Nov-2021].

[22] Obs studio. https://obsproject.com. [Online; accessed 01-Nov-2022].

[23] ocrnet-hrnet-w48-paddle. https://docs.openvino.ai. [Online; accessed 01-Nov-2022].

[24] Oculus Store. https://www.oculus.com/experiences/quest/. [Online; accessed 01-Nov-2022].

[25] Open Broadcaster Software. https://obsproject.com/. [Online; accessed 28-Nov-2021].

[26] Rewatch. https://rewatch.com/. [Online; accessed 29-Nov-2021].

[27] See No Evil, Hear No Evil: The Use of Deepfakes in Social Engineering Attacks. https://www.tripwire.com/state-of-security/use-of-deepfakes-in-social-engineering-attacks. [Online; accessed 01-Nov-2022].

[28] Sketchfab. https://sketchfab.com. [Online; accessed 01-Nov-2022].

[29] Skype. https://www.skype.com/en/. [Online; accessed 22-May-2020].

[30] Spatial. https://www.spatial.io. [Online; accessed 01-Nov-2022].

[31] The Majestic Million top 1 million websites. https://majestic.com/reports/majestic-million. [Online; accessed 9-Sep-2020].

[32] Twitch. https://www.twitch.tv/. [Online; accessed 22-May-2020].

[33] Unity real-time development platform. https://unity.com. [Online; accessed 01-Nov-2022].

[34] Use a virtual background on Zoom calls for better privacy. https://www.totaldefense.com/security-blog/use-a-virtual-background-on-zoom-calls-for-better-privacy/. [Online; accessed 28-Nov-2021].

[35] Using Zoom Backgrounds to Keep Your Privacy and Clean Up Your Space. https://www.multibrain.net/using-zoom-backgrounds-to-keep-your-privacy-and-clean-up-your-space/. [Online; accessed 28-Nov-2021].

[36] Vicodo video calls archive. https://www.vicodo.com/features/archiving. [Online; accessed 29-Nov-2021].

[37] Video call. https://www.computerhope.com/jargon/v/video-call.htm. [Online; accessed 28-Nov-2021].

[38] VRChat. https://hello.vrchat.com/. [Online; accessed 01-Nov-2022].

[39] Why Video Chat is the New Normal for Millennials and Gen Z. https://talkative.uk/info/why-video-chat-is-new-normal-for-millennials-gen-z/. [Online; accessed 22-May-2020].

[40] Wictionary top 100,000 most frequently-used English words [for john the ripper]. https://gist.github.com/h3xx/1976236. [Online; accessed 22-May-2020].

[41] WIDER FACE: A Face Detection Benchmark. http://shuoyang1213.me/WIDERFACE/. [Online; accessed 22-May-2020].

[42] YouTube. https://www.youtube.com/. [Online; accessed 22-May-2020].

[43] Zoom. https://zoom.us/. [Online; accessed 22-May-2020].

[44] Zoom Optimizes Noise Cancellation for Top-Notch Audio. https://blog.zoom.us/wordpress/2018/08/21/zoom-adds-noise-cancellation-to-our-top-notch-audio/. [Online; accessed 22-May-2020].

[45] ‘Zoombombing’ Becomes a Dangerous Organized Effort. https://www.nytimes.com/2020/04/03/technology/zoom-harassment-abuse-racism-fbi-warning.html. [Online; accessed 22-May-2020].

[46] 7 Great Virtual Reality Travel Experiences. https://www.lifewire.com/virtual-reality-tourism-4129394, Online; accessed 01-Nov-2022.

[47] GAN-for-tabular-data. https://github.com/Diyago/GAN-for-tabular-data, Online; accessed 01-Nov-2022.

[48] Legs are finally coming to Mark Zuckerberg’s metaverse. https://www.vox.com/recode/2022/10/11/23399439/metaverse-mark-zuckerberg-connect-avatar-legs-meta-microsoft-apple-vr-ar, Online; accessed 01-Nov-2022.

[49] Meta Horizon. https://www.oculus.com/FacebookHorizon, Online; accessed 01-Nov-2022.

[50] Virtual Reality in Social Networks - What is This Phenomenon? https://servreality.com/blog/virtual-reality-in-social-networks-what-is-this-phenomenon/, Online; accessed 01-Nov-2022.

[51] VR for Tourism. https://immersionvr.co.uk/about-360vr/vr-for-tourism/, Online; accessed 01-Nov-2022.

[52] VRChat. https://mmostats.com/game/vrchat, Online; accessed 01-Nov-2022.

[53] Devon Adams, Alseny Bah, Catherine Barwulor, Nureli Musaby, Kadeem Pitkin, and Elissa M Redmiles. Ethics emerging: the story of privacy and security perceptions in virtual reality. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, pages 427–442, 2018.

[54] Kamran Ali, Alex X Liu, Wei Wang, and Muhammad Shahzad. Keystroke recognition using wifi signals. In *International Conference on Mobile Computing and Networking (MobiCom)*, 2015.

[55] Christoph Amma, Marcus Georgi, and Tanja Schultz. Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3d-space handwriting with inertial sensors. In *ACM International Symposium on Wearable Computers (ISWC)*, 2012.

[56] Christoph Amma, Marcus Georgi, and Tanja Schultz. Airwriting: A wearable handwriting recognition system. *Springer Personal and Ubiquitous Computing*, 18(1), 2014.

[57] S Abhishek Anand and Nitesh Saxena. Keyboard emanations in remote voice calls: Password leakage and noise(less) masking defenses. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, 2018.

[58] Luca Ardüser, Pascal Bissig, Philipp Brandes, and Roger Wattenhofer. Recognizing text using motion data from a smartwatch. In *IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, 2016.

[59] Dmitri Asonov and Rakesh Agrawal. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy (S&P)*, 2004.

[60] Michael Backes, Tongbo Chen, Markus Duermuth, Hendrik P. A. Lensch, and Martin Welk. Tempest in a teapot: Compromising reflections revisited. In *IEEE Symposium on Security and Privacy (S&P)*, 2009.

[61] Michael Backes, Tongbo Chen, Markus Dürmuth, Hendrik PA Lensch, and Martin Welk. Tempest in a teapot: Compromising reflections revisited. In *2009 30th IEEE Symposium on Security and Privacy*, pages 315–327. IEEE, 2009.

[62] Michael Backes, Markus Dürmuth, and Dominique Unruh. Compromising reflections-or-how to read lcd monitors around the corner. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 158–169. IEEE, 2008.

[63] Davide Balzarotti, Marco Cova, and Giovanni Vigna. Clearshot: Eavesdropping on keyboard input from video. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 170–183. IEEE, 2008.

[64] Mathieu Barnachon, Saïda Bouakaz, Boubakeur Boufama, and Erwan Guillou. Ongoing human action recognition with motion capture. *Pattern Recognition*, 47(1):238–247, 2014.

[65] Yigael Berger, Avishai Wool, and Arie Yeredor. Dictionary attacks using keyboard acoustic emanations. In *ACM CCS*, 2006.

[66] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[67] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65, 2005.

[68] Antoni Buades, Bartomeu Coll, and Jean Michel Morel. On image denoising methods. *CMLA Preprint*, 5, 2004.

[69] Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pages 671–679. Elsevier, 1987.

[70] Liang Cai and Hao Chen. {TouchLogger}: Inferring keystrokes on touch screen from smartphone motion. In *6th USENIX Workshop on Hot Topics in Security (HotSec 11)*, 2011.

[71] Liang Cai and Hao Chen. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In *USENIX Conference on Hot Topics in Security*, 2011.

[72] Stuart K Card. *The psychology of human-computer interaction*. Crc Press, 1983.

[73] Loren Carpenter. The a-buffer, an antialiased hidden surface method. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 103–108, 1984.

[74] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[75] Yimin Chen, Tao Li, Rui Zhang, Yanchao Zhang, and Terri Hedgpeth. Eyetell: Video-assisted touchscreen keystroke inference from eye movements. In *IEEE Symposium on Security and Privacy (S&P)*, 2018.

[76] Girija Chetty and Matthew White. Body sensor networks for human activity recognition. In *2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 660–665. IEEE, 2016.

[77] Alberto Compagno, Mauro Conti, Daniele Lain, and Gene Tsudik. Don't skype & type! acoustic eavesdropping in voice-over-ip. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017.

[78] PaddlePaddle Contributors. Paddleseg, end-to-end image segmentation kit based on paddlepaddle. https://github.com/PaddlePaddle/PaddleSeg, 2019.

[79] Foad Dabiri, Navid Amini, Mahsan Rofouei, and Majid Sarrafzadeh. Reliability-aware optimization for dvs-enabled real-time embedded systems. In *Proc. of the 9th int'l symposium on Quality Electronic Design*, pages 780–783, 2008.

[80] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang. The tangled web of password reuse. In *NDSS*, number 2014, pages 23–26, 2014.

[81] Erhan Davarci, Betul Soysal, Imran Erguler, Sabri Orhun Aydin, Onur Dincer, and Emin Anarim. Age group detection using smartphone motion sensors. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 2201–2205. IEEE, 2017.

[82] Fernando De la Torre, Jessica Hodgins, Adam Bargteil, Xavier Martin, Justin Macey, Alex Collado, and Pep Beltran. Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. 2009.

[83] Douglas DeCarlo and Dimitris Metaxas. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 231–238. IEEE, 1996.

[84] Nicholas Diakopoulos and Deborah Johnson. Anticipating and addressing the ethical implications of deepfakes in the context of elections. *New Media & Society*, 23(7):2072–2098, 2021.

[85] Lijun Ding and Ardeshir Goshtasby. On the canny edge detector. *Pattern Recognition*, 34(3):721–725, 2001.

[86] Guo Freeman, Samaneh Zamanifard, Divine Maloney, and Alexandra Adkins. My body, my avatar: How people perceive their avatars in social virtual reality. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–8, 2020.

[87] Saumya Gupta, Theresa Jean Tanenbaum, Meena Devii Muralikumar, and Aparajita S Marathe. Investigating roleplaying and identity transformation in a virtual reality narrative experience. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[88] T. Halevi and N. Saxena. A closer look at keyboard acoustic emanations: Random passwords, typing styles and decoding techniques. In *ACM Symposium on Information, Computer and Communications Security*, 2012.

[89] Jun Han, Albert Jin Chung, and Patrick Tague. Pitchln: eavesdropping via intelligible speech reconstruction using non-acoustic sensor fusion. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 181–192, 2017.

[90] Jun Han, Emmanuel Owusu, Le T Nguyen, Adrian Perrig, and Joy Zhang. Accomplice: Location inference using accelerometers on smartphones. In *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*, pages 1–9. IEEE, 2012.

[91] Duncan Hodges and Oliver Buckley. Reconstructing what you said: Text inference using smartphone motion. *IEEE Transactions on Mobile Computing*, 18(4):947–959, 2018.

[92] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[93] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4507–4515, 2017.

[94] Daesung Jang, Joon-Seok Kim, Ki-Joune Li, and Chi-Hyun Joo. Overlapping and synchronizing two worlds. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 493–496, 2011.

[95] Edmund Y.S.ChaoNozomuInoueFrank J.FrassicaJohn J.Elias. Image-based computational biomechanics of the musculoskeletal system. In *Handbook of Medical Image Processing and Analysis*, 2009.

[96] Gilbert B. Bluhm Andrew Brook John T. Sharp, Donald Y. Young. How many joints in the hands and wrists should be included in a score of radiologic abnormalities used to assess rheumatoid arthritis? 1985.

[97] Stamatis Karnouskos. Artificial intelligence in digital media: The era of deepfakes. *IEEE Transactions on Technology and Society*, 1(3):138–147, 2020.

[98] Jeong Ho Kim, Lovenoor Aulck, Michael C Bartha, Christy A Harper, and Peter W Johnson. Differences in typing forces, muscle activity, comfort, and typing performance among virtual, notebook, and desktop keyboards. volume 45. Elsevier, 2014.

[99] Jacob Leon Kröger, Otto Hans-Martin Lutz, and Florian Müller. What does your gaze reveal about you? on the privacy implications of eye tracking. In *IFIP International Summer School on Privacy and Identity Management*, pages 226–241. Springer, 2019.

[100] Markus G. Kuhn. Optical time-domain eavesdropping risks of crt displays. In *IEEE Symposium on Security and Privacy (S&P)*, 2002.

[101] Amit Kumar, Kristina Yordanova, Thomas Kirste, and Mohit Kumar. Combining off-the-shelf image classifiers with transfer learning for activity recognition. In *Proceedings of the 5th international Workshop on Sensor-based Activity Recognition and Interaction*, pages 1–9, 2018.

[102] Gen Li and Hiroyuki Sato. Handwritten signature authentication using smartwatch motion sensors. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1589–1596. IEEE, 2020.

[103] Mengyuan Li, Yan Meng, Junyi Liu, Haojin Zhu, Xiaohui Liang, Yao Liu, and Na Ruan. When csi meets public wifi: Inferring your mobile phone password via wifi signals. In *ACM CCS*, pages 1068–1079, 2016.

[104] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[105] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014.

[106] Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, and Kehuan Zhang. When good becomes evil: Keystroke inference with smartwatch. In *ACM CCS*, 2015.

[107] Chris Xiaoxuan Lu, Bowen Du, Hongkai Wen, Sen Wang, Andrew Markham, Ivan Martinovic, Yiran Shen, and Niki Trigoni. Snoopy: Sniffing your smartwatch passwords via deep sequence learning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):1–29, 2018.

[108] Anindya Maiti, Oscar Armbruster, Murtuza Jadliwala, and Jibo He. Smartwatch-based keystroke inference attacks and context-aware protection mechanisms. In *ACM Asia Conference on Computer and Communications Security*, 2016.

[109] Anindya Maiti and Murtuza Jadliwala. Light ears: Information leakage via smart lights. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(3), 2019.

[110] Anindya Maiti, Murtuza Jadliwala, Jibo He, and Igor Bilogrevic. Side-channel inference attacks on mobile keypads using smartwatches. *IEEE Transactions on Mobile Computing*, 17(9), 2018.

[111] Divine Maloney, Samaneh Zamanifard, and Guo Freeman. Anonymity vs. familiarity: Self-disclosure and privacy in social virtual reality. In *26th ACM Symposium on Virtual Reality Software and Technology*, pages 1–9, 2020.

[112] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. (sp)iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *ACM CCS*, 2011.

[113] Oge Marques. Machine learning with core ml. In *Image Processing and Computer Vision in iOS*, pages 29–40. Springer, 2020.

[114] R. Melhem, D. Mossé, and E. (Mootaz) Elnozahy. The interplay of power management and fault recovery in real-time systems. *IEEE Trans. on Computers*, 53(2):217–231, 2004.

[115] Douglas De Rizzo Meneghetti, Pedro Henrique Silva Domingues, Bruno de Freitas Vece Perez, Thiago Spilborghs Bueno Meyer, Kimberlin Kariny Gonçalves Cardoso, Amanda Maciel de Lima, Marina Yukari Gonbata, Fagner de Assis Moura Pimentel, and Plinio Thomaz Aquino Junior. Annotated image dataset of household objects from the robofei@home team, 2020.

[116] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing speech from gyroscope signals. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 1053–1067, 2014.

[117] Mark Roman Miller, Fernanda Herrera, Hanseul Jun, James A Landay, and Jeremy N Bailenson. Personal identifiability of user tracking data during observation of 360-degree vr video. *Scientific Reports*, 10(1):1–10, 2020.

[118] Robert Miller, Natasha Kholgade Banerjee, and Sean Banerjee. Combining real-world constraints on user behavior with deep neural networks for virtual reality (vr) biometrics. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 409–418. IEEE, 2022.

[119] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. Tapprints: your finger taps have fingerprints. In *ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2012.

[120] Clinton Mo, Kun Hu, Shaohui Mei, Zebin Chen, and Zhiyong Wang. Keyframe extraction from motion capture sequences with graph based deep reinforcement learning. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 5194–5202, 2021.

[121] Arsalan Mosenia, Xiaoliang Dai, Prateek Mittal, and Niraj K Jha. Pinme: Tracking a smartphone user around the world. *IEEE Transactions on Multi-Scale Computing Systems*, 4(3):420–435, 2017.

[122] Shishir Nagaraja and Ryan Shah. Voiploc: passive voip call provenance via acoustic side-channels. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 323–334, 2021.

[123] Vivek Nair, Gonzalo Munilla Garrido, and Dawn Song. Exploring the unprecedented privacy risks of the metaverse. *arXiv preprint arXiv:2207.13176*, 2022.

[124] Sashank Narain, Triet D Vo-Huu, Kenneth Block, and Guevara Noubir. Inferring user routes and locations using zero-permission mobile sensors. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 397–413. IEEE, 2016.

[125] Isaac Newton. Axioms or laws of motion. *The Mathematical Principles of Natural Philosophy*, 1:19, 1729.

[126] Teresia R Ostrach. Typing speed: How fast is average. *Orlando, FL, USA*, 1997.

[127] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: password inference using accelerometers on smartphones. In *proceedings of the twelfth workshop on mobile computing systems & applications*, pages 1–6, 2012.

[128] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: Password inference using accelerometers on smartphones. In *ACM Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2012.

[129] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[130] D. K. Pradhan. *Fault Tolerance Computing: Theory and Techniques*. Prentice Hall, 1986.

[131] William H Press and Saul A Teukolsky. Savitzky-golay smoothing filters. *Computers in Physics*, 4(6):669–672, 1990.

[132] Rahul Raguram, Andrew M White, Dibyendusekhar Goswami, Fabian Monrose, and Jan-Michael Frahm. ispy: automatic reconstruction of typed input from compromising reflections. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 527–536, 2011.

[133] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[134] Grégory Rogez and Cordelia Schmid. Mocap-guided data augmentation for 3d pose estimation in the wild. *Advances in neural information processing systems*, 29, 2016.

[135] Eyal Ronen and Adi Shamir. Extended functionality attacks on iot devices: The case of smart lights. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016.

[136] Mohd Sabra, Anindya Maiti, and Murtuza Jadliwala. Zoom on the keystrokes: Exploiting video calls for keystroke inference attacks. *NDSS*, 2021.

[137] Allen Sarkisyan, Ryan Debbiny, and Ani Nahapetian. Wristsnoop: Smartphone pins prediction using smartwatch motion sensors. In *2015 IEEE international workshop on information forensics and security (WIFS)*, pages 1–6. IEEE, 2015.

[138] Homeland Security. Increasing threat of deepfake identities - addendum examples. 2019.

[139] Yan Shoshitaishvili, Christopher Kruegel, and Giovanni Vigna. Portrait of a privacy invasion: Detecting relationships through large-scale photo analysis. In *In PETS*. Citeseer, 2015.

[140] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in Neural Information Processing Systems*, 32:7137–7147, 2019.

[141] Leonid Sigal, Alexandru O Balan, and Michael J Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International journal of computer vision*, 87(1):4–27, 2010.

[142] Laurent Simon and Ross Anderson. Pin skimmer: Inferring pins through the camera and microphone. In *Proceedings of the Third ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pages 67–78, 2013.

[143] Shirish Singh, Devu Manikantan Shila, and Gail Kaiser. Side channel attack on smartphone sensors to infer gender of the user. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, pages 436–437, 2019.

[144] Chen Song, Feng Lin, Zhongjie Ba, Kui Ren, Chi Zhou, and Wenyao Xu. My smartphone knows what you print: Exploring smartphone-based side-channel attacks against 3d printers. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 895–907, 2016.

[145] David Stavens. The opencv library: computing optical flow, 2007.

[146] Sophie Stephenson, Bijeeta Pal, Stephen Fan, Earlence Fernandes, Yuhang Zhao, and Rahul Chatterjee. Sok: Authentication in augmented and virtual reality. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1552–1552. IEEE Computer Society, 2022.

[147] Michael Stokes, Matthew Anderson, Srinivasan Chandrasekar, and Ricardo Motta. A Standard Default Color Space for the Internet - sRGB. https://www.w3.org/Graphics/Color/sRGB. [Online; accessed 22-May-2020].

[148] Jingchao Sun, Xiaocong Jin, Yimin Chen, Jinxue Zhang, Yanchao Zhang, and Rui Zhang. Visible: Video-assisted keystroke inference from tablet backside motion. In *NDSS*, 2016.

[149] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[150] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.

[151] Rahmadi Trimananda, Hieu Le, Hao Cui, Janice Tran Ho, Anastasia Shuba, and Athina Markopoulou. {OVRseen}: Auditing network traffic and privacy policies in oculus {VR}. In *31st USENIX security symposium (USENIX security 22)*, pages 3789–3806, 2022.

[152] Timothy Van Renterghem, Pieter Thomas, Frederico Dominguez, Samuel Dauwe, Abdellah Touhafi, Bart Dhoedt, and Dick Botteldooren. On the ability of consumer electronics microphones for environmental noise monitoring. *Journal of Environmental Monitoring*, 13(3):544–552, 2011.

[153] Martin Vuagnoux and Sylvain Pasini. Compromising electromagnetic emanations of wired and wireless keyboards. In *USENIX Security Symposium*, 2009.

[154] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. Friend or foe?: Your wearable devices reveal your personal pin. In *ACM Asia Conference on Computer and Communications Security*, 2016.

[155] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. Mole: Motion leaks through smartwatch sensors. In *International Conference on Mobile Computing and Networking (MobiCom)*, 2015.

[156] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4), 2004.

[157] Mika Westerlund. The emergence of deepfake technology: A review. *Technology Innovation Management Review*, 9(11), 2019.

[158] Raveen Wijewickrama, Anindya Maiti, and Murtuza Jadliwala. dewristified: handwriting inference using wrist-based motion sensors revisited. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, pages 49–59, 2019.

[159] Raveen Wijewickrama, Anindya Maiti, and Murtuza Jadliwala. Write to know: on the feasibility of wrist motion based user-authentication from handwriting. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 335–346, 2021.

[160] John Wojewidka. The deepfake threat to face biometrics. *Biometric Technology Today*, 2020(2):5–7, 2020.

[161] Qingxin Xia, Feng Hong, Yuan Feng, and Zhongwen Guo. Motionhacker: Motion sensor based eavesdropping on handwriting via smartwatch. In *IEEE Conference on Computer Communications Workshops (INFOCOM Workshops)*, 2018.

[162] Chao Xu, Parth H Pathak, and Prasant Mohapatra. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. In *ACM Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2015.

[163] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems*, 32, 2019.

[164] Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. Deep image matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2970–2979, 2017.

[165] Weitao Xu, Girish Revadigar, Chengwen Luo, Neil Bergmann, and Wen Hu. Walkie-talkie: Motion-assisted automatic key generation for secure on-body device communication. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 1–12. IEEE, 2016.

[166] Yi Xu, Jan-Michael Frahm, and Fabian Monrose. Watching the watchers: Automatically inferring tv content from outdoor light effusions. In *ACM CCS*, 2014.

[167] Zhi Xu, Kun Bai, and Sencun Zhu. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2012.

[168] Jian Ye, Zhe Chen, Juhua Liu, and Bo Du. Textfusenet: Scene text detection with richer fused features. In *IJCAI*, pages 516–522, 2020.

[169] Qinggang Yue, Zhen Ling, Benyuan Liu, Xinwen Fu, and Wei Zhao. Blind recognition of touched keys: Attack and countermeasures. *arXiv preprint arXiv:1403.4829*, 2014.

[170] Jerrold H Zar. Significance testing of the spearman rank correlation coefficient. *Journal of the American Statistical Association*, 67(339):578–580, 1972.

[171] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z Li. Faceboxes: A cpu real-time face detector with high accuracy. In *IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–9, 2017.

[172] Tong Zhu, Qiang Ma, Shanfeng Zhang, and Yunhao Liu. Context-free attacks using keyboard acoustic emanations. In *ACM CCS*, 2014.

[173] Li Zhuang, Feng Zhou, and J. D. Tygar. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security (TISSEC)*, 13(1), 2009.

[174] Zoran Zivkovic et al. Improved adaptive gaussian mixture model for background subtraction. In *ICPR (2)*, pages 28–31, 2004.

VITA

Mohd Sabra, at August 2018 entered the University of Texas at San Antonio and received a PhD. in Computer Science in May 2023.