Hidden Reality: Caution, Your Hand Gesture Inputs in the Immersive Virtual World are Visible to All!

Sindhu Reddy Kalathur Gopal and Diksha Shukla, *University of Wyoming;*
James David Wheelock, *University of Colorado Boulder;*
Nitesh Saxena, *Texas A&M University, College Station*

https://www.usenix.org/conference/usenixsecurity23/presentation/gopal

This paper is included in the Proceedings of the
32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Hidden Reality: Caution, Your Hand Gesture Inputs in the Immersive Virtual World are Visible to All!

Sindhu Reddy Kalathur Gopal[1] Diksha Shukla[1] James David Wheelock[2]
Nitesh Saxena[3]
[1]*Department of Electrical Engineering and Computer Science, University of Wyoming, Laramie, WY 82071*
[2]*Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309*
[3]*Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843*
{skalathu@uwyo.edu, dshukla@uwyo.edu, jawh0915@colorado.edu, nsaxena@tamu.edu}

Abstract

Text entry is an inevitable task while using Virtual Reality (VR) devices in a wide range of applications such as remote learning, gaming, and virtual meeting. VR users enter passwords/pins to log in to their user accounts in various applications and type regular text to compose emails or browse the internet. The typing activity on VR devices is believed to be resistant to direct observation attacks as the virtual screen in an immersive environment is not directly visible to others present in physical proximity. This paper presents a video-based side-channel attack, *Hidden Reality (HR)*, that shows – *although the virtual screen in VR devices is not in direct sight of adversaries, the indirect observations might get exploited to steal the user's private information.*

The *Hidden Reality (HR)* attack utilizes video clips of the user's hand gestures while they type on the virtual screen to decipher the typed text in various key entry scenarios on VR devices including typed pins and passwords. Experimental analysis performed on a large corpus of 368 video clips show that the *Hidden Reality* model can successfully decipher an average of over 75% of the text inputs. The high success rate of our attack model led us to conduct a user study to understand the user's behavior and perception of security in virtual reality. The analysis showed that over 95% of users were not aware of any security threats on VR devices and believed the immersive environments to be secure from digital attacks. Our attack model challenges users' false sense of security in immersive environments and emphasizes the need for more stringent security solutions in VR space.

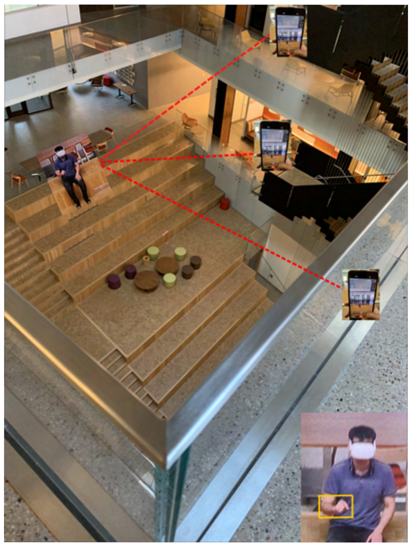


Figure 1: An adversary's sweet-spots to videotape target users' hand gestures while they type on the virtual keypad in an immersive environment is presented here. The boundary box shows the visible hand gestures of the user captured in the video clip.

1 Introduction

Research on attacks has demonstrated that adversaries can steal target user's private and sensitive information by unobtrusively videotaping user's keyboards, monitors or touch screens using hand-held camera-enabled devices [1–3]. However, security and privacy threats in VR devices where a user operates and types on their device's virtual screen in presence of other people in a real environment [4] have been rarely investigated and often discarded. While performing typing operation on VR devices, VR user is required to make hand movements using either controllers [5] or hand gestures [6] to register their clicks or for keyboard navigation (e.g., the bounding box in Figure 1 shows user's interaction with his VR device using hand gestures). Taking the case of typing scenario in the immersive environment, we investigate security vulnerabilities in VR devices by presenting a video-based side-channel attack, *Hidden Reality*.

The *Hidden Reality* attack model relies only on the spatio-temporal dynamics of the user's hand gestures from stealthily captured video clips. The HR attack can easily be launched using only the recorded video footage and does not require any user-specific training. The attack model requires (i) the user's hand gestures to be visible in the recorded video for the targeted typing activity and (ii) knowledge of the used virtual keypad layout[1] Upon launching, the HR model locates and tracks the user's hands, and filters the frames where the clicks are detected. A click operation during typing indicates that the desired key is entered or pressed on a virtual keyboard. The HR model then maps the tracked hand pixel locations of clicks to the known geometry of the virtual keyboard. The attack deciphers the typed content or graphical-lock input patterns by utilizing the mapped pixel locations on the keypad layout. To the best of our knowledge, the *Hidden Reality* is the first attack model on virtual reality inputs that relies only on the stealthily captured hand gestures to decipher the typed content and uncovers serious security threats in various typing scenarios: (1) pin entry, (2) alphanumeric password entry, (3) graphical pattern lock input, and (4) regular text typing.

The efficacy of the Hidden Reality model evaluated on a large corpus of 368 video clips captured in various text entry scenarios show that the attack can successfully: decode $\approx$ 95% of the digits in typed pins, infer an average of $\approx$ 90% of the characters in the regular typed text, and decipher $\approx$ 75% of the characters in typed passwords in the first five guesses. The attack analysis of the graphical lock patterns in our dataset shows that the HR model was able to reconstruct 100% of the pattern locks within the first three attempts.

The high inference rate by the Hidden Reality attack model: (i) exposes VR users' false perception of security and privacy in immersive environments [9], and (ii) raises alarm to the VR vendors and the security community to re-evaluate the VR input design for security threats posed by the attacks such as ours.

To put the practicality of the *Hidden Reality* attack model into the perspective, we briefly discuss the viable assumptions and requirements to successfully launch the attack. The Hidden Reality attack needs: (1) the captured video footage of the targeted victim's hand gestures while they type on their VR device, and (2) knowledge of the victim's keyboard layout.

Videotaping the victim's hand gestures– The attacker can stealthily record videos of a targeted victim's hand gestures while s/he interacts with their VR device. The head-mounted displays (HMD) such as VR headsets partly or sometimes completely block the users' view of their physical surroundings [10]. The users are generally unaware of events in their physical surroundings while using VR headsets. To re-affirm this, we conducted a user study on user behavior and perception of security for virtual reality devices. The survey results show that $\approx$ 80% of users are generally unaware of their surroundings in the real world when they wore the HMD and were immersed in virtual reality (see Section 6).

[1]The keypad layout design can easily be guessed by the adversaries using the publicly available information from the make and model of the VR device's vendor or by owning the same brand device themselves as the target victim. Studies show that the majority of the population uses default keypad design [7], and those who use custom-designed/random keypad layouts face usability challenges [8].

Inference of the keyboard layout. The adversary can easily determine the manufacturer of the VR device by looking at the recorded videos or while videotaping the users. For example, the HTC VR headset can be identified with the presence of a headrest [11]. Similarly, Meta Quest is sleek in design and has a head strap [12]. After knowing the VR device's make and model, details of default keyboard geometry can be obtained either by owning it or finding google search images of the keypad layout for the targeted device. The current version of the HR attack model can easily be launched to decipher the typed contents on the default keyboard of a VR device. This shows a major security threat when the majority of the users (over 80%) prefer using default and in-built keyboards [7].

Motivation Behind the Attack Model – VR devices are no longer confined only to applications in the gaming industry and are extended widely to other application domains [13–20]. Existing studies show that usage of VR devices is not only restricted to home or private spaces, but also in semi-public and public places such as schools, universities, office spaces, labs, and libraries where other people are present in the physical surroundings [21], [22]. Considering a threat scenario where the VR user interacts with their VR device in places where other people are nearby, we introduce the *Hidden Reality* attack model that successfully reconstructs over 75% of the typed content while relying only on visible hand gestures captured in video recordings of the user. An example scenario is illustrated in Figure 1 where a user is located on the first floor of a university building and the attacker records the user's hand gestures stealthily from a distance to not raise any suspicion. In the case of long-distance videotaping, as shown in Figure 1, an optical zoom camera configuration can be utilized to capture a clear view of hand movements.

***Contributions* –** The following are the contributions of the paper.

- We introduce a video-based side-channel attack, *Hidden Reality*, on the users' typing in immersive environments. The attack model does not require user-specific training and can be launched easily without raising any suspicion. The model relies only on the spatio-temporal dynamics of the user's hand gestures captured in stealthily obtained video clips.

- We collected a large corpus of 368 videos using a consumer-grade smartphone camera, and rigorously evaluated the attack performance. The experiment results show that the HR model can successfully decipher an average of $\approx$ 90% of characters in the typed text, decode $\approx$ 95% of digits in pins, and decipher $\approx$ 75% of characters in the typed password in the first five guesses. Our attack could successfully reconstruct all the graphical lock patterns chosen by volunteer participants in our dataset in less than three attempts.

- We conducted a user study to understand the users' perception of the security of VR devices. The summary results of our study show that – (i) the majority of users believe that VR devices are secure for sensitive transactions, (ii) over 95% of users were unaware that VR devices can be subjected to digital attacks, and (iii) ≈ 45% of users currently use passwords, pins, or graphical lock patterns to secure their sensitive information stored on these devices. Through this work, we attempt to raise awareness among users about security threats presented by attacks such as ours in immersive environments.

Paper Organization - We present the threat scenarios in Section 2. The implementation details of the *Hidden Reality* attack model are included in Section 3. Section 4 discusses the dataset and the experimental analysis. Performance evaluation of the attack model is presented in Section 5. Section 6 presents a summary of the results of our user study. Section 7 discusses the limitations and countermeasures of the attack model. We discuss the related work in Section 8 and finally, we draw our conclusions in Section 9.

2 Threat Model

The Attacker – An attacker is a person who has some malicious intent to decipher the context of the user-typed text or steal the targeted user's passwords, pins, or graphical patterns. In the context of *Hidden Reality* attack model, the attacker identifies the target victim, stealthily videotapes the victim's hand movement without raising suspicion while the victim types on their VR devices, and launches the HR attack on the target user to steal their private information. This can easily be done when the target user uses their device and is directly or indirectly present in the attacker's sight. Additionally, VR users are generally partly or completely unaware of their physical surroundings when they wear HMD.
The Attacker's Perspective – The attacker who is interested in stealing the target user's information will monitor the user's activities and will start videotaping as soon as the user turns on their device and wears the HMD. In some cases, the attacker may get access to only part of the video recordings in the case of stealthily obtaining existing videos (e.g., through surveillance cameras). In this scenario, even if the attacker could infer some characters of the password or pin, or context of the text typed, they can still use other methods [23] with highly reduced search space and make the victim vulnerable. This can pose a serious security breach to users' accounts and their data.

2.1 Threat Scenarios

Assuming that the attacker developed or has access to the HR model, video clips of the target user, and a known geometry and keyboard layout of the virtual keyboard, below mentioned realistic threat scenarios are designed.

❶ **Threat 1 – Graphical Pattern Lock Input.** The user draws the graphical lock pattern as a first step upon turning on and wearing a VR device such as Meta Quest 2. An adversary being aware of this setting can start video recording the hand gestures of the user knowing the fact that the user will enter a graphical lock pattern. Applying the steps involved in the HR attack model (see Section 3), a person with malicious intent can reconstruct the graphical lock patterns.

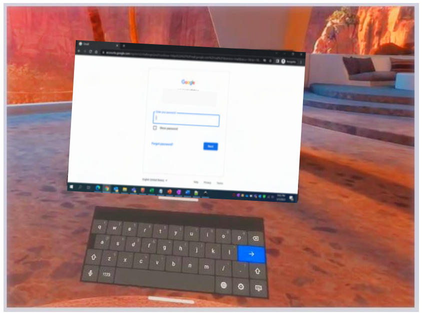


Figure 2: The user's view of the virtual screen and virtual keyboard while typing the password for their Gmail account on Meta Quest 2.

❷ **Threat 2 – Password Entry Process.** To login into a web-based Gmail account on a VR device, the user enters the word 'Gmail' on the browser and then types their user credentials. Figure 2 depicts this scenario where the target user is ready to type in their password. When the word 'Gmail' is encountered, the HR attack model identifies the scenario where a user is most likely to type their Gmail id and password. The attack can be extended easily to other applications where password entry is required for device and account access.

❸ **Threat 3 – email Entry Process.** The HR model checks for the criteria where the typed characters include '@', '.com', or 'Gmail' to conclude that the user is composing and sending an email to friends, family, or colleagues. Again, the HR attack can be extended to other email applications such as 'outlook', and 'yahoo'. By employing the steps of the HR attack model, an adversary can infer the content of the email.

❹ **Threat 4 – Pin Entry Process.** Users enter their pin on the VR device's number keypad to verify their identity for applications that require pins. For example, in Meta Quest 2, a user could send or request money from a Facebook user using the Facebook Messenger application. The criteria to confirm this financial transaction is to enter a four-digit pin on the virtual numeric keypad. By determining the keyboard geometry (relatively small and square in shape) during the character inference step of the *Hidden Reality* model, the HR attack model can decipher the user's pin.

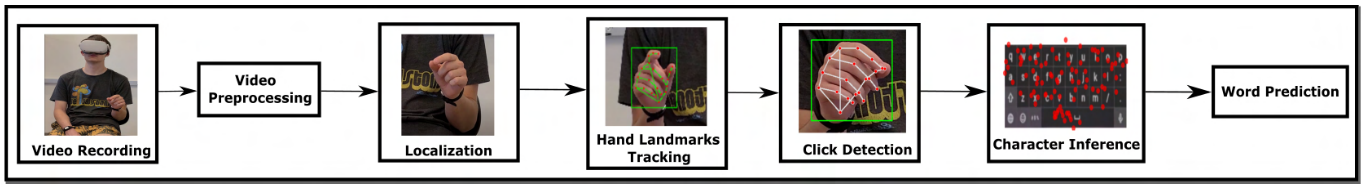


Figure 3: Figure shows the steps involved in the execution of the *Hidden Reality* attack model. An adversary captures the video footage of a targeted user while the user is performing typing operation on their virtual screen. Region of interest is obtained by localizing the typing hand and tracking the hand landmarks points on the preprocessed video. By mapping the tracked hand pixel location coordinates obtained in the click detection step onto the known keyboard geometry, the typed text gets inferred.

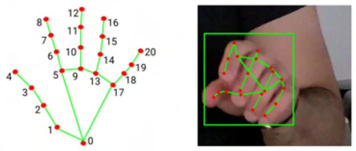


Figure 4: Tracked hand landmark points of a targeted user's hand. *MediaPipe_Hands* library [24] was used to track all 21 landmark points throughout the video clip. The landmarks of interest for implementing the attack model and identifying the clicks are marked as #3, #4, #7, and #8 which represent the fingertips of the index finger and thumb of the typing hand. The image on the right shows the localization of the typing hand as the rectangular bounding box. Red dots show the hand landmark location coordinates in that frame.

5 Threat 5 – Surfing or Text Entry on Browser. In addition to the scenarios mentioned above, the user would also surf or type random text on a VR device. When none of the conditions mentioned in the above scenarios are satisfied, the HR attack model concludes that the target user is browsing, or typing random text, and then applies the steps of the model to decode the text typed by the user. An example scenario is when a user types the 'amazon' keyword, then the user is most likely going to search for a product or perform a financial transaction for purchasing the searched product using their Amazon account.

3 Attack Approach

In this section, we provide implementation details of the *Hidden Reality* attack model that utilizes the spatio-temporal information of the user's hand gestures while s/he types on their device's screen in an immersive VR environment. The attack model takes a video segment of the typing activity corresponding to a threat scenario as an input (e.g., pin unlocking) and then employs the key steps involved in the implementation of the *Hidden Reality* model as depicted in Figure 3.

3.1 Implementation of HR Model

Details of the steps involved in the implementation of the *Hidden Reality* attack for various attack scenarios follow.

Algorithm 1: Hand Localization and Tracking

```
Data: $frames$
Result: $hand\_locations$
1  $frame\_num \leftarrow 100$, $pad \leftarrow 100$
   /* Find hands in every 100 frames                */
2  while frames remain do
3      $landmarks.append(get\_hands(frame, 2))$
4      $frame\_num \leftarrow frame\_num + 100$
5      $frame \leftarrow get\_frame(frame\_num)$;
6  end
   /* Find the hand that moves the most             */
7  $labels, centers \leftarrow cluster(landmarks, n = 2)$
8  foreach $label$ in $(0,1)$ do
9      $indices \leftarrow where(labels = label)$
10     $hand \leftarrow landmarks[indices]$
11     $hand\_vars.append(var(hand.x))$
12     $hand\_bounds.append(max(hand) +$
         $pad, min(hand) - pad)$
13 end
14 $bounds \leftarrow hand\_bounds[argmax(hand\_vars)]$
15 $hand\_locs \leftarrow [\ ]$
   /* Crop video to typing hand, find
      landmarks for this hand                      */
16 foreach $frame$ in $frames$ do
17     $cropped \leftarrow frame[bounds]$
18     $landmarks \leftarrow get\_hands(cropped, 1)$
19     $hand\_locs.append([frame\_num, landmarks])$
20 end
21 return $hand\_locs$
```

3.1.1 Video Preprocessing

The beginning and ending segments of the captured video footage that contain non-typing operations such as setting up the user's working boundary, and just visualizing the VR environment are removed from the video recording. This was done by watching the captured videos at a slow playback speed on the VLC player. The resulting video clip is used for further processing.

3.1.2 Localization and Hand Landmark Tracking

Firstly, the preprocessed video is converted into video frames. Most of the video frames contained the presence of both the hands of the targeted user: a non-stationary hand performing the typing operation using hand gestures on the virtual keyboard, and a resting hand. For the subsequent steps in our attack model to be efficient and accurate, videos are trimmed to only contain the region of interest (ROI). We define ROI as an area of video frames where only the typing hand is present throughout the duration of the input.

We used *MediaPipe_Hands* [24] library for hand tracking in the video segment. *MediaPipe_Hands* is a machine learning-based hand tracking solution to track multiple landmark points of the user's hand in a video segment as shown in Figure 4. The hand landmarks coordinates were considered for every 100^{th} frame in the video. This assumption is reasonable because consideration of every frame to localize the typing hand would take significantly larger computation time and this interval provided a balance between the approximation of ROI and the efficiency of this step. This process of extracting hand locations on every 100^{th} frame is described in steps 1-6 of Algorithm 1.

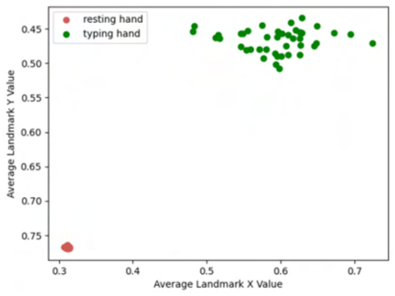


Figure 5: The Figure shows the average hand landmark positions of typing hand and the resting hand. Red dots denote a resting hand as there is no or very minimum variance between the data points and green dots represent the hand landmark pixel location data of the typing hand.

After extracting the hand landmarks coordinates, the next step is the creation of positional data lists to determine the typing hand and the bounds of the typing region. Positional data consists of average hand landmark (x, y) coordinates lists for each hand in the video. x and y coordinates represent the horizontal and vertical positions respectively. The x-coordinates list that showed the most variation in its $x-axis$ is considered to be the typing hand, as the other hand typically remained stationary throughout the video. The positional data plotted for typing and resting hand is shown in Figure 5.

If only the typing hand is present in the input video, this step produces a list of positional coordinates for the typing hand and, an empty list for the other hand. Using the positional coordinates of the typing hand, we obtained the bounds of the typing region. The right image in Figure 4 shows an

Algorithm 2: Click Detection

```
   Data: $hand\_locations$
   Result: $frameNum, clickCoordinates$
1  $thumb \leftarrow [3,4], index \leftarrow [7,8]$
   /* get average hand positions, distances
      between the thumb and index finger     */
2  foreach entry in $hand\_data$ do
3      $handPos.append(mean(entry))$
4      $distance.append(||entry[thumbPosition] - entry[indexPosition]||_2)$
5  end
   /* find the speed of hand                 */
6  $speed \leftarrow ||handPos[1:] - handPos[:-1]||_2$
   /* normalize the metrics and combine      */
7  $distance \leftarrow normalize(smooth(distance))$
8  $speed \leftarrow normalize(smooth(speed))$
9  $metric \leftarrow 0.5 \cdot distance + 0.5 \cdot speed$
   /* find peaks in the data                 */
10 $frameNum \leftarrow find\_peaks(metric)$
11 return $frameNum, handPos[frameNum]$
```

example of localization, where a rectangular bounding box represents the ROI. The bounding box shows the bounds of the typing hand in that specific frame. Red dots represent the hand landmark pixel coordinates in that frame. Steps 8-13 of Algorithm 1 computes the variation in positional coordinates and determines the bounds of detected hands. Steps 16-21 in Algorithm 1 show the extraction of hand landmark position data in the cropped ROI for each frame, returning a list of the detected hand pixel data with the corresponding frame numbers.

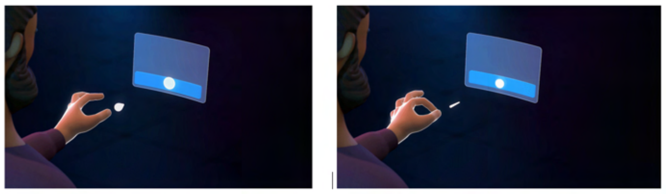


Figure 6: Hand gesture to point at the desired character on Meta Quest 2 [6] (left). A user performs a click operation to enter the selected character (right).

3.1.3 Click Detection

In the click detection step, clicks performed by the users are identified using the hand pixel data acquired from the previous step. A click on the virtual keyboard implies that the user has entered the desired character by performing a pinch operation on that key. In a pinch operation, the thumb and index fingertips will be in close contact as shown in Figure 6. The landmark points #3 and #4 represent the thumb fingertips and the landmark points #7 and #8 are for index fingertips as shown in Figure 4. To identify a click, two metrics are computed based on the aforementioned landmark data for thumb and index fingertips. The first metric is distance - the euclidean distance between the landmark points: *thumbPosition* and the *indexPosition* of the typing hand for each frame is computed. That is, if the distance between the *thumbPosition* and *indexPosition* is minimum in a frame compared to its adjacent frames, then that frame is detected as a clicked frame. The distance formula is given in Equation (1) where *thumbPosition* and *indexPosition* are the average coordinates of thumb and index fingertips respectively.

$$distance = ||thumbPosition - indexPosition||_2 \quad (1)$$

Speed is the second metric to detect the click. The speed of the hand movement reduces when the user reaches the desired character and the user's hand comes to rest during the click operation. Speed is computed as the euclidean distance between average hand landmarks data on the adjacent frames in the video. For example, consider the current frame, previous frames, and next frames as i, $i-10$, and $i+10$ respectively. If the user had registered a click and entered a character in frame i, then the speed value will be minimum for i^{th} frame when compared to the neighboring frames.

The speed formula is given below where $handPos[i]$ denotes the average value of the hand landmark points of the i^{th} frame and $handPos[i-1]$ represents the average hand landmarks data of the previous frame, and so on:

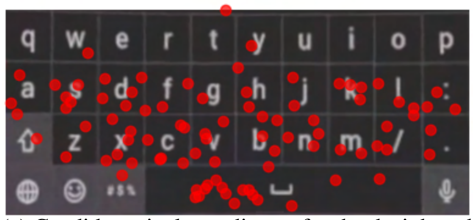
$$speed = ||handPos[i] - handPos[i-1]||_2 \quad (2)$$

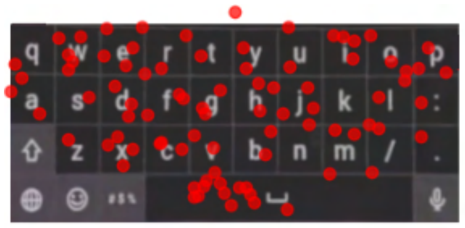
Steps 1-5 of Algorithm 2 computes the distance metric and stores it in a list. Step 6 in Algorithm 2 computes speed values. Steps 7-9 describe the process of normalizing distance and speed metrics and their combined values are stored in another list. Utilizing this combined list along with the frame numbers, the *find_peaks* function from SciPy.signal is applied that returns the local minima as shown in step 10. The local minima returned from the previous step represent that the click operations are performed by the users. Frame numbers along with the hand position coordinates are returned for the frames in which the click is detected.

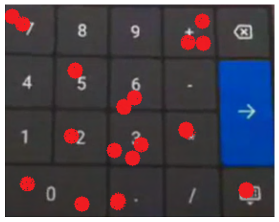
3.1.4 Character Inference

The frame numbers where a click is detected and their corresponding hand location coordinates returned from the click detection step are used to infer the characters typed by the users. To do this, the hand location coordinates are mapped to the keyboard geometry of the Meta Quest 2, returning the five closest character predictions, sorted by their distance. The steps involved in inferring the characters typed by the target user are as follows:

- The keyboard geometry is determined based on the bounds of the hand location data. The bounds represent the minimum and maximum values of the hand location coordinates obtained from the click detection step. With edges evenly spaced across the bounds, the keyboard is divided into ten columns and four rows to adhere to the keys layout of the virtual keyboard as shown in Figure 7a. For mapping a pin, the numeric keypad is divided into four rows and five columns as shown in Figure 7c.


(a) Candidate pixel coordinates for the deciphered characters.


(b) Final mapped coordinates for the deciphered characters.


(c) Pin lock virtual keypad.

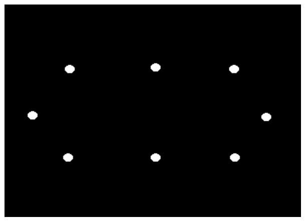

(d) Pattern lock virtual keypad.

Figure 7: Figure 7a shows the mapping of hand pixel location coordinates with that of the keyboard geometry. The figure shows the existence of outliers. Figure 7b depicts the mapping of the location coordinates after removing the outlier points. Figure 7c shows the mapping of clicked hand location coordinates onto the number keypad of Meta Quest 2. The graphical lock pattern grid on Meta Quest 2 is shown in Figure 7d.

- Figure 7a shows the mapping of hand location coordinates to the keyboard geometry. Since some of the hand location data points were outliers, the HR model excluded them from the data. This is done using the local outlier factor algorithm from *sklearn.neighbors*. Figure 7b shows the mapping of the location coordinates after removing the outlier points.
- For each video segment, the key center coordinates (x_n,y_n) of each key in the keyboard are computed. Character inference is performed by computing the euclidean distance between each hand location data point and the key centers. The resulting distance values are sorted in order of increasing distances, from d=1 to d=35.

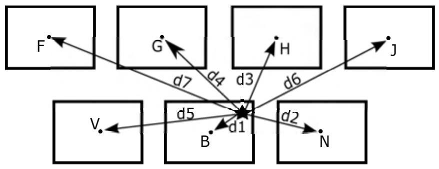


Figure 8: 'Star' marker depicts the hand pixel location coordinates of a click in an example frame and dots represent the key center coordinates (x_n,y_n). The top seven key predictions which are closer to the clicked hand location coordinates sorted by distances (d1 to d7) are shown here.

d=35 represents the number of keys on the keyboard. The top five distance values (d=5) were considered for our analysis as there was no significant improvement in accuracy after including values of d greater than 5. Hence we concluded that distance values (d=5) result in the optimal character predictions or guesses in our study. The indices of these distance values were used to return the corresponding characters. Figure 8 shows the prediction of the top 7 characters based on the hand location data of an identified click.

- Finally, the character predictions based on the top five guesses are added to a data frame.

3.1.5 Word Prediction

Some of the attack scenarios require a word prediction step; e.g., decoding email content. To predict the words, the characters inferred from the previous step were combined into different groups, delimited by the space character. In addition to the space delimiter, the length of the word is considered an attribute to group the characters. After the groups are identified, a similarity metric is computed between the combined groups and words in the dictionary. The dictionary used for word prediction is Brown Corpus [25]. The similarity metric computation is performed by using a modified edit distance algorithm [26] to find the similarity of the group to the given word. The word from the dictionary with the highest similarity metric value is the predicted word.

4 Experiments

4.1 Data Collection

With the approval of our university's Institutional Review Board (IRB), we invited 42 volunteer participants for our data collection study. The participants were first briefed about the purpose and the procedure[2] of our study, and were asked to provide their written consent to participate in the study and for videotaping their hand gestures while they typed on the virtual keyboard of the Meta Quest 2. The volunteer participants (Male: 23, Female: 18, Other: 1) recruited for our data collection activity include students and postdocs between the ages of 19 and 35 years. As per the approved IRB protocol, every participant was briefed and was asked to report immediately if they experience uneasiness or any kind of discomfort during the experiment[3]. However, during our data collection activity, none of the participants experienced or reported any such side effects.

Data Collection Sessions. We collected data in two different sessions[4]: Session I and Session II. In each session, the volunteer participant, first, wore the Meta Quest 2 device, then set the working boundary and stays within it for performing the typing operation. However, the users were allowed to move within the working boundary. The registered volunteers were invited to participate in Session II, 5-7 days after Session I. Because the participant pool also consisted of inexperienced VR users, before these two sessions, there was a practice session where these subjects were given time to learn to type on the Meta Quest 2 and themselves familiar with the device. Also, during this time, the participants chose their passwords, pins, graphical-lock patterns, and email content. We did not record any data during the practice sessions. A total of 368 videos were recorded from 42 volunteer participants from both sessions while the users performed typing operations during different attack scenarios. The duration of videos ranged from 10 to 60 sec depending upon the attack scenario.

4.2 Attack Scenarios

The HR attack model presented in this paper can differentiate the attack scenarios depending on the action the user performs and the text the user types on the virtual keyboard. Based on the identified threat scenario, a suitable attack model is deployed. Detailed information is provided below.

Scenario 1: Graphical Pattern Lock Input
SETUP. After switching on a VR device and then wearing it, the first step the user does is to unlock the device by using a graphical lock pattern if there exists a lock pattern. To launch an attack during this case, the following setup was performed

[2]The participants were told that their hand gestures while typing on the virtual screen will be videotaped and the same will be used to analyze the security and privacy vulnerabilities in the VR devices. The participants were not specifically informed about the usage of their video clips for inferring the typed text/passwords.

[3]Our data collection experiment protocol had procedures in place to handle situations of users feeling nausea, dizziness, loss of balance, etc. In addition, participants with a history of conditions such as vertigo, epilepsy, seizures, and balance disorder were not allowed to participate in our experiments.

[4]Existing studies [27] show the intra-user variance in the users' hand movements behavior while they type on a virtual screen using their hand gestures. Hence, we collected hand movements data in two different sessions to capture regular variations in the participants' gestures and to have a stringent evaluation of our attack model. The volunteer participants entered the same chosen password, pin, and graphical lock pattern for both sessions (I and II) with the same experimental setup.

by the volunteer participants on Meta Quest 2 during the practice session. The participants picked a pattern of their choice connecting dots on the screen in a particular shape and order from the Settings $\rightarrow$ Security option available in the VR device. To set the graphical lock pattern, the user performs the 'point and pinch' operation on the starting dot of the lock pattern on the VR screen, draws the pattern with the fingers pinched, and then releases their fingers once the drawing of the pattern is complete. The default graphical pattern input grid on Meta Quest 2 has eight dots as shown in Figure 7d. Quest 2 has a requirement of connecting at least four dots to be considered a valid graphical lock pattern.

Once the participants were successful in unlocking the device multiple times, they were invited to Sessions I and II.

In total, 50 video recordings were captured during both sessions for this attack scenario.

IDENTIFICATION OF ATTACK SCENARIO. The first and foremost operation performed by the user after turning on and wearing a VR device is to unlock it before starting to use it. Even when the user removes the VR device and wears it again, still the user needs to unlock the device. We use this as a criterion to identify that the user is entering a graphical-lock pattern.

THE ATTACK MODEL. Implementation of the HR attack during this scenario includes the following steps: (i) video preprocessing to discard beginning and ending frames where the user does not perform any click/'point and pinch' operation, (ii) localization and hand tracking to identify the ROI to obtain the hand landmark points and store the location coordinates data, (iv) click detection to detect the clicks. The final step in the model is to plot the stored location coordinates to reconstruct the entered graphical pattern.

Scenario 2: Password Entry

SETUP. The HR attack is launched while the user logs in to their user accounts by typing the password. The volunteer participants were asked to choose random passwords[5]. The users generated multiple alphanumeric passwords of 10-17 characters long with atleast one uppercase, lowercase, and special character using the Memorable Password Generator tool [28] until they found a password of their choice for our experiments.

For the password entry scenario, we created test google accounts. Each user logged into the Gmail account using the password practiced and set during the practice session of the data collection activity. A new video was recorded each time the user logged into the Gmail account. This procedure was repeated five times. A total of 50 video recordings were collected from five volunteers during sessions I and II.

[5]To minimize the risk of volunteer participants' sensitive information leaking during our experiment, we asked them to type only randomly generated passwords/pins for our data collection experiments. We specifically instructed participants to not input their present, past, or possible future passwords, pins, or lock patterns.

IDENTIFICATION OF ATTACK SCENARIO. If the text typed by the user contains the word 'Gmail' then the user is very likely to enter their password as the next step to login into their Gmail account. This is a valid assumption for any web-based application such as Gmail, Amazon.

The user needs to open the browser and type the word 'Gmail' to log in to their Google/Gmail accounts. This scenario can be extended to any other user accounts such as 'yahoo', 'outlook', and 'amazon' too by including them in the search keywords.

THE ATTACK MODEL. The steps involved in launching an attack during this scenario to decipher the passwords are from steps 3.1.1 to 3.1.4 of the attack model.

Scenario 3: email Content Entry

SETUP. In this attack scenario, we briefed the situation to users where they will be sending an email to their friend, family, or colleague regarding a meet-up or updating them with some personal or sensitive information. During the practice session of data collection activity, the users thought of a message that is to be sent to a friend, family member, or colleague. Once the users were ready for the actual data recording session, we recorded the videos of them while they were composing and sending the email using the Gmail application on the web browser.

IDENTIFICATION OF ATTACK SCENARIO. Because there is no inbuilt Gmail application on VR devices similar to that of smartphones, and tablets to directly open it and compose an email, the user will need to open the browser, type 'Gmail' to login to their Gmail account, and then compose an email. This scenario can be considered a successor of the previous attack scenario. If the search keyword or search character typed by the user includes '@', '.com', and 'Gmail', then we conclude that the target users are composing an email to be sent. This scenario can be extended to other applications such as 'yahoo', and 'outlook' too by including them in the search keywords.

THE ATTACK MODEL. This attack model comprises of implementation steps from 3.1.1 to 3.1.5.

Scenario 4: Pin Entry

SETUP. For this attack scenario, we created two sample Facebook accounts where one account will serve as a sender account and the other as a receiver account. We then added bank accounts to both of them. During the practice session, volunteer participants get themselves familiarised with Facebook Pay and set the pin of their choice for verifying the financial transaction. Facebook Pay is a functionality in Facebook Messenger where a sender can choose a receiver and send/request money. Facebook Pay has criteria that the length of the pin required to confirm the financial transaction will need to be exactly four characters. During the data recording session, participants confirmed the financial transaction of sending \$1 from the sender account to the receiver account by entering

Table 1: Phrases typed by the volunteer participants during the data collection activity. Every English alphabets appear at least once in each phrase.

#	Phrase	Session
1	The quick brown fox jumps over the lazy dog	Session I/II
2	The five boxing wizards jump quickly	Session I
3	Waxy and quivering jocks fumble the pizza	Session II

the four-digit pin set by them during the practice session. In total, 50 videos were recorded from Sessions I and II.

IDENTIFICATION OF ATTACK SCENARIO. The identification of the pin entry process is done based on the determination of the keyboard geometry in the character inference step. If the keyboard geometry was relatively small and possesses a square shape, as shown in Figure 7c, then this attack model is launched.

THE ATTACK MODEL. To launch this attack model, we implemented steps 3.1.1 to 3.1.4. In step 3.1.4 which is a character inference step, we use a pin keypad in place of a QWERTY keyboard.

Scenario 5: Surfing or Text Entry on Browser

SETUP. During this scenario, users typed the phrases mentioned in Table 1 on the VR device's browser. This scenario can be compared to the task where the user surfs through the browser by typing text for searching a product, learn about new technology, etc. The volunteer participants opened the browser on Quest 2 and typed phrases 1 and 2 from Table 1 during Session I and phrases 1 and 3 during Session II. All 26 alphabets occurred at least once in each phrase. A separate video was recorded for each phrase. We collected 84 videos from 42 participants during session I and 84 videos during session II with a total of 168 videos.

IDENTIFICATION OF ATTACK SCENARIO. If there are no criteria matching any of the above-mentioned attack scenarios, then the model concludes that the target user is performing activities such as browsing, or typing random text.

THE ATTACK MODEL. All the steps of the attack process starting from 3.1.1 which is video preprocessing to 3.1.5 which is word prediction was involved here.

5 Evaluation of HR Model

In this section, we present the overall success rate for inferring the text typed in the video recordings captured from 42 volunteer participants. Evaluation of the HR model was performed during all the attack scenarios where the user types passwords, pins, graphical-lock patterns, email content, and random text.

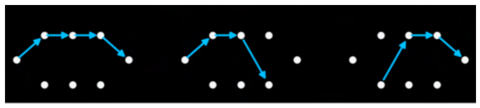
An accuracy metric was used to evaluate the performance of the HR attack model. Accuracy is the ratio of the number of characters that are correctly predicted to the number of actual characters typed, as shown in the following equation.

$$CharacterAccuracy = \frac{\text{Characters correctly inferred}}{\text{Characters typed}} \tag{3}$$

Character accuracy is in turn dependent on the click detection step which identifies the clicks. The click detection algorithm detects whether a click had occurred or not and returns the frame number along with their respective hand pixel coordinates. The performance of the click detection step is denoted using the equation given below.

$$ClicksDetect\% = \frac{\text{UC} + \text{NC}}{\text{UC} + \text{NC} + \text{EC} + \text{MC}} \tag{4}$$

We denote UC as user_clicks that represent the actual click performed by the user, NC as not_a_click where the target user performs other operations such as navigating through the keyboard looking for the desired key. EC and MC denote extra_user_clicks and missed_user_clicks respectively. Extra_user_clicks indicates that the model has falsely detected a user click and missed_user_clicks represents the number of actual user clicks missed by the HR model.


(a) Example graphical-lock pattern 1

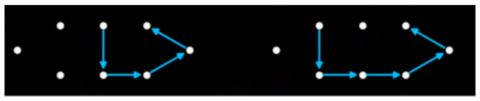

(b) Example graphical-lock pattern 2

Figure 9: Possible ways of overlaying the tracked hand location coordinates predicted by the HR attack model onto the graphical pattern input grid of Meta Quest 2 is shown here. The arrows in Figures 9a and 9b represent the direction of the lines drawn to complete the chosen graphical pattern.

Graphical Pattern Lock Input. The HR attack model's performance for deconstructing graphical patterns is measured using the number of attempts required to unlock a VR device [29, 30]. The number of attempts to deconstruct a pattern is directly proportional to the possible ways of overlaying the geometrical structure predicted by the model onto the graphical-lock pattern grid screen. On evaluating the performance of the HR attack model, the results show that all the graphical-lock patterns chosen by the users in our data collection activity can be predicted and can unlock a VR device in a maximum of three attempts.

Figure 9 shows that the graphical pattern in 9a can be overlaid in three different ways on the input screen and hence this pattern can be guessed in three attempts while the pattern

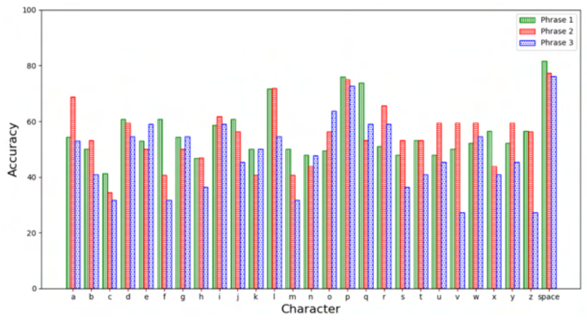


Figure 10: Figure shows the character-wise accuracy of each phrase typed by the user mentioned in Table 1 on a virtual keyboard.

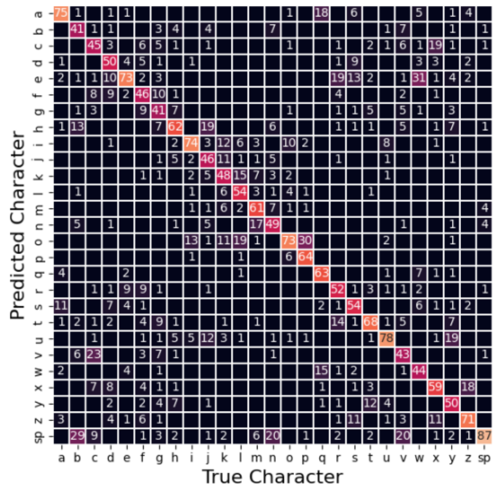


Figure 11: Confusion matrix of characters is shown in this figure. Numbers on the diagonal represent the average percentage values of true positives.

in 9b requires only two attempts for unlocking the device. Arrows on patterns in Figure 9 depict the direction (left to right) of lines. The direction of lines is predicted based on the hand pixel coordinates of the frames. If the coordinate values increase along the x-axis then the direction of the horizontal line is from left to right, otherwise vice versa. The same logic is applied to the prediction of vertical lines too.

Password Entry Process. Upon encountering the keyword ‘Gmail’, our attack model determines that the user is going to login to Gmail or Google account. Our model can decode an average of ≈ 75% of the password characters typed in the top five guesses as shown in Table 2. In addition to character prediction performance, end-to-end password inference was also evaluated. Our model could successfully break over 18% of passwords in the top 15 guesses. The readers are referred to Appendix C for details on end-to-end attack evaluation.

Table 2: Character accuracy of the HR model with top five predictions while the attack was implemented in different attack scenarios such as the user typing password, sending an email, entering text in the browser, and using Facebook Pay for performing financial transactions.

# Guess	**Phrases (in %)**	**Gmail Password (in %)**	**Email Content (in %)**	**Facebook Pay Pin (in %)**
1	57	51.39	50.88	75
2	74.5	62.73	64.24	90
3	82.31	69.58	75.18	90
4	86.93	73.45	81.02	95
5	90.5	74.73	83.59	95

email Entry Process. The HR attack model can decipher ≈ 84% of the characters typed in an email content in the top five guesses. This means that an adversary can easily gain the context of any sensitive information exchanged among friends, family, or colleagues. Table 2 shows the average percentage of characters predicted correctly during the first to fifth guess.

Pin Entry Process. After determining that the user is using a numeric keypad based on the bounds and keyboard geometry, the appropriate attack model was launched. The HR attack model was able to decipher 90% of digits in the first three guesses and could correctly predict 95% of the digits in the top five guesses for pin entry as shown in Table 2. Our results show that the HR model was able to successfully infer three out of five typed pins in the top three guesses and could correctly decipher four out of five typed pins in the top four guesses.

Surfing or Text Entry on Browser. The HR attack model achieved an average accuracy of ≈ 58% with the first guess while the users were typing phrases. Extending the predictions to the top three and five guesses, the number of characters inferred correctly was ≈ 83% and ≈ 90.5% respectively.

The average character accuracy for each phrase is shown in Figure 10. The figure depicts that the space character was the best predicted. This is because it is the only key in the row. The next best-predicted characters were ‘a’, ‘e’, and ‘l’ because these keys are on the edges of the keyboard. In general, the keys on the edges are better predicted, which makes sense as they have fewer neighbors than other keys. Most of the time, the character that was predicted wrong was ‘g’. Figure 11 shows the percentage of times a character is classified as true positive. True positive means the user-typed character and the character predicted by the HR model are both the same. As seen in this plot, false positives are usually the neighboring characters of the true character. An example of a false positive is when the user did not type a character ‘a’ while the model predicted it as ‘a’.

6 User Perception of Security and Privacy in Virtual Reality

At the end of session II of the data collection activity, volunteer participants were asked to complete the user survey study (see Appendix A for more details on the user study) by clicking on the Google survey link shared with them. The aim of this user study was to analyze and assess users' behavior and perceptions of security, and privacy in immersive environments. Figure 12 shows the summary results of our user study. The summary results show that $\approx$ 95% of the users were unaware that VR devices can be subjected to digital attacks. Around 40% of the users who participated in our study indicated their preference for using hand gestures as an input method to perform typing on the VR device compared to the use of the hand controller method. The preference of authentication systems by the users in immersive devices is still passwords, pins, or graphical-lock patterns for $\approx$ 45% of the users in our study. That being mentioned, although many users indicated that they would like biometric authentication if it is available in VR devices, current popular VR/AR devices do not provide biometric authentication mechanisms. Our study also shows that the majority of the users (55%) were comfortable performing sensitive transactions on a VR device such as online payment for an order, sending money using Facebook Pay, Google Pay, etc.

The summary results evidently show the false perception of security among the users. The demonstration of the HR attack is a challenge to users' perception of security and uncovers a serious concern and threat to the users' digital footprints. With the increased use of VR devices for typing private text and performing sensitive transactions, this paper argues the need for defensive mechanisms to ensure the users' digital security.

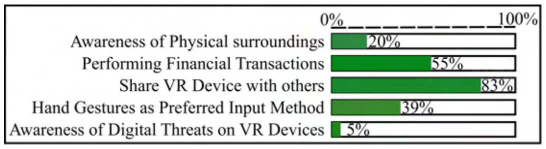


Figure 12: Summary of user responses for the questions related to user experience and perception of privacy and security on virtual reality devices is shown here.

7 Discussion

Evaluation of the HR Model with Varying Parameters – The HR attack design does not make any assumptions about the VR device type, or location (indoor or outdoor of semi-public and public places) from which the video recording is obtained. The attack utilizes the user's visible hand gestures captured in the video footage to decipher the typed text. Although we evaluated the performance of the attack on the Meta Quest2 VR headset, *Hidden Reality* attack will work on other VR headsets as well with slight modification on the known keypad layout according to the VR device used. However, investigating and evaluating the HR model performance on different VR/AR headsets, and with different brightness is part of our future work.

We also evaluated the effect of recording distance and recording angle on the attack performance. The analysis results show that the attack performance decreases when recording is done from a longer recording distance. However, the performance improves when the adversary uses a high-end camera with an optical zoom feature to capture the user's hand gestures. Also, the attack performance decreases when the recording is captured from the right side of the user compared to the recording captured from the front and left angle to the user. The detailed analysis is presented in Appendix B.

Sources of Errors and Incorrect Predictions – Although the HR attack model sometimes makes errors in typed key inferences, it reduces the search space significantly compared to a random guessing attack. The following are sources of errors in inferences made by the HR attack model.

- The model sometimes gets confused with an adjacent key to the actual key typed on the virtual keyboard. In this scenario, the attackers can generate subsequent candidate passwords by replacing the deciphered key with its adjacent keys (i.e. the key in the up, down, left, and right direction to the predicted character) to get the correct inference.
- The attack model sometimes misses detecting a click gesture or detects false clicks leading to an incorrect prediction. This is due to a sudden change in the user's typing speed within a typing session (missed clicks), and an unregistered click where the key pressed by the user is not entered on the VR device's screen (extra clicks).

Partly Obscuring Hand Gestures – The hand gestures of the targetted user can sometimes be partly obscured in video recordings captured by an adversary. This can be due to the entry of humans or objects between an attacker and the user. The HR model would miss detecting keys typed in the obscured portion of the video recording. However, for a targeted user, an attacker can easily obtain several video recordings over a few days (assuming the user password remains unchanged in that duration) to get a more confident prediction. Additionally, fused prediction decisions will be able to infer keys that might get missed using only one video recording with obscured hand gestures in general. That being mentioned, completely obscuring hand gestures while typing may serve as a countermeasure for attacks such as ours.

Human Spy vs HR Model – It could be possible for humans to monitor users' hand gestures and map their clicks onto the known keyboard layout. However, due to the need for precise hand tracking techniques for determining the user's hand location pixel coordinates, humans can lack efficiency when predicting the text typed by the users as shown in the work by Balzarotti *et al.* [1]. Our HR model, on the other hand, uses the hand tracking algorithm [24] to obtain the pixel coordinates of the hand movements and clicks to decode passwords, pins, graphical-lock patterns, and regular text typed by the users with high accuracy and efficiency.

Countermeasures and Potential Mitigation – We propose the following countermeasures to mitigate the security threats by the attacks such as the HR attack model that can be used to steal the users' sensitive information.

- Randomization techniques such as dynamically changing the key layout or using a mutated keyboard by shuffling the key locations and sizes can be used to mitigate this family of attacks [31–33]. However, the solutions like randomized and custom keyboard designs raise usability concerns [8, 34].
- VR users can choose passwords/pins with characters chosen from the center of the keyboard, which has more neighbors than the edges. This is because the HR model has exhibited low accuracy in predicting characters surrounded by neighbors as depicted in Figure 10.
- Completely or partly obscuring the hand gestures from external views while using VR devices will serve as a defense against attacks such as ours. For example, if the targetted victim is facing a wall during a VR interaction, adversaries will have a restricted view of the victim's hand gestures. This method can be used to combat attacks like ours since it requires video recordings with visible hand movements of the targeted user.

8 Related Work

Existing research shows that VR/AR devices are vulnerable to different types of attacks. For example, they can be targeted through network attacks, motion sensors-based attacks, etc.

8.1 Attacks Targeted to VR/AR Devices

Visual side-channels-based attack. The work that most closely relates to our work is perhaps the attack model by Ling *et al.* [35] to predict the passwords typed by users. There are several factors that put the HR attack apart from that designed by Ling *et al. First*, the attack model by Ling *et al.* utilizes 3D video recordings of the headset and videos of fingertip taps on the touchpad of the Samsung VR headset to be able to launch an attack. *Second*, their attack model requires prior calibration of the stereo camera using calibration algorithm [36] for recording the target user maneuvering their device and entering passwords. *Finally*, the attack by Ling *et al.* [35] relies on the user typing an anchor (e.g., go, enter) key to backtrack from that known key location. The combination of these factors makes the HR attack a completely different kind of threat compared to that presented in [35]. *On the other hand, the HR attack model relies only on the video recording of hand movements captured using consumer-grade smartphones or any video recording device, without a need for use of a high-end stereo camera, video clips of a combination of HMD and finger interaction, or calibration of the camera.* The HR attack can easily be launched on the go even by an adversary who does not possess deep technical expertise.

Motion sensors-based attack. An attack presented by Ling *et al.* [35] utilizes the data collected from the motion sensors inbuilt into the pointing device and headsets using malware to predict the passwords.

Kim *et al.* [37] proposed an interface to control VR (Virtual Reality) content, games, and animations in real-time using the user's breath and the acceleration sensor of a mobile device. Shi *et al.* [38] presented a motion sensor-based speech eavesdropping attack referred to as Face-mic. Another work on motion sensors-based attack is presented in [39]. Meteriz-Yildiran *et al.* [40] presented a key inference attack on in-air typing on AR devices. Their attack model utilizes the inbuilt motion sensors data from the AR device to track the user's hand movements. The aforementioned attacks assume the adversary has access to the VR/AR device's motion sensor data. On the other hand, the HR attack model does not make any such assumptions.

Network and WiFi signals-based attack. VR-spy developed by Arafat *et al.* [41] recognizes the virtual keystrokes using channel state information (CSI) of WiFi signals. VR-spy requires access to the user's wireless network setup such as the user's WiFi transmitter, receiver, router, and network interface card. In contrast, the HR model relies only on the short video clips of the user's hand gestures recorded using an external camera and does not require access to the user's wireless network. Yarramreddy *et al.* [42] recover forensically relevant data such as user names, user profile pictures, events, and system information using network traffic. Their attack requires the installation of Wireshark software by an adversary to access network traffic.

Other Attacks. A malicious game referred to as an escape room that deceives the user as a genuine application was designed by Nair *et al.* [43]. This application secretly collects user information such as height and wingspan, demographics like age and gender biometrics, network details, etc. within a few minutes of the gameplay by the target user.

8.2 Attacks on Other Smart Wearables

Some of the related work that includes the attacks performed on wearable devices, and smartphones to infer the text typed by the users is presented below.

Video recording-based attack. Balzarotti *et al.* [1] presented an attack model to recover the text typed by the users while they were performing the text entry process on the physical keyboard. Their model was able to predict 82% of the language-based and context-based text in the top 50 guesses utilizing the video recording captured. Attacks presented by Shukla *et al.* [3,44] and Jingchao et al. [45] infers the pins and passwords typed by the user utilizing the video footage of the part of the user's hands and the backside of the smartphone or tablet on which the pin or password typing is performed. Typing on a smartphone or tablet device requires users to touch and type on the device's screen. Typing on a VR device is completely in the air and requires users to locate and perform pinch and type operations. The attack models presented in [3, 44, 45] will fail to detect pinch and click operations and hence will fail to work on VR devices. In their work, Sabra *et al.* [46] show that an adversary can decipher the text typed by a user when they type on their desktop/laptop keyboard while attending a video meeting. Their attack assumes that the attacker can join the user's video meeting and hence can access the video feed remotely. The HR model accesses the user's typing hand gestures from an external camera while the user types on the device's virtual screen to access any application.

Chen *et al.* introduced an attack, EyeTell [47], that utilizes video recording of the user's eye movements to infer the keys typed by the user on their mobile devices. This kind of attack cannot be launched on a VR device because the user's eyes are obscured while wearing the VR headset and hence the attacker will not have access to the user's eye movements to launch the attack. Some of the other related video recording-based attacks are presented in [48–51].

Reflection-based attack. Yue *et al.* [52, 53] presented a deformable part-based model (DPM) to infer the passwords typed on the touch screen utilizing the shadow formation around the fingertips. The DPM model requires prior training of the images to launch it. A similar attack that recovers the text typed by the user utilizing the reflections is presented by Ye *et al.* [54]. Raguram *et al.* [55] and Backes *et al.* [56] utilized the captured reflections to reconstruct the typed text. In their work, Xu *et al.* [57] demonstrated that their attack model could detect the keystroke inputs by exploiting repeated reflections in the user's eyeball. These attacks cannot be extended to VR devices because reflections in the eyeball cannot be observed by an adversary while the user wears Head Mounted Displays (HMD) devices.

Microphone and motion-sensors based attack. PIN Skimmer presented by Simon *et al.* [58] infers user-typed pins on a smartphone by making use of video recordings and accessing the microphone data. A Trojan application for the Android platform, TapLogger [59] predicts the password and numbers entered during a phone call. [2, 44, 60–65] are some of the other motion-based attacks for predicting the characters typed by users on non-XR devices.

Existing Work vs the HR Attack Model.
The related work requires (i) model training for the target user, (ii) motion sensors or microphone data, (iii) specially designed hardware components, or the installation of software in the target user's devices. The aforementioned models are designed for a specific attack scenario where the user just enters a pin, password, or text.

Hidden Reality model utilizes only the spatiotemporal dynamics of the target user's visible hand gestures captured from video clips to infer the text typed by the user. HR model exposes serious security threats in various key entry scenarios: (1) pin entry process, (2) alphanumeric password entry process, (3) graphical pattern lock input, (4) email entry, and (5) surfing or text entry on the browser. Our model does not need specific model training for the target user.

9 Conclusions

In this paper, we introduced a novel video-based side-channel attack, *Hidden Reality*, to immersive environments. The Hidden Reality attack[6] model does not require any additional information from the user's screen display and relies only on the spatio-temporal dynamics of the target user's hand gestures. Also, our model does not require specific training for a targeted user to launch the attack. Very high success rates of our attack model expose a serious security threat as the attacks such as ours can easily be launched without raising any suspicion by the user. Our user survey study shows that VR users have very limited awareness of the real world while they are immersed in virtual reality (VR). Hence, a person with malicious intent can easily obtain video footage of the targeted user's hand movements. In addition, our user study also indicates that pins, passwords, and graphical patterns locks are still a preferred authentication mechanism for many users.

Attack models such as Hidden Reality challenges the users' false perception of security and uncovers a serious security threat to their digital footprints. With the increased use of VR devices for performing private and sensitive transactions, this paper argues in support of the need for more stringent defensive mechanisms to protect the users' security and privacy.

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) under grants CNS-2055621, IIS-1852579, OAC-2139358, CNS-2201465, and CNS-2152669. James Wheelock carried out this work while he was associated with the University of Wyoming.

[6]Although the *Hidden Reality* attack uncovers major security risks, the attack model could be utilized by potential adversaries as well. The purpose of this work is to alert the security community and VR users of possible security threats posed by attacks such as ours. We do not anticipate any other major ethical or security risks arising from our work.

References

[1] 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.

[2] F. Maggi, Alberto Volpatto, Simone Gasparini, Giacomo Boracchi, and Stefano Zanero. A fast eavesdropping attack against touchscreens. In *2011 7th International Conference on Information Assurance and Security (IAS)*, pages 320–325, 2011.

[3] Diksha Shukla, Rajesh Kumar, Abdul Serwadda, and Vir V Phoha. Beware, your hands reveal your secrets! In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 904–917, 2014.

[4] Marian Harbach, Emanuel von Zezschwitz, Andreas Fichtner, Alexander De Luca, and Matthew Smith. It's a hard lock life: A field study of smartphone (Un)Locking behavior and risk perception. In *10th Symposium On Usable Privacy and Security (SOUPS 2014)*, pages 213–230, Menlo Park, CA, July 2014. USENIX Association.

[5] Matt Brown. Exploring the magic behind the htc vive controller. https://www.vrheads.com/exposing-magic-behind-htc-vive-controller, 2016.

[6] Meta Inc. Meta-point and pinch. https://tinyurl.com/mr23z69m, accessed: October, 2022.

[7] Kseniia Palin, Anna Maria Feit, Sunjun Kim, Per Ola Kristensson, and Antti Oulasvirta. How do people type on mobile devices? observations from a study with 37,000 volunteers. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '19, New York, NY, USA, 2019. Association for Computing Machinery.

[8] Brad L. Aday Xavier A. Gutierrez John D. Platt Young Sam Ryu, Do Hyong Koh. Usability evaluation of randomized keypad. *Journal of User Experience*, 5:65–75, 2010.

[9] Ceenu George, Mohamed Khamis, Emanuel Zezschwitz, Marinus Burger, Henri Schmidt, Florian Alt, and Heinrich Hussmann. Seamless and secure vr: Adapting and evaluating established authentication systems for virtual reality. 02 2017.

[10] Florian Mathis and M. Khamis. Privacy, security and safety concerns of using hmds in public and semi-public spaces. 2019.

[11] HTC. Htc vive. https://www.vive.com/us/, accessed: August, 2022.

[12] Meta Inc. Buy meta quest 2. https://tinyurl.com/599jxvdd, accessed: October, 2022.

[13] Weise Sarah and Mshar Andrew. Virtual reality and the banking experience. *Digital Banking*, 1:146–152, 2016.

[14] Bhone Kyaw, Nakul Saxena, Paul Posadzki, Jitka Všetečková, Charoula Nikolaou, Pradeep George, Ushashree Divakar, Italo Masiello, Andrzej Kononowicz, Nabil Zary, and Lorainne Tudor Car. Virtual reality for health professions education: Systematic review and meta-analysis by the digital health education collaboration (preprint). 11 2018.

[15] The growing use of virtual reality in cognitive rehabilitation: Fact, fake or vision? a scoping review. *Journal of the National Medical Association*, 111(4):457–463, 2019.

[16] Max M. North, Sarah M. North, and Joseph R. Coble. Virtual reality therapy: An effective treatment for the fear of public speaking. international journal of virtual reality, 1998.

[17] Gabriele Pizzi, Daniele Scarpi, Marco Pichierri, and Virginia Vannucci. Virtual reality, real reactions?: Comparing consumers' perceptions and shopping orientation across physical and virtual-reality retail stores. *Computers in Human Behavior*, 96, 02 2019.

[18] E. Edgerton C. Wilson D. Hamilton, J. McKechnie. Immersive virtual reality as a pedagogical tool in education: a systematic literature review of quantitative learning outcomes and experimental design. *Journal of Computers in Education*, 2021.

[19] Maki Sugimoto. *Extended Reality (XR:VR/AR/MR), 3D Printing, Holography, A.I., Radiomics, and Online VR Tele-Medicine for Precision Surgery*, pages 65–70. Springer Nature Singapore, Singapore, 2021.

[20] Bobby Carlton. U.s. air force using vr for sexual assault prevention training. https://vrscout.com/news/us-air-force-vr-sexual-assault-training/, 2021.

[21] Ilya Minyaev, Matti Pouke, Johanna Ylipulli, and Timo Ojala. Implementation of a virtual reality interface for a public library. In *Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia*, MUM 2018, page 513–519, New York, NY, USA, 2018. Association for Computing Machinery.

[22] Christian Mai and Mohamed Khamis. Public hmds: Modeling and understanding user behavior around public head-mounted displays. In *Proceedings of the 7th ACM International Symposium on Pervasive Displays*, pages 1–9, 2018.

[23] Kaspersky Lab. Brute force attack: Definition and examples. `https://www.kaspersky.com/resource-center/definitions/brute-force-attack`, accessed: October, 2022.

[24] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *CoRR*, abs/2006.10214, 2020.

[25] W. N. Francis and H. Kucera. Brown corpus manual. `http://korpus.uib.no/icame/brown/bcm.html`, 1979.

[26] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, February 1966.

[27] Robert Miller, Natasha Kholgade Banerjee, and Sean Banerjee. Temporal effects in motion behavior for virtual reality (vr) biometrics. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 563–572, 2022.

[28] LRC. Memorable password generator. `https://tinyurl.com/fu5f3rv4`, accessed: October, 2022.

[29] Guixin Ye, Zhanyong Tang, Dingyi Fang, Xiaojiang Chen, Kwang Kim, Ben Taylor, and Zheng Wang. Cracking android pattern lock in five attempts. 02 2017.

[30] Seunghun Cha, Sungsu Kwag, Hyoungshick Kim, and Jun Ho Huh. Boosting the guessing attack performance on android lock patterns with smudge attacks. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 313–326, 2017.

[31] Anindya Maiti, Murtuza Jadliwala, and Chase Weber. Preventing shoulder surfing using randomized augmented reality keyboards. *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*.

[32] Ruide Zhang, Ning Zhang, Changlai Du, Wenjing Lou, Yiwei Thomas Hou, and Yuichi Kawamoto. Augauth: Shoulder-surfing resistant authentication for augmented reality. *2017 IEEE International Conference on Communications (ICC)*, pages 1–6, 2017.

[33] S. Kumar, R. Ramya, R. Rashika, and Renu Renu. *A Survey on Graphical Authentication System Resisting Shoulder Surfing Attack*, pages 761–770. 01 2021.

[34] CHRIS HOFFMAN. Alternative keyboard layouts explained: Should you switch to dvorak or colemak? `https://tinyurl.com/44yntvwa`, 2017, accessed: October, 2022.

[35] Zhen Ling, Zupei Li, Chen Chen, Junzhou Luo, Wei Yu, and Xinwen Fu. I know what you enter on gear vr. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 241–249, 2019.

[36] Alex Zelinsky. Learning opencv—computer vision with the opencv library (bradski, g.r. et al.; 2008)[on the shelf]. *IEEE Robotics Automation Magazine*, 16(3):100–100, 2009.

[37] Jong-Hyun Kim and Jung Lee. Controlling your contents with the breath: Interactive breath interface for vr, games, and animations. *PLoS ONE*, 15, 2020.

[38] Cong Shi, Xiangyu Xu, Tianfang Zhang, Payton Walker, Yi Wu, Jian Liu, Nitesh Saxena, Yingying Chen, and Jiadi Yu. Face-mic: inferring live speech and speaker identity via subtle facial dynamics captured by ar/vr motion sensors. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 478–490, 2021.

[39] Shiqing Luo, Xinyu Hu, and Zhisheng Yan. Hologger: Keystroke inference on mixed reality head mounted displays. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 445–454, 2022.

[40] Ülkü Meteriz-Yıldıran, Necip Fazıl Yıldıran, Amro Awad, and David Mohaisen. A keylogging inference attack on air-tapping keyboards in virtual environments. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 765–774, 2022.

[41] Abdullah Al Arafat, Zhishan Guo, and Amro Awad. Vr-spy: A side-channel attack on virtual key-logging in vr headsets. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 564–572. IEEE, 2021.

[42] Ananya Yarramreddy, Peter Gromkowski, and Ibrahim Baggili. Forensic analysis of immersive virtual reality social applications: A primary account. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 186–196, 2018.

[43] Vivek Nair, Gonzalo Munilla Garrido, and Dawn Song. Exploring the unprecedented privacy risks of the metaverse. *arXiv*, abs/1207.0016, 2022.

[44] Diksha Shukla and Vir V Phoha. Stealing passwords by observing hands movement. *IEEE Transactions on Information Forensics and Security*, 14(12):3086–3101, 2019.

[45] Jingchao Sun, Xiaocong Jin, Yimin Chen, Jinxue Zhang, Rui Zhang, and Yanchao Zhang. Visible: Video-assisted keystroke inference from tablet backside motion. 01 2016.

[46] Mohd Sabra, Anindya Maiti, and Murtuza Jadliwala. Zoom on the keystrokes: Exploiting video calls for keystroke inference attacks. *CoRR*, abs/2010.12078, 2020.

[47] Yimin Chen, Tao Li, Rui Zhang, Yanchao Zhang, and Terri Hedgpeth. Eyetell: Video-assisted touchscreen keystroke inference from eye movements. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 144–160, 2018.

[48] Guixin Ye, Zhanyong Tang, Dingyi Fang, Xiaojiang Chen, Willy Wolff, Adam Aviv, and Zheng Wang. A video-based attack for android pattern lock. *ACM Transactions on Privacy and Security*, 21:1–31, 07 2018.

[49] Tao Chen, Michael Farcasin, and Eric Chan-Tin. Smartphone passcode prediction. *IET Information Security*, 12, 04 2018.

[50] Kiran S Balagani, Mauro Conti, Paolo Gasti, Martin Georgiev, Tristan Gurtler, Daniele Lain, Charissa Miller, Kendall Molas, Nikita Samarin, Eugen Saraci, et al. Silk-tv: Secret information leakage from keystroke timing videos. In *Computer Security: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I 23*, pages 263–280. Springer, 2018.

[51] Qinggang Yue, Zhen Ling, Xinwen Fu, Benyuan Liu, Wei Yu, and Wei Zhao. My google glass sees your passwords! In *Computer Science*, 2014.

[52] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao. Blind recognition of touched keys on mobile devices. In *In Proceedings of the CM SIGSAC Conference on Computer and Communications Security*, pages 1403 – 1414, 2014.

[53] Q. Yue, Z. Ling, W. Yu, B. Liu, and X. Fu. Blind recognition of text input on on mobile devices via natural language. In *Workshop on Privacy-Aware Mobile Computing*, pages 19 – 24, 2015.

[54] G. Ye, Z. Tang, D. Fang, X. Chen, K. Kim, B. Taylor, and Z. Wang. Cracking android pattern lock in five attempts. In *In Proceedings of the Network and Distributed System Security Symposium (NDSS'17)*, 2017.

[55] Rahul Raguram, Andrew White, Dibyendusekhar Goswami, Fabian Monrose, and Jan-Michael Frahm. ispy: Automatic reconstruction of typed input from compromising reflections. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 527–536, 10 2011.

[56] 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, 2008.

[57] Yi Xu, Jared Heinly, Andrew M. White, Fabian Monrose, and Jan-Michael Frahm. Seeing double: Reconstructing obscured typed input from repeated compromising reflections. CCS '13, page 1063–1074, New York, NY, USA, 2013. Association for Computing Machinery.

[58] 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.

[59] Zhi Xu, Kun Bai, and Sencun Zhu. Taplogger: inferring user inputs on smartphone touchscreens using on-board motion sensors. In *WISEC '12*, 2012.

[60] 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)*, San Francisco, CA, August 2011. USENIX Association.

[61] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: Password inference using accelerometers on smartphones. New York, NY, USA, 2012. Association for Computing Machinery.

[62] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. Tapprints: Your finger taps have fingerprints. New York, NY, USA, 2012. Association for Computing Machinery.

[63] Chen Wang, Xiaonan Guo, Yingying Chen, Yan Wang, and Bo Liu. Personal pin leakage from wearable devices. *IEEE Transactions on Mobile Computing*, 17(3):646–660, 2018.

[64] Benxiao Tang, Zhibo Wang, Run Wang, Lei Zhao, and Lina Wang. Niffler: A context-aware and user-independent side-channel attack system for password inference. *Wireless Communications and Mobile Computing*, 2018:1–19, 05 2018.

[65] 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.

[66] pyenchant 3.2.2. https://pypi.org/project/pyenchant/, accessed: January, 2023.

[67] WILLIAM J. BURNS. Common password list. https://www.kaggle.com/datasets/wjburns/common-password-list-rockyoutxt, accessed: January, 2023.

[68] 83% of americans are using weak passwords. https://press.avast.com/83-of-americans-are-using-weak-passwords, accessed: February, 2023.

[69] 55 important password statistics you should know: 2023 breaches reuse data. https://financesonline.com/password-statistics/, accessed: February, 2023.

Appendices

A User Study Questionnaire

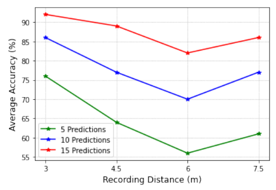
Post completion of the data collection activity for both the recording sessions, the volunteer participants were asked to fill in a user survey on their behavior and perception of the security of VR devices. The survey questionnaire was shared with volunteer participants through a Google forms link in their email. The purpose of this study was to understand users' behavior and perceptions of the security related to VR devices. Table 3 presents a detailed list of questions asked to the participants in our user perception study.

Table 3: User Study Questionnaire on Security and Privacy of VR Devices.

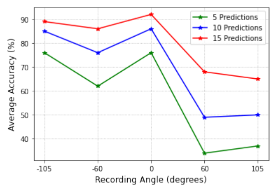
#	Survey Questions
1	How often do you use a VR device?
2	Which input method do you prefer to use when typing on a VR device? Controller (using hand controller to click and type) (or) Hand tracking (locating the characters using fingertips and pinching to type). This was the method used in our experiments.
3	In your opinion, what type of authentication system is most secure for a VR device?
4	Are you comfortable using a shared VR device or sharing your own VR device with friends and family ?
5	Would you be comfortable making financial transactions using a VR device, such as ordering food, buying products on Amazon, sending money through google pay, facebook pay, etc. ?
6	I am aware of what is happening around me in the real world when I wear a VR headset
7	Are you aware that a VR device can be subjected to a digital attack known as data theft? Data theft is an act of adversaries to gain access to the users' private and sensitive information.
8	Rate your overall experience with our data collection experiments.

B Attack Performance with Varying Recording Distances and Recording Angles

Experiment Design – To analyze the effect of recording distances and recording angles on the attack performance for password cracking, we recruited eight volunteer participants. We collected additional video clips from three different recording distances (viz. 3 m, 4.5 m, and 6 m) constituting a total of 24 video clips. To analyze the effect of recording angle, we collected data from seven[7] volunteer participants from five different recording angles (viz. -105 degrees, -60 degrees, 0 degrees, 60 degrees, and 105 degrees) constituting a total of 35 videos. Since the Samsung S20 smartphone camera was not able to capture visible hand gestures from a recording distance longer than 4 meters, we used Nikon D3400 camera with an optical lens for videotaping in these experiments. We kept the optical zoom setting of the Nikon camera to 1X and also kept other parameters fixed to capture hand movements from various distances and angles. 0 degree signifies the recording captured from the front of the user where the attacker and the user face each other. The negative angle represents the left side of the user at the respective angle value while the positive angle denotes the angle on the right side of the user. We kept the recording distance fixed at 3m while varying the recording angle. Similarly, we kept the recording angle at 0 degrees while varying the recording distance.



(a) Attack performance for videos captured from different recording distances.



(b) Attack performance for videos captured from different recording angles.

Figure 13: The effect of recording distances and recording angles on the attack performance. Figure 13a shows that the increase in recording distance decreases the performance of the attack model. However, an adversary who utilizes a camera with the optical zoom feature will still be able to achieve good attack performance as shown for prediction performance from a recording distance of 7.5 m using 2.5X optical zoom. In Figure 13b, 0 degrees signifies that the user and the attacker are facing each other. A negative angle represents that the videos were recorded from the left side of the user while a positive angle means the videotaping is performed from the right side of the user at the respective angles.

Attack Performance with Recording Distance – We evaluated the performance of the HR attack model on the video recordings captured from various distances. Figure 13a

[7]One of the recruited volunteer participants did not return to participate in the data collection for the recording angle experiment.

presents the average attack performance at different recording distances. The evaluation results show that the attack performance decreases by increasing the camera-to-device recording distance. We believe that the attack performance decreases due to the reduced size of hand gestures in the video clip which makes it difficult for the model to detect click (or pinch) gestures.

We also recorded *eight* videos from a recording distance of 7.5 m with 2.5x optical zoom using a Nikon D3400 camera to evaluate the attack performance from a longer recording distance while utilizing the optical zoom feature. This is to simulate the scenario of an adversary that uses a high-end camera with an optical zoom to attack a targeted user from a long distance without raising suspicion. Figure 13a shows that the average character inference performance increases if an attacker uses a camera with optical zoom capability when capturing the hand movements from a longer recording distance. This is due to the increase in the visibility of the targeted user's hand gestures.

Attack Performance with Recording Angle – We evaluated the attack performance from different recording angles using 35 videos captured from different angles as discussed under Experiment Design. Figure 13b shows the average performance at different recording angles. As shown in the Figure, the HR attack model could achieve significantly high inference accuracy using videos captured from the left side of the user. The HR model recorded a comparatively low prediction accuracy for the videos recorded from the right side of the user. We observed that this is because knuckles/parts of other fingers partly obscured the visibility of the index and the thumb anchor points required to detect click gestures.

C End-to-end Performance Evaluation of the HR Attack on Password Typing

We conducted an end-to-end attack evaluation for the password-cracking scenario by utilizing the deciphered characters from the top 5 predictions of the HR model. Here, we evaluate the performance of the attack model for complete password prediction.

Candidate Passwords Generation – For this, we used the following procedure.

1. We first applied our HR attack model to the recorded video. The password inferred by the HR attack model was considered the first candidate password.

2. The attack model then finds the character that has the highest probability of getting confused with their adjacent keys. This was detected by checking the distance of the predicted point from the predicted key center on the projected keypad (see Section 3.1.4). We generated the next four candidate passwords by replacing the most confused inferred keys with the closest adjacent key. This gave us our top five password guesses.

3. We then used a dictionary model [66] that uses a leaked passwords dataset [67]. We used this model with input as – (i) the first guess to generate 6^{th}, 7^{th}, and 8^{th} candidate passwords, (ii) the second guess obtained in Step 2 to generate 9^{th}, 10^{th}, and 11^{th} candidate passwords, (iii) the third guess to generate 12^{th}, 13^{th}, and 14^{th} candidate passwords, (iv) the fourth guess to generate 15^{th}, 16^{th}, and 17^{th} candidate passwords, and (v) the fifth generated password in Step 2 was used to predict 18^{th}, 19^{th}, and 20^{th} guesses. The reason we used each of the guessed passwords generated in Step 2 as input for the password dictionary model is that the top five generated passwords are roughly equally correct. The HR model only replaces the most confused key with its closest adjacent key one at a time to generate a new guess.

End-to-end Password Inference Performance – We generated the top 5, top 10, top 15, and top 20 guesses using the candidate password generation procedure. The evaluation results show that the HR attack model could break $\approx 23\%$ of the passwords in the top 20 guesses (see Table 4). The passwords typed in our experiments were computer-generated random passwords, whereas, the users generally use easier passwords that they can remember and type every time to access their device [68, 69]. Our experiment analysis shows that the available leaked password datasets typically utilized to learn relationships and patterns of passwords do not generalize well to completely random passwords, resulting in low end-to-end password cracking performance. We believe that the password prediction performance will only improve with actual user passwords.

Another source of error in password prediction is due to the missed/extra clicks detected by the HR model (see Section 7). We noticed that the candidate password generation mechanism using the leaked password dataset is not very effective in correcting missed/extra clicks for random passwords. However, using the HR model with multiple videos of password typing, and using the fused decision to detect clicks would help accurately locate actual clicks by the user. Moreover, using the leaked passwords-based candidate password generation mechanism on the clicks detected from fused decisions would improve the end-to-end attack performance on actual passwords.

Table 4: The end-to-end accuracy of the top 5, top 10, top 15, and top 20 guesses for the password-cracking scenario.

End-to-end Password Level Inference Performance				
Top n Guesses	**5**	**10**	**15**	**20**
Average Inference Accuracy	0 %	9.0%	18.2%	22.7%