GAZEPLOIT: Remote Keystroke Inference Attack by Gaze Estimation from Avatar Views in VR/MR Devices

Hanqiu Wang
Department of Electrical and Computer Engineering, University of Florida,
Gainesville, FL, USA
wanghanqiu@ufl.edu

Zihao Zhan
Department of Computer Science, Texas Tech University,
Lubbock, TX, USA
zihao.zhan@ttu.edu

Haoqi Shan
CertiK,
New York, NY, USA
haoqi.shan@certik.com

Siqi Dai
Department of Electrical and Computer Engineering, University of Florida,
Gainesville, FL, USA
dais@ufl.edu

Maximilian Panoff
Department of Electrical and Computer Engineering, University of Florida,
Gainesville, FL, USA
m.panoff@ufl.edu

Shuo Wang
Department of Electrical and Computer Engineering, University of Florida,
Gainesville, FL, USA
shuo.wang@ece.ufl.edu

Abstract

The advent and growing popularity of Virtual Reality (VR) and Mixed Reality (MR) solutions have revolutionized the way we interact with digital platforms. The cutting-edge gaze-controlled typing methods, now prevalent in high-end models of these devices, e.g., Apple Vision Pro, have not only improved user experience but also mitigated traditional keystroke inference attacks that relied on hand gestures, head movements and acoustic side-channels. However, this advancement has paradoxically given birth to a new, potentially more insidious cyber threat, **GAZEPLOIT**.

In this paper, we unveil **GAZEPLOIT**, a novel eye-tracking based attack specifically designed to exploit these eye-tracking information by leveraging the common use of virtual appearances in VR applications. This widespread usage significantly enhances the practicality and feasibility of our attack compared to existing methods. **GAZEPLOIT** takes advantage of this vulnerability to remotely extract gaze estimations and steal sensitive keystroke information across various typing scenarios—including messages, passwords, URLs, emails, and passcodes. Our research, involving 30 participants, achieved over 80% accuracy in keystroke inference. Alarmingly, our study also identified over 15 top-rated apps in the Apple Store as vulnerable to the **GAZEPLOIT** attack, emphasizing the urgent need for bolstered security measures for this state-of-the-art VR/MR text entry method.

CCS Concepts

• **Security and privacy → Privacy protections**; **Domain-specific security and privacy architectures**; • **Human-centered computing → Virtual reality**.

Keywords

Virtual Reality, Gaze Estimation, Keystroke Inference, User Privacy

ACM Reference Format:
Hanqiu Wang, Zihao Zhan, Haoqi Shan, Siqi Dai, Maximilian Panoff, and Shuo Wang. 2024. GAZEPLOIT: Remote Keystroke Inference Attack by Gaze Estimation from Avatar Views in VR/MR Devices. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3690285

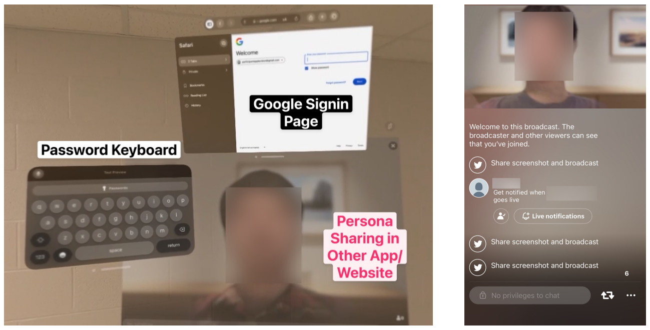
1 Introduction

The proliferation of Virtual and Mixed Reality (VR/MR) headsets in the consumer market is a recent phenomenon, despite the existence of the technology for many years. With an impressive user base of 171 million devices and counting [24], VR/MR technology is quickly breaking free from its initial entertainment confines. Today, it is being utilized in a wide array of fields, including education, professional training, social interaction, and remote work environments. Given that text entry is a fundamental interaction with electronic devices, VR devices commonly incorporate virtual keyboards to facilitate efficient typing tasks.

The evolution of typing methods on VR's virtual keyboards is noteworthy. Earlier VR devices, such as the HTC Vive and Oculus Rift, employed controllers [36] as optical pointers for key selection on a virtual keyboard [10]. More recent devices like the Meta Quest have transitioned to using external cameras to capture users' hand gestures for typing. In the latest VR devices, eye-tracking technology has gained prominence, enabling the potential implementation of gaze-controlled typing, an intuitive, efficient, and user-friendly text entry method. Major manufacturers, including Microsoft, Meta, and HTC, have acknowledged the potential of this method. Recent research [13, 38, 50] has further confirmed its feasibility. Notably, Apple's latest VR device, the Apple Vision Pro MR headset, has integrated gaze-controlled typing as the primary text entry method, marking a significant shift in the industry [19].

Given that text entry often involves confidential data such as passwords, PINs, and private messages, recent researches have identified vulnerabilities associated with different virtual keyboard typing methods. Prior studies [4, 15, 27, 29, 34, 42] have demonstrated the ability to recover keystrokes from side-channel information, captured by motion sensors on the headset or through external optical and acoustic recordings of users. These investigations focused on controller-based, head-pose-based, and hand-gesture-based typing methods. The approach taken by Apple's Vision Pro, which utilizes an eye tracker as the pointer, minimizes the need for hand and controller movements as well as head movements, making many traditional attack vectors ineffective.

While the integration of gaze-controlled typing methods in Apple Vision Pro has significantly enhanced user interaction, it also introduces new security and privacy risks. Apple recognizes the sensitivity of eye-tracking data and has implemented stringent measures to confine its access [2]. This data is processed exclusively at the system level, with applications only receiving the final computation results, not the underlying camera data or information used to generate them. Moreover, Apple's security measures stipulate that the virtual keyboard is managed only by visionOS [3] and cannot be accessed or controlled by any app. Despite these precautions, our research uncovers a significant vulnerability associated with the virtual avatar technologies like Apple's Persona. These technologies, while enhancing digital communication, can inadvertently expose critical facial biometrics, including eye-tracking data, through video calls where the user's virtual avatar mirrors their eye movements. Leveraging this vulnerability, we developed GAZEploit, a novel attack that can infer eye-related biometrics from the avatar image to reconstruct text entered via gaze-controlled typing. This finding shows that even high-level abstractions provided to applications and users can be reverse-engineered to reveal sensitive data at the system level, effectively bypassing Apple's security measures such as the app-inaccessible virtual keyboard and eye-tracking data.


Google Signin Page
Password Keyboard
Persona Sharing in Other App/ Website
Welcome to this broadcast. The broadcaster and other viewers can see that you've joined.
Share screenshot and broadcast
Get notified when goes live
Live notifications
Share screenshot and broadcast
Share screenshot and broadcast
No privileges to chat

(a) (b)

Figure 1: (a) A victim is logging into a Google account while sharing the Persona. (b) An attacker viewing the victim's Persona from a live video on X.

The GAZEploit attack leverages the vulnerability inherent in gaze-controlled text entry when users share a virtual avatar, as illustrated in Figure 1. Virtual avatars, whether shared through video calls, online meeting apps, live streaming platforms, or potentially malicious websites, pose a significant privacy risk by potentially exposing user information such as login credentials. By remotely capturing and analyzing the virtual avatar video, an attacker can reconstruct the typed keys. Notably, the GAZEploit attack is the first known attack in this domain that exploits leaked gaze information to remotely perform keystroke inference.

To evaluate the effectiveness of our GAZEploit attack, we conducted experiments with 30 participants. This involved collecting data from videos recorded during Zoom meetings in two sessions. The data collection was designed to mirror real-world user behavior closely, with minimal constraints imposed on participants. Our evaluation demonstrated that our attack model achieved more than 85.9% precision and 96.8% recall in identifying click candidates within the video clips **without any prior knowledge of victims' personal typing habits nor their typing speed**. Key findings from the top-5 keystroke inference accuracy include: 92.1% for messages, 77.0% for passwords, 73.0% for passcodes (PIN), and 86.1% for email, URL, and webpage text entry.

To summarize, the main contributions of this paper are:

- We analyze popular VR/MR devices, identifying two new attack scenarios and associated attack vectors exploiting eye-tracking systems in benign applications such as conference apps and public broadcasts.
- We develop the GAZEploit attack, the first known method exploiting the vulnerabilities discovered in our research to perform remote keystroke inference on VR/MR devices.
- We evaluate the GAZEploit attack using a dataset collected from 30 participants in real-world VR settings demonstrating over 80% precision in top 5 keystroke guesses.
- We present end-to-end attack scenarios [1] and discuss the security implications of our findings. We also propose potential countermeasures for both VR device manufacturers and developers.

The organization of the paper is as follows: Section 2 provides an overview of the VR device, its key features, and the gaze estimation techniques utilized in this attack. Section 3 elaborates on the identified vulnerabilities and the associated threats. Section 4 offers a comprehensive demonstration of the preliminary attack, including detailed discussions of the steps, challenges, and methods involved. Section 5 describes our approach to data collection, while Section 6 presents the performance and accuracy results of our attack model. In Section 7, we discuss potential countermeasures and significant insights. Section 8 reviews relevant literature, and Section 9 concludes the paper.

2 Background

2.1 Eye trackers for VR/MR and Text Entry

In this section, we explore the underlying technologies and features of modern VR/MR devices that are pivotal to both enhancing user interaction and presenting new cybersecurity challenges.

VR Eye Trackers and Their Usage Advanced VR/MR devices such as the Apple Vision Pro [19], PSVR 2 [43], and Meta Quest Pro [33] integrate eye trackers into their headsets. This technology

[1]Readers can view our practical attack scenarios and associated video clips by visiting https://sites.google.com/view/Gazeploit/.

operates by using cameras and sensors near the lenses to capture detailed images of the eyes. These images, calibrated to each individual's unique anatomical features, are then processed to generate eye-tracking data, including metrics like pupil size, eye openness, and gaze direction. From these basic measurements, complex features such as fixations, saccades, smooth pursuits, and scanpaths can be derived [5]. For further explanation, fixations represent the concentration of a user's gaze on a specific area over a certain period, while saccades are the rapid eye movements between these fixations. The sequence of these movements and fixations is known as a scanpath, while smooth pursuits refer to the eye's movement while tracking moving objects.

The integration of these features into VR/MR systems significantly enhances user experience and system efficiency [37]. By enabling users to interact with and navigate the virtual environment through eye movements, the technology provides a more immersive experience. For instance, devices like the Meta Quest Pro allow avatars to mimic realistic eye contact and facial expressions, enhancing the authenticity of virtual conversations [21]. Similarly, in the Apple Vision Pro, eye tracking is used to animate the avatar's eyes accurately, thereby improving the realism of social interactions in virtual settings.

Gaze-controlled Typing in VR Building on the eye-tracking capabilities discussed in the previous section, developers have proposed gaze-assisted typing in VR devices [50]. The Apple Vision Pro represents a significant step in this direction, being the first VR device to incorporate an eye tracker as the default keystroke entry method at the system level. The Vision Pro offers two typing methods. The first employs eye gaze to select keys, with a finger snap to confirm. The second method involves poking in mid-air at the virtual keyboard using a single finger [20]. This differs from air-tapping, which uses multiple fingers and is generally less preferred due to its complexity and lower precision [16]. The effectiveness of gaze-controlled typing is also supported by participants' feedback, All favored it for its higher accuracy, the flexibility it offers in positioning the virtual keyboard, and the reduced need for extensive hand or body movements.

2.2 Virtual Avatar and Virtual Camera

Advanced VR systems often allow users to create digital avatars, which serve as their virtual representations. These avatars vary in complexity, from simple head-and-hand depictions to fully detailed human-like characters, depending on the VR system and its purpose. For example, Meta VR devices have a feature [32] called "Metaverse Avatar" enabling users to interact as 3D avatar in apps like Zoom and Horizon Workrooms. Similarly, popular VR games like VRChat [22] and Roblox [11] use avatars to facilitate user interactions in virtual environments.

In the Apple Vision Pro, the Persona [39] feature takes this concept further by displaying a dynamic representation of the user's face and hands, capturing facial expressions and eye movements with high fidelity using the integrated eye-tracker. Unlike other VR systems that use 3D avatars, Persona operates more like a 2D camera view. This allows apps requiring camera access to use Persona once granted permission by the user, even if they aren't specifically designed for VR. Websites can also access Persona, extending its usability beyond typical VR contexts. However, this flexibility also presents additional security risks. As discussed earlier in Section 1, VisionOS privacy standards protect sensitive eye-gaze data. Yet, developers may overlook that this data could be exposed through the Persona view, creating a vulnerability that our GAZEploit attack exploits.

2.3 Appearance-based Eye Gaze Estimation

Appearance-based eye gaze estimation is a method used in computer vision and human-computer interaction that determines where a person is looking based solely on images of their eye or face. Thus, it offers greater flexibility, but can also result in reduced accuracy due to variables like lighting changes or head movements [9]. To achieve this, modern gaze estimation techniques frequently utilize deep learning architectures, with convolutional neural networks (CNNs) [47] being prevalent due to their strong performance with image data. Additionally, other architectures such as recurrent neural networks (RNNs) and Transformers are also being investigated [9]. The inherent flexibility and minimal hardware requirements of appearance-based gaze estimation make it particularly suited for environments where traditional tracking devices are impractical. However, this also opens up significant privacy concerns, as these methods can be exploited by attackers to extract confidential gaze data [12].

3 Threat Model

3.1 Attack Scenarios

Our threat model is based on a VR user who participates in a variety of activities while sharing their virtual avatar, including multiple text entry sessions via eye tracking. The user can freely arrange application windows within the virtual space, place them in any direction at desired distances, and select their own text content for input. We assume that **the attacker only needs a video recording of the user's virtual avatars during these activities, which can be obtained through various channels such as video conferencing, live streaming** that request access to the user's virtual representation. The virtual avatars only contain users facial expressions and eye gaze information. Hand gestures are not needed in the captured video. No additional knowledge about the user's behaviors, such as the timing of text entry sessions or the placement of windows/keyboards, is required. By analyzing these video recordings, the attacker can infer when the user is typing and recover the typed text content. We define two main attack scenarios under this model:

(1) Streamers who publicly share their Persona views during a live stream, exposing their eye gaze data to potential attackers.
(2) Eye gaze data leaked via video conference apps, such as Zoom and Microsoft Teams, if a user shares the Persona virtual camera view with an attacker during a meeting or a call.

The complexity, stealthiness, and frequency of the attack differ based on the attack vectors. For streamers, no prerequisites are required for attackers, making it the most straightforward scenario. However, the potential victims are limited to those who stream their

VR activities. In the case of attacks via conference apps, the complexity and prevalence depend on the nature of the online meetings. Highly confidential meetings may present more obstacles, while public webinars, open to all attendees, may be easier to exploit.

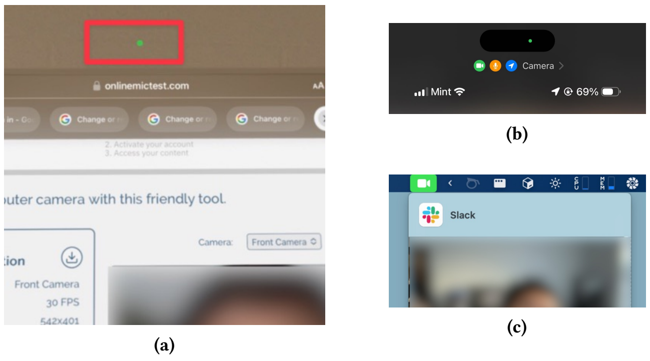


Figure 2: Camera access indicators on different platforms (a) Apple Vision Pro's subtle green dot that can be covered by any other window. (b) iPhone's camera access indicator. (c) MacBook's system menu bar camera indicator.

Moreover, GAZEPLOIT can be exceptionally stealthy, with users potentially unaware their Persona is being recorded. Unlike laptops or smartphones that alert users with a hardware LED or system notification, the Apple Vision Pro only uses a small green dot to indicate virtual camera usage, as shown in Figure 2. This indicator can easily be ***obscured*** if another app window overlays the Persona recording window, hiding the visual cue. The attacker's view remains unaffected. Additionally, the green dot is very small compared to the large VR/MR workspace, making it easy to miss. This subtle indicator, combined with the unique spatial configurations in a 3D VR workspace, enhances the attack's stealthiness, making it less noticeable than on traditional 2D devices.

3.2 Attack Vectors on Various Typing Scenarios

Building upon the attack scenarios discussed earlier, we further divide the attack into four distinct vectors. These vectors are based on different typing scenarios in VR/MR environments. Each scenario has unique characteristics, such as keyboard size and typing patterns, which influence the attacker's strategy and the attack's effectiveness.

Passcode (PIN) Inference: The passcode (PIN) keyboard, as shown in Figure 3a, is exclusively numeric. While Apple Vision Pro employs Optic ID [18] as an alternative to passcodes, this keyboard remains integral for tasks such as device unlocking, phone number entry, PIN input in financial apps, and two-factor authentication. The compact design of this keyboard results in gaze angles focused within a narrow area, providing a distinct pattern for the attacker to exploit. Additionally, the typical input lengths—6 digits for PINs and passcodes, 10 for phone numbers—give further clues about the nature of the data being entered.

Password Inference: The entry of passwords is unique in that it often requires toggling between numeric and alphabetic segments of the keyboard. This may involve using a "`shift`" or "`num`" key to access special characters and numbers. Additionally, passwords usually do not include spaces. These features can serve as key indicators to identify password entry sessions, distinguishing them from other types of text entry. The keyboards used for password entry are shown in Figure 3b and Figure 3c.

URL and Email Address Input: The typing of URLs or email addresses presents unique patterns that can be exploited for inference attacks. These inputs often contain specific sequences or symbols, such as "`.com`", "`.edu`", "`.net`", or "`@`", which can signal the completion of a web address or email address. The keyboard used for these inputs remains the QWERTY layout (shown in Figure 3b), but with the space key replaced by space, "`@`", and "`.`" keys.

Word Inference for Message Recovery: In scenarios where users draft messages or articles outside of web browsers, they often switch between the keyboards displayed in Figure 3b and Figure 3c. These keyboards, in comparison to the passcode keyboard, cover a broader range of gaze regions. Furthermore, the presence of multiple spaces between words, which are absent in password, URL, or email address inputs, can serve as an identifiable pattern.

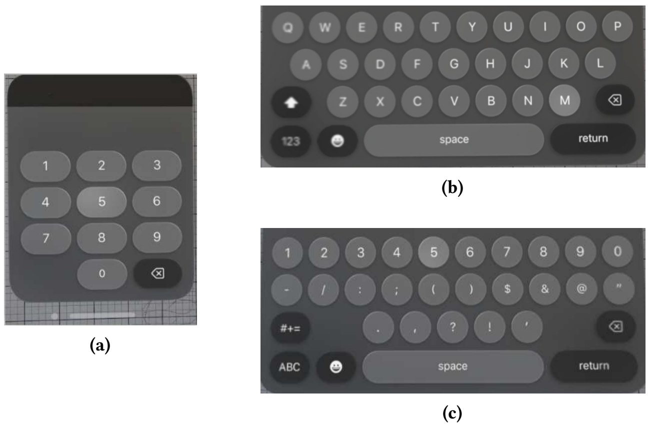


Figure 3: Keyboards used on Apple Vision Pro (a) Passcode(PIN) keyboard (b) Default QWERTY keyboard (c) Number and special character keyboard

3.3 Limitations on Attack Scenarios

Our attack can be executed as long as several conditions are met by the VR platform: (1) the VR model is equipped with an eye tracker, (2) the VR device can create sharable virtual avatars displaying users' eye movements, and (3) the VR device supports a gaze-controlled keystroke input method.

While many VR devices have already met the first two requirements, few of them met requirement (3). Currently, only the Apple Vision Pro satisfies all three requirements, which limits the applicability of GAZEPLOIT. This limitation is not due to any shortcomings of GAZEPLOIT but rather the limited adoption of gaze-typing technology in VR devices currently available in marketplace.

However, gaze-typing is an emerging technology that is expected to become more popular in consumer devices. Companies like Meta and Microsoft are also exploring the potential of integrating gaze-typing [50]. Additionally, Apple has announced that gaze-typing will be implemented in iOS 18 (beta) [1] to assist users with disabilities. As this technology becomes more prevalent, a broader range of devices will be susceptible to GAZEPLOIT.

4 Preliminary Attack Analysis

This section evaluates the risk and explores various attack techniques, emphasizing the realistic possibility of executing keystroke inference attacks. We have developed the GAZEPLOIT attack to tackle these challenges and include a demonstration of its initial effectiveness. The attack process is detailed in the following subsections. Section 4.1 outlines our methods for extracting biometrics such as eye aspect ratios (EARs) and gaze direction from video frames. In Section 4.2, we discuss our technique for identifying typing sessions using these metrics and describe our approach to determine the timing of keystrokes. Section 4.3 explains how we project gaze directions onto a virtual keyboard plane to identify pressed keys, while Section 4.4 introduces a probability calculation technique for inferring likely keystrokes. Finally, Section 4.5 details a unique mapping strategy for passcode inference.

In this section, we utilize the data that was collected on us and perform study for experiment purpose. Specifically, we recorded demo videos while the authors wearing and using Apple Vision Pro in a normal campus scenario. The videos includes several scenarios such as browsing a webpage, typing in Slack, attending Zoom meeting, unlock devices with passcodes, etc. We further randomly choose a demo video with the duration of 120 seconds among them to analyze the keystroke inference attack.

To begin with, we find successfully inferring keystrokes based on very limited information of a video that only showing the users' virtual face can be difficult. Specifically, we identify the following three main challenges:

Main Challenges

Challenge 1: How to extract the essential biometrics?

Challenge 2: When do users type?

Challenge 3: Where is the virtual keyboard and which key is being pressed?

4.1 Eye-related Biometrics Extraction

To address **Challenge 1**, we need to find suitable eye-related biometrics besides gaze estimation. Neurologists already confirm that eye blink rate decreases with increasing attentional demand [30]. Given typing in VR is typically an activity requiring high attention level, we expect a lower eye blink rate during typing sessions. So we choose Eye Aspect Ratio(EAR) as the biometric to measure eye blink rate. And thus our GAZEPLOIT attack relies on two biometrics, EAR and eye gaze estimation, as shown in Figure 4.

Eye Aspect Ratio (EAR) To quantify this, we use the 68-point face landmark model to calculate. The EAR, calculated as the eye width, which is the Euclidean distance between the eye corners, divided by the eye height, the distance between the midpoints of the upper and lower eyelids, serves as a metric for blink detection. A eye blink event is identified when this ratio reach a local maximum as shown in Figure 6.

Gaze estimation Our gaze estimation model employs a pre-trained Resnet18 model [17], which takes video frames with human faces as input and output the yaw and pich angle values of eye gaze. The model is trained on the ETH-XGaze dataset [46]. This dataset features a wide range of head poses with yaw and pitch angles up to $\pm 80^\circ$,$\pm 80^\circ$ and gaze yaw and pitch angles reaching $\pm 120^\circ$,$\pm 70^\circ$ respectively. This range is broader compared to other well-known datasets such as MPIIGaze [48] or RT-GENE [14], making it particularly suitable for our needs. This is because in a virtual reality setting, the virtual camera's position corresponds to the app's window, which can be located anywhere around the user, not just directly in front. This flexibility ensures that our model accurately captures gaze direction under various user orientations relative to the camera. We also evaluate how robust the GAZEPLOIT attack is regarding the position of the Persona virtual camera in Section 6.4.

4.2 Typing Activity Identification

We divide **Challenge 2** into two smaller problems: (1) how to identify typing sessions, in which users type a string of characters consecutively, and (2) how to identify the timings of individual keystrokes in these typing sessions. To address these problems, we need to extract higher-level features from the selected two biometrics, gaze estimation and EAR. Specifically, we utilize an RNN for identifying typing sessions and design filters to extract fixations from gaze estimation. Individual keystrokes are very likely happening on these fixations lasting sufficient time.

Identify Typing Sessions The gaze estimation depicted in Figure 6 reveals a noteworthy pattern: during consecutive keystrokes on a virtual keyboard, the direction of eye gaze tends to be more concentrated and exhibits a periodic pattern, in contrast to the gaze patterns during other activities, which appear more erratic and random. Further analysis indicated that the frequency of eye blinking decreases during typing sessions compared to other regular activities due to high attention demands. During typing, eye blinks were significantly rarer. These observations prompted us to consider Recurrent Neural Networks, RNNs, as a suitable method for distinguishing typing actions from other activities. RNNs excel at recognizing patterns in sequential data, such as the traces of eye gaze direction, making them exceptionally well-suited for tasks where the context or sequence of inputs is crucial [41].

We define a typing session as the period between the saccade of the first keystroke and the saccade of the last keystroke, as depicted in Figure 6. We labeled the typing sessions manually and later use the labeled period in training and validation stages. We utilized a Bidirectional RNN model with its architecture shown in Figure 5, specifically designed to classify typing events from other VR usage scenarios. This model processes inputs representing the yaw and pitch angles of eye gaze and eye aspect ratio. The model has a hidden layer size of 128, and differentiates between two classes: typing as label 1 or other activities as label 0. The RNN is using cross entropy as the loss function and the Adam optimizer to update its weights. This output will segment the raw gaze trace into several clips labeled with the two classes. The model was trained using 18

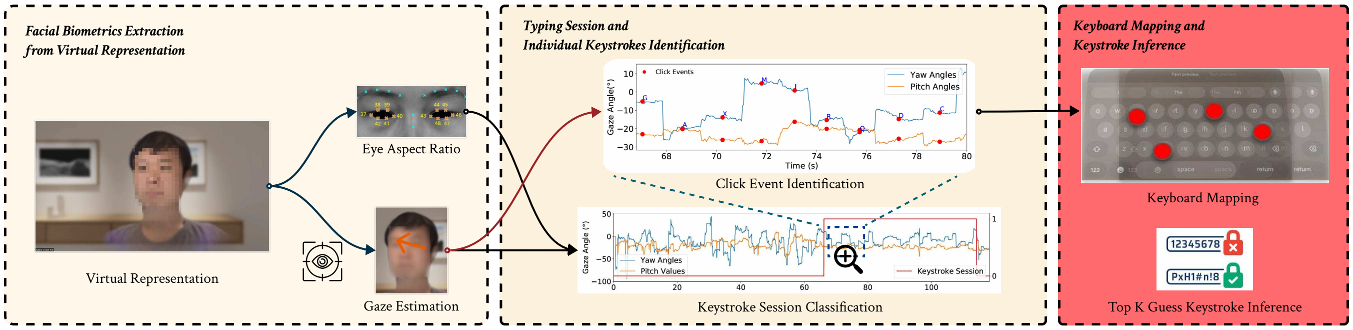

Facial Biometrics Extraction from Virtual Representation
Virtual Representation
Eye Aspect Ratio
Gaze Estimation
Typing Session and Individual Keystrokes Identification
Click Event Identification
Keystroke Session Classification
Keyboard Mapping and Keystroke Inference
Keyboard Mapping
Top K Guess Keystroke Inference

Figure 4: GAZEPLOIT Attack Overview: The attacker starts by extracting facial biometrics from the victim's virtual representation, focusing on gaze estimation and eye aspect ratios (EARs). These biometrics are then used to distinguish typing sessions from other activities and to identify the timing of each keystroke. Finally, the attacker maps the gaze vectors to a virtual keyboard to infer the pressed keys.

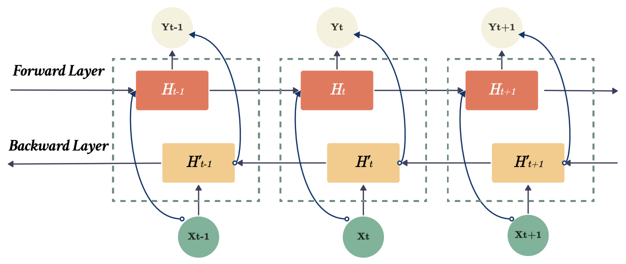


Figure 5: Bidirectional RNN Architecture for classifying typing sessions

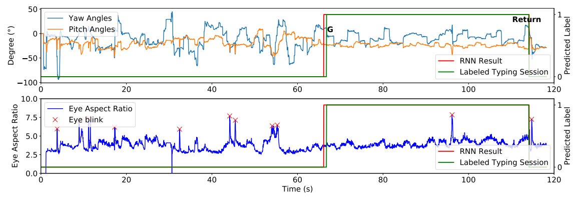


Figure 6: Eye gaze estimation and Eye Aspect Ratio of user a) browsing a webpage b) typing in iMessage, beginning with 'G' and ending with 'Return' c) sending the message and closing the apps

videos of the collected 23 videos in the comprehensive dataset. All the typing sessions of the videos are labeled during the collection as described in Section 5. The accuracy of this model is detailed in Section 6.1, and its application to the demonstration video is illustrated in Figure 6. In the demo attack, the user first browses a webpage for a minute and then switches to iMessage from 0s to 66.8s. Then the user inputs 35 letters from 66.8s to 114.1s, followed by sending the message and closing the app. It can be observed that during the predicted typing session, the yaw and pitch angles are within a range of (40°, 20°) while in other activities the yaw and pitch surpass (100°, 50°). There is also a periodic pattern of gaze staying at each fixation for approximately 1s. Besides, the eye blink rate is much lower when typing, occurring only 1 time in 40 seconds, whereas typically there is about one blink every 7 seconds during other activities. These observations are crucial features for identifying typing sessions.

Identify Individual Keystrokes After typing session identified, the next step is to recognize individual keystrokes. Clicks during gaze typing are confirmed by a snipping hand gesture. While this gesture can be captured by the headset's external camera and displayed in the Persona avatar view, it is not always visible because the Persona view typically shows the user from the chest up, and users often rest their hands out of view. Therefore, we cannot rely on hand gestures to identify keystrokes. Instead, we use consistently visible features: saccades and fixations [5]. During gaze typing, users' gazes shift between keys and fixate on the key to be clicked, resulting in saccades followed by fixations. We developed an algorithm to detect these patterns in the gaze traces.

First, we calculate gaze stability S, the average cosine similarity between gaze directions within a time window of length N, as shown in Equation 1. In the equation, $\alpha_{i,j}$ denotes the angle between two gaze directions (ϕ_i,θ_i) and (ϕ_j,θ_j). As illustrated in Figure 8, saccades induce lower stability and fixations induce higher stability.

$$S(n) = \frac{1}{N^2} \sum_{i=n}^{n+N-1} \sum_{j=n}^{n+N-1} \cos(\alpha_{i,j}) \tag{1}$$
$$\cos(\alpha_{i,j}) = \cos(\phi_i)\cos(\phi_j)\cos(\theta_i - \theta_j) + \sin(\phi_i)\sin(\phi_j)$$

Saccades and fixations can be distinguished by analyzing dips in the stability trace. Saccades cause large dips, while noise during fixations results in small dips. To differentiate between them, we need a threshold S_T that captures most saccade-induced dips while excluding noise-induced ones. We observed that saccades are often followed by minor gaze adjustments that create intermediate dips, which are smaller than saccade-induced dips but larger than noise-induced dips. An effective strategy is to select S_T within the range of these intermediate dips, as it includes most saccade-induced dips while rejecting noise-induced ones. Although this threshold captures some intermediate dips from post-saccade adjustments, they can be filtered out based on their timing, as they always closely follow deeper saccade-induced dips.

To estimate the optimal S_T, we use function `findpeaks(−S)` in MATLAB, setting the `MinPeakHeight` parameter to `median(−S)` to identify significant dips in the trace. We then analyze each trace to find all intermediate dips, defined as those occurring within 500 ms of a deeper dip, as shown in Figure 8. We identify all dip depth ranges that consist entirely of intermediate dips and select the range with the highest number of intermediate dips. This range is considered the optimal threshold range, and we record the depths of all dips within it. We repeat this process for all data, recording dip depths in the optimal threshold range for all traces. We then plot the distribution of all recorded dip depths in Figure 7. The results show that selecting 0.9996 as S_T is most optimal since it aligns with the peak of the overall intermediate dip depth distribution.

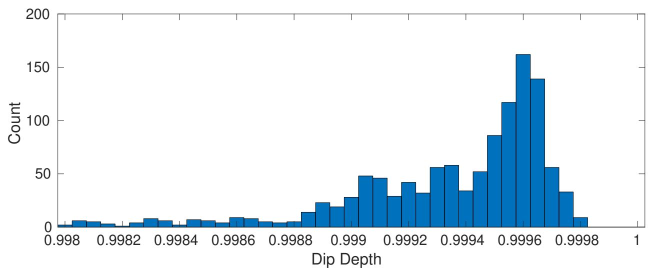


Figure 7: Averaged Optimal Threshold Distribution

After determining S_T, we apply it to the gaze traces to select dips deeper than S_T as saccade candidates. We then remove intermediate dips that occur within 500 ms after deeper dips, as these are typically caused by minor adjustments. Finally, the intervals between the preserved saccades are identified as fixations, each corresponding to a single keystroke click. We show part of the identified clicks in the demo attack in Figure 8. As illustrated in the figure, blue labeled letters indicate the correctly identified clicks. The overall precision and recall of the click identification on the collected dataset are presented in Section 6.1.

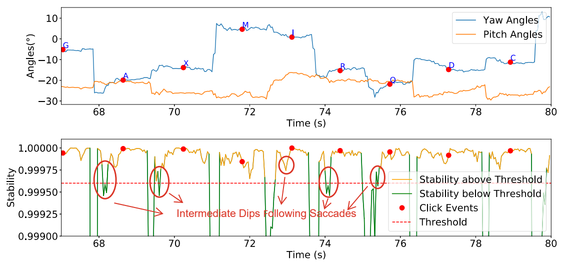


Figure 8: Identify individual keystrokes in the keystroke session. The red dashed line is the S_T to distinguish saccades (green traces) and fixations(yellow traces).

In our demo attack, there were 35 actual keystrokes. The attack model identified 36 keystrokes in the trace, comprising 35 true positives and 1 false positive, with no false negatives. This resulted in a precision of 97.2% and a recall of 100% for the demo attack.

4.3 Adaptive Virtual Keyboard Layout Mapping

To accurately map gaze points to specific keys during individual keystrokes, it is essential to precisely determine the location of the virtual keyboard in virtual space. Typically, the keyboard is a rectangular area on a plane, with a line connecting its center to the user acting as the plane's normal vector, as shown in Figure 9. Users are free to adjust the keyboard's size, direction, and distance within the virtual space, causing its position to vary widely on a sphere. While these specific adjustments are not directly observable to an attacker, this section explains how we can utilize eye-movement statistics to accurately estimate the keyboard's location.

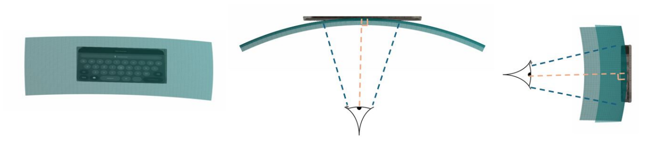


Figure 9: Virtual Keyboard is on a sphere facing the center at the user's eyes. The normal vector of the keyboard points to the midpoint of the user's eyes.

Keyboard Plane Estimation We first estimates the plane where the virtual keyboard sits in the virtual space. Because the line connecting the user to the keyboard center is perpendicular to this plane, the average gaze direction can estimate the plane's normal vector. Subsequently, the gaze directions are rotated into a new coordinate system where the estimated normal vector of the keyboard plane aligns with the x-axis. The rotated gaze directions described by yaw angle θ and pitch angle ϕ are then transformed into gaze directions represented by the unit vector $\mathbf{u}$ in the XYZ coordinate using Equation 2.

$$\hat{\mathbf{u}} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \begin{bmatrix} \cos(\theta)\cos(\phi) \\ \sin(\theta)\cos(\phi) \\ \sin(\phi) \end{bmatrix} \tag{2}$$

The keyboard plane is parallel to the y-z plane. The gaze direction can be projected to the gaze points on the keyboard plane using equation 3, with p_x being the distance between the keyboard and the user. $\mathbf{p}$ represents the gaze point coordinates on the keyboard plane in the y-z space.

$$\mathbf{p} = \begin{bmatrix} p_y \\ p_z \end{bmatrix} = \begin{bmatrix} u_y \\ u_z \end{bmatrix} \times \frac{p_x}{u_x} \tag{3}$$

Keyboard Area Estimation After determining the keyboard plane, we must also identify the specific area within that plane where the keyboard is located. The keyboard typically has a much wider horizontal span than vertical, leading to a predominance of horizontal gaze direction movements during typing. Consequently, the orientation of the keyboard can be estimated by analyzing the distribution of gaze points. Using Principal Component Analysis (PCA), we can identify the principal directions of variance in the dataset, which corresponds to the horizontal direction of the keyboard.

This involves computing the eigenvectors and eigenvalues of the dataset's covariance matrix. First, we compute the covariance matrix **C** of the dataset using Equation 4

$$C = \frac{1}{n-1}(\mathbf{P}-\overline{\mathbf{P}})^T(\mathbf{P}-\overline{\mathbf{P}}) \tag{4}$$

where **P** represents the matrix of all data points $\mathbf{p}_i$ and $\overline{\mathbf{P}}$ is the mean vector of these points. Eigenvectors $\mathbf{v}_i$ and their corresponding eigenvalues λ_i are computed using Equation 5

$$C\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{5}$$

The eigenvector with the highest eigenvalue indicates the direction of the greatest variance, i.e., the horizontal direction. To reorient the original dataset into principal components, the data points are projected onto these eigenvectors using Equation 6

$$\mathbf{p}' = \mathbf{V}^T\mathbf{p} \tag{6}$$

where **V** comprises the columns formed by the eigenvectors $\mathbf{v}_i$ of **C**. This transforms the gaze points to $\mathbf{p}'$ so that the transformed axes align with the horizontal and vertical directions of the keyboard.

After identifying the keyboard orientation, the keyboard boundary can be estimated using the distributions of the gaze points. We hypothesize that the gaze points extend linearly across the maximum possible area of the keyboard, spanning horizontally from the letter Q to the letter P, and vertically from the letter Y to the Space key. Initially, this assumption may seem questionable, as users might typically focus on a limited section of the keyboard. However, upon evaluating the four typing scenarios, this approach is justified by very high key coverage on the edge of the keyboard. For instance, in message inference scenarios, the Space key is frequently used. The message is concluded with a press of the return key. Further validation comes from analyzing two Wikipedia pages. Out of 676 sentences longer than 15 characters, only 8 lacked the leftmost letters "q", "a", or "z". In password inference scenarios, presses of the return, numberspace, and number keys must be included as modern passwords require a combination of letters and numbers. For URL inference, the "w" letter and "." and the return key are usually included. All these keys are edge-located keys that guarantee the typing area is the whole keyboard instead of a limited region. As shown in Figure 10, with more than 4 edge-located keystrokes spanning both horizontally and vertically, we can locate the keyboard within the error of the distance of two adjacent keys.

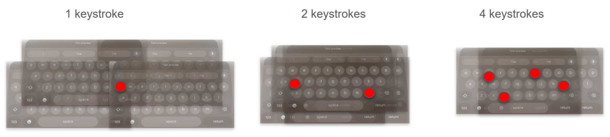
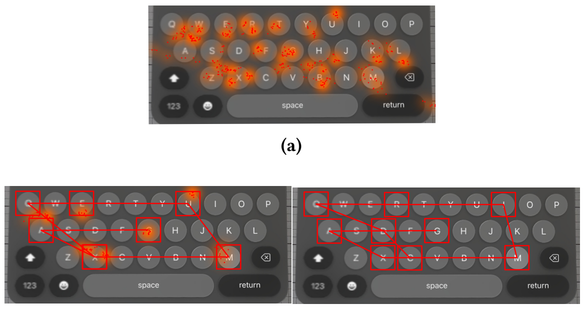


Figure 10: Virtual Keyboard possible locations decrease with more edge-located keystroke angle information.

4.4 Top K Keystroke Candidate Accuracy

After locating the keyboard, the gaze points of the individual keystrokes are mapped to the keyboard layout as shown in Figure 11. We assume a 2D Gaussian probability distribution centered at the gaze direction for each frame in a fixation of one individual keystroke, the distribution has a 2σ value of the radius of one key. The probability of each key is averaged across every frame in each candidate keystroke period as depicted in Equation 7. In the equation, $P(C_k)$ denotes the probability of click event C inferred as key k, N_C denotes frame numbers in the fixation duration of one individual keystroke, $\mathcal{N}_i$ denotes the 2D Gaussian distribution of the gaze in the ith frame, S_k denotes the 2D region of key k. p_{y_i}, p_{z_i} are the horizontal and vertical coordinates of the gaze direction mapping in the ith frame, $\sigma_{y_i}, \sigma_{z_i}$ are horizontal and vertical standard deviations, which are fixed values of half the radius of a letter key.

$$P(C_k) = \sum_{i}^{i\in N_C} \frac{\iint_{S_k} \mathcal{N}_i((p_{y_i}, p_{z_i}), (\sigma_{y_i}, \sigma_{z_i}))}{N_C} \tag{7}$$



(a)

(b) Predicted (c) True

Figure 11: Mapped gaze directions for keystroke inference of the demo attack (a) Adaptive virtual keyboard mapping (b) Predicted first guess keystrokes from 67-80s (c)Actual keystrokes (True Labels) from 67-80s

We then look at the top K keystroke inference guesses that have the top K highest probability, and infer the keystroke based on these candidates as shown in Figure 12. The red letter in blue 5 letters is the correct label of the keystroke. In this demo attack, When K is set to be 5, the character inference accuracy of the attack reaches 100%.

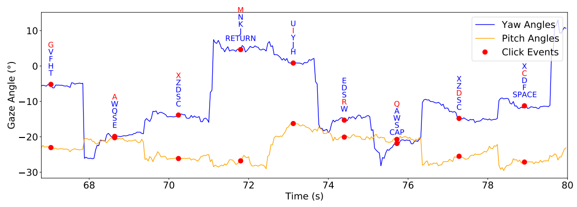


Figure 12: Top 5 guesses of keystroke letters listed from top to bottom of 67-80s in the keystroke session in the demo attack, red letters represent the actual keystroke labels

4.5 Passcode Inference

The passcode inference attack is executed by identifying the key patterns that best match the observed gaze point patterns. This involves calculating a translation vector **t** that minimizes the distances between the gaze points and their nearest corresponding keys. The process is mathematically represented in Equation 8, where $\mathbf{p_i}$ represents the coordinates of the ith gaze point and $\mathbf{k_j}$ represents the coordinates of the jth key on the keyboard.

$$\mathbf{t} = \min_{\mathbf{t}} \sum_{i=1}^{N} \min_{j} \|(\mathbf{p_i} + \mathbf{t}) - \mathbf{k_j}\|^2 \quad (8)$$

After applying the calculated translation vector to the gaze points, the probability of each key being typed can be calculated using Equation 7, and the pressed key can be inferred.

Two attack examples are shown in Figure 13. A potential issue arises when multiple passcodes share the same pattern. If the pattern is unique, as in Figure 13 (a) and (b), the correct passcode might be guessed on the first attempt. However, if the same pattern corresponds to multiple passcodes, additional guesses may be necessary, as shown in Figure 13 (c) and (d). Nonetheless, this issue only slightly increases the passcode search space.

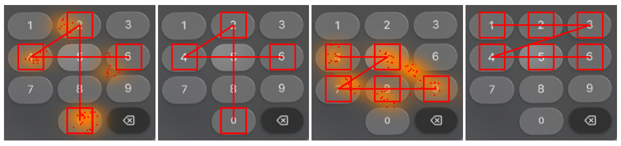


Figure 13: Mapped gaze points and the first guess inferred passcode for keystroke inference on the PIN keyboard, (a) can be correctly inferred within one guess to match (b), (c) can be correctly inferred within four guesses to match (d).

5 VR Gaze-typing Data Collection

5.1 Comprehensive Dataset

We began data collection only after receiving approval from the Institutional Review Board (IRB) to further validate our attack designs in previous sections. Our study included 23 participants representing diverse racial, gender, and cultural backgrounds. Prior to data collection, participants were provided with an information form detailing the tasks involved, potential risks, and measures taken to ensure data confidentiality, as per IRB guidelines. Participants were required to sign this form before proceeding.

During the initial phase, all participants underwent eye-tracking feature recalibration on the Apple Vision Pro device and updated their Persona view. Subsequently, they joined a Zoom meeting from their VR headset, sharing their Persona view in Zoom to commence recording. The duration of the recorded meetings averaged 16 minutes and 10 seconds per participant, constituting the raw data.

Participants were instructed to complete seven tasks during the recorded meeting, presented in a randomized order to compose a comprehensive dataset: (1) Open a web browser, navigate to `www.google.com`, enter the Gmail address and password to sign in to a test Google account, perform a Google search of arbitrarily chosen queries, and browse the search results. Then, navigate to another commonly used website. (2) Watch a 3-4 minute video on Apple TV. (3) Open Slack and send several messages. (4) Engage in conversation with Zoom participants who have their cameras on. (5) Engage in conversation with individuals in the physical world, situated close to the participants. (6) Play Super Fruit Ninja. (7) Change the passcode.

For all text entry tasks, participants are instructed to indicate the beginning and end of their typing sessions, which serves as a reference for labeling the corresponding video segments during the typing session classification phase. To label the typing sessions, we manually check the gaze estimation traces around the reference timings of all participants to find the saccade pairs of the first and last keystrokes of each typing session. The time intervals between these saccade pairs are labeled as typing sessions (Label 1). All other time intervals are labeled as other activities (Label 0). Within each typing session, we further label the keystrokes with the correct keys being clicked. This detailed labeling is used to evaluate the accuracy of keystroke inference. In the comprehensive dataset, the typing session duration averaged 2 minutes and 25 seconds per participant.

During text entry tasks in the web browser, participants navigate to "`www.google.com`" and optionally another commonly used website of their choice. They use a Gmail account with the email address to be either "participantvisionpro@gmail.com" or "visionproparticipant@gmail.com". Participants search for either a predetermined phrase ("`VR headset`") or a phrase of their choosing, which we subsequently verify through Google search history as the correct label.

For message inference, participants are instructed to send a message to another Slack account. This includes both a predetermined message ("`A quick brown fox jumps over the lazy dog`") and a message of their own composition. These messages sent to our Slack account serve as labels for correct keystrokes.

To facilitate password inference, participants select a random password in adherence to Google account password requirements: (1) a minimum of 8 characters, (2) a combination of letters, numbers, and/or symbols, (3) no leading or trailing blank spaces, and (4) not easily guessable (e.g., "`password123`"). The chosen password is then temporarily set as the password for a test Google account. During the data collection process, participants use this password to sign in, allowing us to capture their typing behavior accurately. Regarding Passcode (PIN) inference, participants select a 6-digit PIN code prior to the experiment, enabling us to identify the correct keystrokes during data analysis. To change the Passcode, each participant is required to input the original Passcode once and input the new Passcode twice to confirm the change. 3 participants did an extra round of Passcode input due to misclicks. All video recordings are conducted in five distinct indoor scenarios over a period exceeding four weeks.

In the dataset, 23 participants input 24 unique PINs, totaling 72 6-digit PIN entries. For password inference, there are 23 different password strings with an average length of 12.43 characters. In message inference, aside from the fixed sentence, the self-chosen sentences have an average length of 20.52 characters. The average length of the 40 collected email/URL strings is 19.8 characters. Participants are allowed to use "`BACKSPACE`" to correct the typos made in the dataset. There are 84 "`BACKSPACE`" keystrokes in the dataset.

IRB approval and Responsible Disclosure Our data collection process has been approved by the university's Institutional Review Board (IRB). We have reported these vulnerabilities to Apple. Apple Security Team has reproduced GAZEPLOIT using our artifacts and assigned CVE-2024-40865. Patches are deployed on visionOS 1.3.

5.2 Sentence Typing Dataset

We collect an additional dataset specifically for sentence typing inference evaluations. This dataset involves 15 participants, 7 of whom are not included in the initial group of 23, bringing the total number of participants to 30. The experimental design builds upon the sentence typing experiment used in Typose [42], but with significant improvements. Typose required participants to input 5-word "sentences" synthesized by random permutations from a pool of 60 selected words. In contrast, our experiment asks each participant to type 15 different 7-word real sentences generated by ChatGPT. A total of 225 unique sentences are created, using words from the 10,000 most frequent English words found in the OpenSubtitles Dataset GitHub Repository [6], without any other restrictions on word selection.

In this experiment, video recordings averaging 21 minutes and 30 seconds in length are captured for each participant, with the typing sessions averaging 13 minutes and 15 seconds in length. The dataset consists of 10329 keystrokes, including 1351 "SPACE" keys, 359 "BACKSPACE" keys, and 8619 English letters. In total, it contains 1,575 words, of which 594 are unique. Figure 14 shows the distribution of word lengths in the typed sentences, which range from 1 to 13 characters. Given that only 14 out of 10,000 words from the OpenSubtitles Dataset exceed 13 characters, our dataset is highly representative of the English vocabulary.

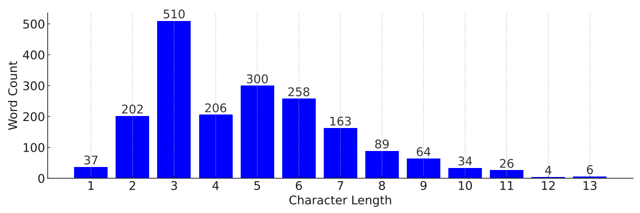


Figure 14: Word Counts of different word length

Compared to Typose [42], our dataset better represents real-world attack scenarios in the following ways:

More unique words: Our dataset contains 594 unique words, significantly more than the 60 used in Typose.

No word length constraints: Our dataset includes words ranging from 1 to 13 characters in length, whereas Typose uses fixed-length words of 2 or 6 characters.

More realistic sentences: We use meaningful, grammatically correct sentences to better reflect real-world attack scenarios compared to the simple random word permutations used in Typose.

Allow typo corrections: We allow participants to make and correct typos using the "BACKSPACE" key to reflect real-world typing behavior, whereas in Typose, words with typographical errors are considered unclassifiable.

6 Evaluation

The evaluation consists of two stages. The first stage assesses the system's ability to detect when typing activity occurs—referred to as typing sessions and individual keystroke identification, which addresses **Challenge 2**. The second stage evaluates the accuracy of mapping the detected gaze to specific keys on a keyboard, thus determining which keys were pressed—termed keystroke inference accuracy addressing **Challenge 3**.

6.1 Typing Activity Identification

We use precision, recall, and accuracy in Equation 9 as metrics for typing session and individual keystroke identification results.

$$
\begin{aligned}
Preceision &= \frac{TP}{TP + FP}, \; Recall = \frac{TP}{TP + FN} \\
Accuracy &= \frac{TP + TN}{TP + FP + TN + FN}
\end{aligned}
\tag{9}
$$

Typing Session identification results We assume the attacker can first collect labeled data from non-victims and train an RNN model for typing session identification in the attack. We use the data with labels of typing sessions from some participants in the comprehensive dataset for training the RNN model and validate it on both the remaining participants' data in the comprehensive dataset and also all data from sentence typing dataset. And we iterate through all the 23 participants and train and validate 23 rounds, following the K-fold cross-validation style. Each round is trained with a batch number of 64 and an epoch number of 100. We change the participants' number in the training dataset from 1 to 22 and plot the average, upper, and lower bound of precision, recall, and accuracy in Figure 15. The accuracy, precision, and recall will rise and stabilize around 98.1%, 90.5%, and 97.2% when the training dataset size exceeds 18 participants. This high accuracy result validates our theory on combining both gaze vectors and eye aspect ratio to predict gaze typing sessions in VR.

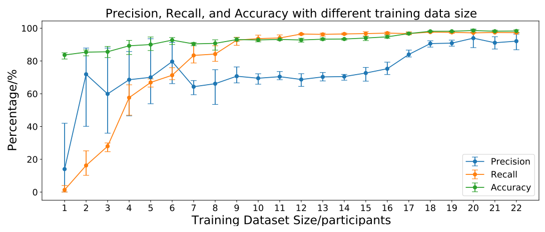


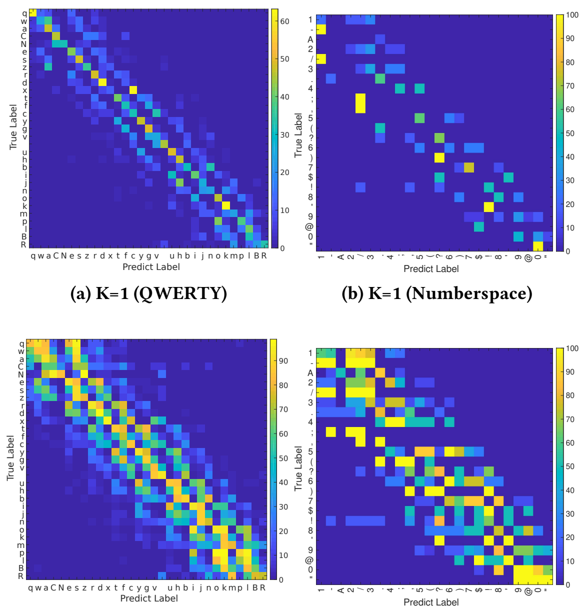
Figure 15: Accuracy, Precision, Recall vs. Participants as training dataset

Individual Keystrokes Identification Results We evaluate the precision and recall of the individual keystrokes identification method. We have 12839 actual clicks in identified typing sessions, 12428 of which were successfully identified as true positives, GAZEPLOIT also infers 2039 false positives where there are no clicks but identified as a click. Overall the average precision rate and recall rate are 85.9% and 96.8%. There is a trade-off between precision and recall rate because we use a threshold of gaze stability to identify keystrokes. A less sensitive threshold value can lead to a lower recall rate and higher precision rate and thus be beneficial in certain attack scenarios.

6.2 Keystroke Inference Accuracy

In this section, we show the confusion matrices of keystroke inference on all characters on both the QWERTY keyboard and the numbers and special characters keyboard in Figure 16. From the Figure, we can clearly find the pattern that keys that are close to one another has a higher chance of being falsely inferred. Also, people tend to include fewer numbers and special characters. In our dataset, the number of keystrokes on the QWERTY keyboard is 100 times the number of keystrokes on the special characters and numbers keyboard because numbers and special characters appear more in password entry scenarios. Consequently, the confusion matrices for the numbers and special characters keyboard appear less smooth compared to those for the QWERTY keyboard.

Next, we show the accuracy of inferring the keystroke input of 23 participants in different typing scenarios in Table 1 including 1. message sent in social media software 2. password 3. URL and email address 4. Passcode(PIN).



(a) K=1 (QWERTY) (b) K=1 (Numberspace)

(c) K=5 (QWERTY) (d) K=5 (Numberspace)

Figure 16: confusion matrix of QWERTY / numberspace keyboard for each character, top 1 and top 5 guesses accuracy

Password Inference As stated in Section 3.2, GAZEploit attack model recognize keystrokes with letters, and numbers but without space keys as potential passwords. Our GAZEploit model could achieve 34.6% character inference accuracy in the first guess and 77.0% character inference accuracy in the top five guesses.

According to the accuracy of correctly inferred characters, we can reduce the password search space of a 8 to 12 character random password containing letters and numbers from 2.2×10^{14}–3.2×10^{21} to 3.9×10^{5}–2.4×10^{8}. This search space decrease makes a brute force attack possible in seconds or hours even using an out-of-date CPU model like Pentium 100 [35].

URL and Email address As mentioned in Section 3.2, our attack recognize patterns including "`@`", "`com`", "`.edu`", "`.us`" as potential URL or email address. Our model could achieve 35.3% character inference accuracy in the first guess and 86.1% character inference accuracy in the top five guesses.

Passcode Inference As introduced in Section 3.2, GAZEploit attack model recognize keystrokes restricted in a smaller typing region as potential passcodes. We evaluate the key pattern prediction accuracy. The first guess will infer 59.1% keys correctly. And with the top 5 guesses, the accuracy reached 73.0% as shown in Table 1. We also count how many attempts we need to recover the 6-digit passcodes. We could infer 8.3% of all 6-digit passcode strings in the first attempt, 16.6% in 4 attempts, and 25.0% in 32 attempts. This success rate guarantees the attack model could unlock the device or hack financial apps within hours, which poses a great security risk.

Table 1: Top K character prediction accuracy (%) in four typing scenarios

K	Message	Password	URL/Email	Passcode
1	38.7	34.6	35.3	59.1
2	64.4	50.8	57.1	64.2
3	77.4	61.8	70.2	69.6
4	86.2	71.0	81.0	70.5
5	92.1	77.0	86.1	73.0

6.3 Message Inference

The message inference attack is evaluated on both the messages typed in the comprehensive dataset and the additional sentence typing dataset. The message inference accuracy is evaluated based on three metrics: 1. character inference accuracy, 2. word segmentation accuracy, 3. word inference accuracy. we can tell from Table 1 that the top 5 character prediction accuracy is over 92%, and top 6 accuracy is 94%, which does not increase significantly. Thus, we infer word segmentation and words based on the top 5 character prediction matrices.

Sentence Segmentation An example of segmenting a sentence is shown in Figure 17. To segment sentences into words, we first identify all potential segmentation points where the "`SPACE`" key appears in the top 5 predictions. Treating all these keys as "`SPACE`" may result in false positives, as many are actually keys near "`SPACE`". To find correct segmentations, we iterate over possible segmentations and use a dictionary to verify potential words. We consider only segmentations that produce valid words to derive sentence candidates. If there are no valid segmentations for more than 15 consecutive keystrokes (the largest word length in the dictionary), it indicates a true character is not within the top 5 predictions. In such cases, we skip to the next potential segment point and mark the skipped segment as unknown.

Using this method, we can significantly reduce the false positive and accurately identify "`SPACE`" keys to correctly segment sentences.

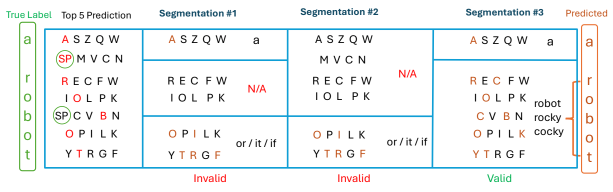


Figure 17: Segmentation Candidates Selection

We show the confusion matrix of "SPACE" key prediction in the sentence inference experiment in Table 2.

Table 2: Confusion Matrix of SPACE key prediction as word segmentation in Message Inference

Keystrokes	SPACE	Characters
SPACE (Predicted)	1457	242
Characters (Predicted)	153	9608

Typo Correction Inference During data collection, participants are allowed to make typos and correct them using the "BACKSPACE" key. Such typo correction behaviors can also be identified by our method. When a "BACKSPACE" key is in the top 5 predictions, we consider it as a potential typo correction. If removing this key along with the key preceding it results in a valid word in the dictionary, we consider the user to have made a typo correction. Otherwise, we ignore the "BACKSPACE" key and infer the words with the remaining letters in the top 5 predictions. This method successfully corrected 312 out of 398 typos, with no false positives.

Dictionary-based Word Inference We use the 10,000 most frequent words in English from the OpenSubtitles Dataset [6] for word prediction. GAZEPLOIT attempts to infer possible words in each word segmentation based on the top 5 character predictions. The result is further filtered using the Enchant Grammar Check [25] to remove the non-English words. If multiple word candidates exist, we attempt to guess the true word in the order of the words' frequencies. Among the 1215 correctly inferred segmentations, 858 words were identified with an average of 4.3 attempts. The word inference accuracy within 5 attempts, based on word length, is shown in Figure 18. GAZEPLOIT effectively handles words of varying lengths.

Consecutive Identical Keystrokes GAZEPLOIT has limitations in identifying multiple identical keystrokes as separate events due to the absence of saccade features, as the user's gaze does not move. This affects the inference of words like "better". To address this, we modified the dictionary by adding variants for words with consecutive identical keystrokes, such as "beter" for "better" and "wek" for "week". Using this method, 67 out of 134 words with consecutive identical keystrokes can be successfully identified.

6.4 GAZEPLOIT attack Robustness

We have already demonstrated that the capabilities of our attack in remote, end-to-end, and universal scenarios. Given that the attack does not require any prior knowledge or recalibration on users, we evaluate two other aspects of attack robustness in this subsection.

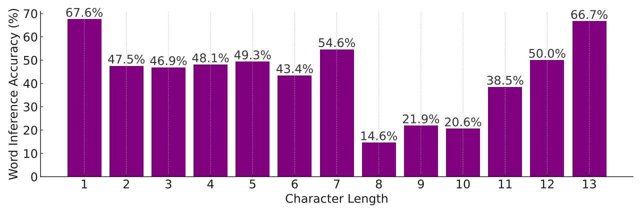


Figure 18: Word Inference Accuracy of different word length

Recording Persona Virtual Camera Placement The Persona virtual camera position is by default centered at the app window that is accessing the camera. What restricts the camera position is that the head pose it films cannot surpass the yaw and pitch angles up to ±80°,±80° because these angles are the max angles in the ETH-XGaze dataset that our gaze estimation model is trained on. We examined the head rotation during all typing sessions and find the rotations are within yaw and pitch angles ±25°,±20°. This finding leads to the Persona virtual camera placement should be within ±55°,±60° area, centered at the direct front of the users' head position to guarantee the gaze can be extracted without large errors.

We test GAZEPLOIT attack at these theoretical extreme virtual camera placement angles. Our keystroke recovery accuracy does not change very much within the angles. However, when the camera placement surpasses the angles, the accuracy drops drastically, as illustrated in Figure 19.

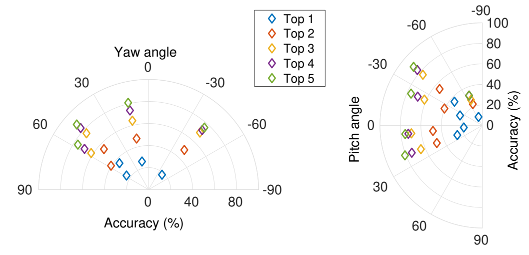


Figure 19: Key recovery accuracy with different camera placement angles

Attack Performance across participants We do not observe significant differences in attack performance across different genders, races, and ages. However, in the sentence typing inference experiment, there is a strong correlation between a participant's gaze typing proficiency and the prediction accuracy. To prove this, we plot the averaged character inference accuracy of all participants with deviations for the 15 sentences in Figure 20. It is evident that as users gain more practice in gaze typing and become more proficient, the inference accuracy increases, especially during the initial few sentence inputs, where character prediction accuracy improves by around 10%. This improvement is likely due to more proficient participants having more regular and predictable fixation and saccade patterns, as discussed in Sections 4.2 and 4.3. This observation indicates an increased threat level of our attack, as most real-world users will eventually become proficient in gaze typing.

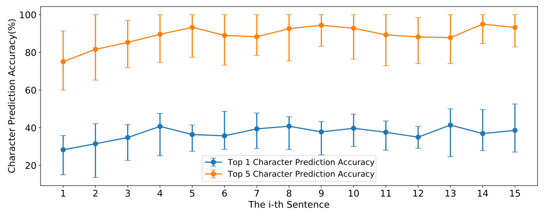


Figure 20: Character prediction Accuracy vs. 15 sentences in time sequence, averaged across participants

7 Discussion

In this section, we discuss the lack of awareness about VR/MR security vulnerabilities among hardware designers, software developers, and users, and how this can be addressed. We also compare our work, GAZEPLOIT, with prior works, emphasizing the unique contributions and advantages of our method. Finally, we propose countermeasures to mitigate the identified vulnerabilities, balancing the need for user security with the functionality and usability of VR/MR devices.

7.1 Unawareness of Vulnerabilities

There exists a common misconception among hardware designers, software developers, and users regarding VR/MR's virtual camera. This misconception leads them to treat it as a regular camera, overlooking the fact that it inherently captures more biometric data due to the immersive nature of VR/MR technologies. A notable example in our study is the eye gaze information, which can unintentionally leak typing information.

Our analysis of applications on the Apple App Store, including popular conference apps like Zoom, Microsoft Teams, and Slack, social media apps such as Reddit, WeChat, Tinder, Twitter, Discord, and video chat apps like FaceTime and Skype, reveals that many of these are vulnerable to this security flaw. This indicates a widespread lack of awareness and understanding of these unique vulnerabilities among developers and users alike.

However, fundamentally, hardware vendors bear the responsibility to ensure the security of these applications. The placement of avatar views, for instance, should not simply replicate the virtual camera's view due to the potential exposure of sensitive data, which may be even more critical than revealing the user's appearance itself. This reveals the need for a deeper understanding and more careful consideration of VR/MR's unique characteristics and potential risks among all parties involved.

7.2 Comparison with Prior Work

GAZEPLOIT is an end-to-end keystroke inference attack, uniquely combining VR environments and gaze estimation, setting it apart from prior works in both domains.

Comparison with prior work in VR Our work, GAZEPLOIT, stands out when compared with six state-of-the-art keystroke inference attacks in VR/MR devices, as outlined in Table 3 detailing the practical implementation and evaluation design.

A significant advantage of GAZEPLOIT is its ability to execute remote attacks by leveraging gaze information leaked through avatars, a method that integrates seamlessly into regular VR/MR usage. In contrast, Hidden Reality (HR) [15], Heimdall [29], and VR-spy [4] depend on side channels such as video, EM, and acoustics, requiring on-site equipment like cameras or smartphones for data capture. This makes them less feasible to carry out in practical scenarios. Meanwhile, GearVR [27] and Typose [42] utilize motion sensor data from the headset. However, unlike GAZEPLOIT utilizing the virtual camera in regular VR/MR usage, these attacks require malicious apps to access the motion sensor data which increase the complexity of the attack.

Table 3: Comparison with prior works in VR

Work	✈	👥	⌨	🔑	👁
Gear VR[27]	✓	✓	×	**M**	×
Typose[42]	✓	×	×	**N/A**	×
Hidden Reality[15]	×	✓	×	**H**	×
Heimdall[29]	×	✓	✓	**L**	×
VR-spy[4]	×	✓	✓	**M**	×
GAZEPLOIT	✓	✓	✓	**H**	✓

✈: Remote Attack, 👥: Universal Attack, ⌨: Identify Typing Sessions, 🔑: Password Strength in Dataset, 👁: Targeting Gaze Typing, **H/M/L**: High/Medium/Low

GAZEPLOIT also excels in its universal applicability, a feature shared with all other attacks except Typose which requires initial victim profiling. Regarding typing session identification, Heimdall incorporates preliminary results analyzing click frequency to identify typing sessions and VR-spy analyzed the frequency domain feature of channel state information to identify typing sessions. Our approach goes a step further by applying an RNN method to distinguish typing sessions with high precision. This level of sophistication and accuracy is unparalleled in other solutions.

Regarding password security, GAZEPLOIT ensures robustness by requiring passwords to meet Google account standards. This includes at least 8 characters involving letters, numbers, and special characters. In contrast, GearVR and VR-spy used simpler passwords, and Heimdall opted for weak passwords from a list of common choices. Only HR matches the complexity of GAZEPLOIT's password criteria, using computer-generated random passwords.

Most significantly, GAZEPLOIT is the only solution among those compared that targets gaze typing, an emerging feature in VR/MR technologies. Our study found that users overwhelmingly prefer this intuitive and efficient input method over hand-gesture-based or controller-based typing. Despite the growing adoption of gaze typing in state-of-the-art VR systems and its potential to become the standard input method, no other keystroke inference attack has considered this feature. GAZEPLOIT pioneers in this regard, identifying and addressing the security vulnerabilities associated with gaze typing.

Comparison with prior work in 2D scenarios All works that attempt to attack tablets/smartphones/laptops without using eye gaze information are ineffective in VR gaze-typing. For instance, "Zoom on the Keystrokes" [40] focuses on shoulder movement. This feature is absent in gaze-typing on a virtual keyboard since the victim does not need to move their hands.

In contrast, several works use eye gaze information to infer keystrokes on touchscreens but face limitations when applied to VR gaze-typing. Eyetell [8] utilizes features such as limbus detection which mandates fixed head postures and front-view recorded videos with narrow camera angles, making it impractical for VR scenarios. Furthermore, Eyetell identifies clicks based on **turn-points** in gaze traces, failing to recognize same-line keystrokes (e.g., "POWER" or "ROUTE") and relies on dictionary lookup, which **only works on words** but fails with more complex texts like passwords.

Similarly, GazeRevealer [44] uses the front camera of a smartphone to record the victim's view, limiting its applicability in VR where the camera can be placed at a wide-angle range. It requires the installation of a malicious app and can only infer PIN keyboards on smartphones and does not address typing session identification. While GazeRevealer employs pattern recognition to segment keystrokes, **its dataset does not include more complicated scenarios like message or password inference**.

Cazorla et al. [7] focus on laptops and physical keyboard and evaluate the accuracy of mapping gaze to keys using machine learning classifiers. However, they do not address keystroke segmentation or typing session identification, making their approach impractical for real-world scenarios. Additionally, they face limitations related to camera angle, which restricts their effectiveness in VR scenarios. Their work also requires participants to gaze at the physical keyboard while typing, which is not practical because experienced physical keyboard typers does not need to look at the keyboard.

Table 4: Comparison with prior works in 2D scenarios

Work	🔧	⌨	🔑	📷	👁
Eyetell[8]	✓	×	**N/A**	**L**	×
GazeRevealer[44]	×	×	**N/A**	**L**	×
Cazorla et al.[7]	×	×	**N/A**	**L**	×
GAZEPLOIT	✓	✓	**H**	**H**	✓

🔧: Practical Attack Scenario, ⌨: Identify Typing Sessions, 🔑: Password Strength in Dataset, 📷: Camera Angle Range, 👁: Targeting Gaze Typing, **H/M/L**: High/Medium/Low

Overall, these works demonstrate the challenges and limitations of adapting existing 2D keystroke inference techniques to VR gaze-typing scenarios, highlighting the unique strengths of GAZEPLOIT in effectively addressing these challenges, as depicted in Table 4.

7.3 Countermeasures

Addressing the vulnerabilities identified in the use of Vision Pro and Persona virtual camera involves implementing several countermeasures. These strategies aim to balance user security and usability while mitigating risks associated with the dual use of the eye tracker and virtual camera.

Randomized Keyboard: A randomized keyboard can obscure visual feedback of keystrokes, mitigating the risk of inferring passwords or sensitive information from eye movements [31]. Although this method could effectively prevent password leakage, it may reduce typing speed and accuracy, deviating from conventional interaction paradigms users are accustomed to [23].

Visual Indicators for Persona Sharing: Persistent visual indicators in the user's view during Persona sharing can reduce the risk of unintentional data exposure. These include icons or overlays signaling Persona's active status and notifications alerting users when Persona is activated.

Disabling Persona During Sensitive Inputs: Similar to security measures for screenshots during sensitive inputs, disabling the Persona view during password or passcode entry can enhance user privacy. However, it may lower user experience during social interactions using Persona.

8 Related Work

Keystroke Inference on VR Devices: Various side channels can be exploited for malicious keystroke inference during VR device usage, including graphic-based, physical movement, acoustic, and architecture-level side channels. GearVR [27] introduced video-based and motion sensor-based keystroke inference attacks in VR devices. Subsequent studies have explored subtle head movements, captured by motion sensors when users type on virtual keyboards [28, 42] or controller movement [26]. Other approaches include identifying hand gestures [15, 34] and analyzing clicking sounds from hand controller movements [29]. VR-spy [4] targets EM side-channels, while Yicheng Zhang et al. [49] target architecture-level performance data side channels. However, none of these approaches consider gaze data, which GAZEPLOIT uniquely exploits for a more effective and less intrusive remote attack.

Video-based Keystroke Inference Attack: Video-based keystroke inference attacks have been extensively explored for real keyboards and touchscreens [8, 40, 44, 45]. However, these methods often impose specific conditions and assumptions that limit their applicability. In contrast, GAZEPLOIT can be executed under a broader range of conditions. It leverages the virtual camera's view in VR/MR systems and extract the eye gazing data, which is consistently available and unaffected by physical world constraints.

Appearance-based Gaze Estimation: Zhang et al. proposed a CNN and appearance-based gaze estimation [47] and published a large dataset [48] for gaze estimation research. They later published the ETH-XGaze dataset [46], which includes over 1 million images of participants gazing at a wide range. GAZEPLOIT uses a model trained on this dataset, leveraging its robustness for effective gaze-based keystroke inference.

9 Conclusion

In this paper, we introduced GAZEPLOIT, a keystroke inference attack leveraging eye gaze data from VR/MR devices. Our research highlighted significant security vulnerabilities with eye-tracking technologies. Through experiments, we demonstrated how attackers could reverse engineer confidential keystrokes by analyzing video recordings of eye movements during text entry. Apple has implemented robust security measures to mitigate these risks. Future efforts should also focus on improving security protocols to protect users in VR/MR environments and exploring other attacks exploiting the leaked gaze information in VR/MR devices.

10 Acknoledgement

This work was partially supported by the National Science Foundation under Award Number 2028897 and 1916175 and partially supported by Intel Gift Donation.

References

[1] Apple. 2024. Apple announces new accessibility features, including Eye Tracking. https://www.apple.com/newsroom/2024/05/apple-announces-new-accessibility-features-including-eye-tracking/

[2] Apple. 2024. *Apple Vision Pro Privacy Overview Learn How Apple Vision Pro and visionOS Protect Your Data.* https://www.apple.com/privacy/docs/Apple_Vision_Pro_Privacy_Overview.pdf

[3] Apple. 2024. visionOS Overview. https://developer.apple.com/visionos

[4] Abdullah Al Arafat, Zhishan Guo, and Amro Awad. 2021. VR-Spy: A Side-Channel Attack on Virtual Key-Logging in VR Headsets. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. 564–572. https://doi.org/10.1109/VR50410.2021.00081

[5] Tanja Blascheck, Kuno Kurzhals, Michael Raschke, Michael Burch, Daniel Weiskopf, and Thomas Ertl. 2017. Visualization of eye tracking data: A taxonomy and survey. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 260–284.

[6] Nacho Caballero. 2024. Word-By-Frequency. https://github.com/nachocab/words-by-frequency

[7] José Reverte Cazorla, José María De Fuentes, and Lorena González-Manzano. 2022. Eye-based keystroke prediction for natural texts – a feasibility analysis. In *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 375–382. https://doi.org/10.1109/TrustCom56396.2022.00059

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

[9] Yihua Cheng, Haofei Wang, Yiwei Bao, and Feng Lu. 2024. Appearance-based Gaze Estimation with Deep Learning: A Review and Benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024), 1–20. https://doi.org/10.1109/TPAMI.2024.3393571

[10] HTC Corporation. 2024. *About the VIVE controllers.* https://www.vive.com/au/support/cosmos-external-tracking-faceplate/category_howto/about-the-controllers.html

[11] Roblox Corporation. 2018. Roblox. https://www.roblox.com/

[12] Mayar Elfares, Pascal Reisert, Zhiming Hu, Wenwu Tang, Ralf Küsters, and Andreas Bulling. 2024. PrivatEyes: Appearance-based Gaze Estimation Using Federated Secure Multi-Party Computation. *arXiv preprint arXiv:2402.18970* (2024).

[13] Fabio. 2022. Eye Tracking Keyboard. https://github.com/fabio914/EyeTrackingKeyboard

[14] Tobias Fischer, Hyung Jin Chang, and Yiannis Demiris. 2018. Rt-gene: Real-time eye gaze estimation in natural environments. In *Proceedings of the European conference on computer vision (ECCV)*. 334–352.

[15] Sindhu Reddy Kalathur Gopal, Diksha Shukla, James David Wheelock, and Nitesh Saxena. 2023. Hidden Reality: Caution, Your Hand Gesture Inputs in the Immersive Virtual World are Visible to All!. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 859–876. https://www.usenix.org/conference/usenixsecurity23/presentation/gopal

[16] Mamoru Hirota, Ayumu Tsuboi, Masayuki Yokoyama, and Masao Yanagisawa. 2018. Gesture recognition of air-tapping and its application to character input in VR space. In *SIGGRAPH Asia 2018 Posters* (Tokyo, Japan) *(SA '18)*. Association for Computing Machinery, New York, NY, USA, Article 45, 2 pages. https://doi.org/10.1145/3283289.3283335

[17] hysts. 2021. *A demo program of gaze estimation models (MPIIGaze, MPIIFaceGaze, ETH-XGaze).* https://github.com/hysts/pytorch_mpiigaze_demo

[18] Apple Inc. 2024. About Optic ID advanced technology. https://support.apple.com/en-us/118483

[19] Apple Inc. 2024. *Apple Vision Pro.* https://www.apple.com/apple-vision-pro/#:~:text=Data%20from%20cameras%20and%20sensors,you%20tap%20your%20fingers%20together.

[20] Apple Inc. 2024. Type with the virtual keyboard on Apple Vision Pro. https://support.apple.com/zh-cn/guide/apple-vision-pro/tana14220eef/visionos

[21] Meta Platforms Inc. 2024. *Eye tracking on Meta Quest Pro.* https://www.meta.com/help/quest/articles/getting-started/getting-started-with-quest-pro/eye-tracking

[22] VRChat Inc. [n. d.]. VRChat. https://hello.vrchat.com/

[23] Xinhui Jiang, Jussi PP Jokinen, Antti Oulasvirta, and Xiangshi Ren. 2022. Learning to type with mobile keyboards: Findings with a randomized keyboard. *Computers in Human Behavior* 126 (2022), 106992.

[24] Jasmine Katatikarn. 2022. Virtual Reality Statistics: The Ultimate List in 2024. https://academyofanimatedart.com/virtual-reality-statistics/

[25] Dom Lachowicz. 2024. Enchant. https://abiword.github.io/enchant/

[26] Jiyeon Lee, Hyosu Kim, and Kilho Lee. 2023. VRKeyLogger: Virtual keystroke inference attack via eavesdropping controller usage pattern in WebVR. *Computers & Security* 134 (2023), 103461.

[27] Zhen Ling, Zupei Li, Chen Chen, Junzhou Luo, Wei Yu, and Xinwen Fu. 2019. I Know What You Enter on Gear VR. In *2019 IEEE Conference on Communications and Network Security (CNS)*. 241–249. https://doi.org/10.1109/CNS.2019.8802674

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

[29] Shiqing Luo, Anh Nguyen, Hafsa Farooq, Kun Sun, and Zhisheng Yan. 2024. Eavesdropping on Controller Acoustic Emanation for Keystroke Inference Attack in Virtual Reality. In *The Network and Distributed System Security Symposium (NDSS)*.

[30] Antonio Maffei and Alessandro Angrilli. 2018. Spontaneous eye blink rate: An index of dopaminergic component of sustained attention and fatigue. *International Journal of Psychophysiology* 123 (2018), 58–63.

[31] Anindya Maiti, Murtuza Jadliwala, and Chase Weber. 2017. Preventing shoulder surfing using randomized augmented reality keyboards. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 630–635. https://doi.org/10.1109/PERCOMW.2017.7917636

[32] Meta. 2022. Metaverse Avatars: Reinvent Yourself in VR. https://www.meta.com/avatars/

[33] Meta. 2024. Meta Quest Pro: Our Most Advanced New VR Headset. https://www.meta.com/quest/quest-pro/

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

[35] Saraju P Mohanty. 2021. Intel Pentium Processor. (2021).

[36] Tomas Novacek and Marcel Jirina. 2020. Overview of controllers of user interface for virtual reality. *PRESENCE: Virtual and Augmented Reality* 29 (2020), 37–90.

[37] Weiyang Qian and Rodolfo WL Coutinho. 2023. A Reinforcement Learning-based Orchestrator for Edge Computing Resource Allocation in Mobile Augmented Reality Systems. In *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 1–6.

[38] Vijay Rajanna and John Paulin Hansen. 2018. Gaze typing in virtual reality: impact of keyboard design, selection method, and motion. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications* (Warsaw, Poland) *(ETRA '18)*. Association for Computing Machinery, New York, NY, USA, Article 15, 10 pages. https://doi.org/10.1145/3204493.3204541

[39] Emma Roth. 2023. Apple's Vision Pro headset will turn you into a digital avatar when FaceTiming. *The Verge* (Jun 2023). https://www.theverge.com/2023/6/5/23750096/apple-vision-pro-headset-persona-facetime

[40] Mohd Sabra, Anindya Maiti, and Murtuza Jadliwala. 2021. Zoom on the Keystrokes: Exploiting Video Calls for Keystroke Inference Attacks. In *Network and Distributed Systems Security (NDSS) Symposium 2021*.

[41] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[42] Carter Slocum, Yicheng Zhang, Nael Abu-Ghazaleh, and Jiasi Chen. 2023. Going through the motions: AR/VR keylogging from user head motions. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 159–174. https://www.usenix.org/conference/usenixsecurity23/presentation/slocum

[43] Sony. [n. d.]. PlayStation®VR2 | The next generation of VR gaming on PS5. https://www.playstation.com/en-us/ps-vr2/

[44] Yao Wang, Wandong Cai, Tao Gu, and Wei Shao. 2019. Your eyes reveal your secrets: An eye movement based password inference on smartphone. *IEEE transactions on mobile computing* 19, 11 (2019), 2714–2730.

[45] Zhuolin Yang, Yuxin Chen, Zain Sarwar, Hadleigh Schwartz, Ben Y. Zhao, and Haitao Zheng. 2023. Towards a General Video-based Keystroke Inference Attack. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 141–158. https://www.usenix.org/conference/usenixsecurity23/presentation/yang-zhuolin

[46] Xucong Zhang, Seonwook Park, Thabo Beeler, Derek Bradley, Siyu Tang, and Otmar Hilliges. 2020. Eth-xgaze: A large scale dataset for gaze estimation under extreme head pose and gaze variation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer, 365–381.

[47] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. 2015. Appearance-based gaze estimation in the wild. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4511–4520. https://doi.org/10.1109/CVPR.2015.7299081

[48] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. 2017. Mpiigaze: Real-world dataset and deep appearance-based gaze estimation. *IEEE transactions on pattern analysis and machine intelligence* 41, 1 (2017), 162–175.

[49] Yicheng Zhang, Carter Slocum, Jiasi Chen, and Nael Abu-Ghazaleh. 2023. It's all in your head(set): Side-channel attacks on AR/VR systems. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 3979–3996. https://www.usenix.org/conference/usenixsecurity23/presentation/zhang-yicheng

[50] Maozheng Zhao, Alec M Pierce, Ran Tan, Ting Zhang, Tianyi Wang, Tanya R Jonker, Hrvoje Benko, and Aakar Gupta. 2023. Gaze Speedup: Eye Gaze Assisted Gesture Typing in Virtual Reality. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*. 595–606.