I Know What You Enter on Gear VR

Zhen Ling*, Zupei Li†, Chen Chen*, Junzhou Luo*, Wei Yu‡, and Xinwen Fu§
*Southeast University, China, Email:{zhenling, cchen2016, jluo}@seu.edu.cn
†University of Massachusetts Lowell, Email: zli1@cs.uml.edu
‡Towson University, Email: wyu@towson.edu
§University of Central Florida, Email: xinwenfu@ucf.edu

***Abstract*—Virtual reality (VR) techniques offer users immersive experiences in a virtual environment (VE) where they can enjoy gaming, training, shopping and social activities. However, security and privacy issues of VR are rarely investigated. In this paper, we introduce novel computer vision-based and motion sensor-based side channel attacks to infer keystrokes in a virtual environment. In the computer vision-based attack, a stereo camera records a user maneuvering a pointing device and inputting passwords. Various computer vision techniques are used to detect the touching frames in which the user taps a touchpad to enter keys on the virtual keyboard within the VE. We then estimate the pose of the headset and derive the orientation of the virtual pointing ray in each touching frame. In the motion sensor-based attack, we exploit sensors including accelerometer, gyroscope and magnetometer built inside the pointing device to retrieve orientation angles of the pointing device. With orientation angles of the pointing device, we can derive the empirical rotation angles and infer the clicked keys by assessing the similarity between the empirical rotation angles and the reference ones between keys. Extensive real-world experiments are performed to demonstrate the feasibility and effectiveness of these side channel attacks.**

I. Introduction

Augmented reality (AR) and virtual reality (VR) technologies have been booming. AR and VR applications create an immersive virtual environment (VE) for people to perform various activities including gaming, training, shopping and socializing. VR devices such as Samsung Gear VR often use head-mounted displays (HMD). Ray pointing techniques are commonly adopted for the human and VE interaction. A VR headset or a VR controller can be employed as a pointing device to cast a virtual laser ray and to select an object, functioning like a laser pointer. To input, the user points the ray to a key on a keyboard in the VE, and taps the touchpad of the headset or controller.

In this paper, we explore the security of the human and VE interface of AR and VR techniques. We will use VR in our discussion while the results and observations from this research can be extended to AR too. Intuitively, the movement of the pointing device creates a side channel that may leak sensitive information the user enters with the pointing device. We will present novel computer vision-based and motion sensor-based side channel attacks to infer the user's password entered with a VR headset and a controller of the HMD-based virtual reality system. The attacks can also be extended to infer other inputs with the pointing device. We choose Samsung Gear VR as an example to demonstrate the security issues of a VR system given its popularity. The observations and related techniques can be extended to other VR devices since the underlying inputting techniques are similar. To the best of our knowledge, **we are the first to explore such side channel attacks against the HMD-based VR systems**. George *et al.* [5] find that the PIN and pattern-based authentication methods in VR can be more resistant to shoulder surfing attacks than the methods applied to the smartphone in the physical world. Our work shows that these traditional mobile authentication techniques are not secure under the side channel attacks in VR. and show the necessity of securing the human and VR interface.

Major contributions of this paper are summarized as follows. We explore a stereo vision-based side channel to infer user passwords entered with the Samsung Gear VR headset. A stereo camera is leveraged to record the user operating a VR headset. We use the optical flow to track the headset as the user rotates it and controls a ray pointing at a virtual keyboard to input. By analyzing the finger's interaction with the touchpad of the headset, we can automatically identify the key click frames in which the finger taps the touchpad and inputs. We estimate the headset's pose in each key click frame so as to calculate the virtual pointing ray's orientation change, denoted as empirical rotation angle, between two sequentially tapped keys. We can then infer the clicked keys by assessing the similarity between the empirical rotation angles and the reference ones, which are derived by measuring the virtual keyboard in the VE.

We also investigate side channel attacks in which a malicious app exploits motion sensors in VR input devices. The effectiveness of vision-based attacks is limited by computer vision techniques and the capability of the stereo camera. It is also hard to conduct vision side channel attacks against the Gear VR controller since there is no salient feature to track on the controller. However, in the case of a user using a controller to input passwords, a malicious app can obtain orientation sensor data from the controller through the Oculus mobile SDK. Note that permissions are not needed to obtain orientation sensor data on mobile devices. We can also identify the key click events from the sensor data. Therefore, we can compute the empirical rotation angles between two sequentially pressed keys. A password inference method similar to the one used in the vision attack can then be employed to infer the clicked keys.

We perform extensive experiments to validate these side channel attacks against Gear VR. Due to the limitations of computer vision techniques, the stereo camera used in the vision attack has to be deployed close to the victim. Our attacks generate a list of password candidates. If the very top candidate is the original password, we call it a top 1 success. If any one of the top 3 candidates is the original password, we call it a top 3 success. The vision attack's top 3 success rate for **8-character** passwords is 63% at a distance of 1.5m when the headset is used as the input device. The advance of stereo cameras will improve the attack distance. In contrast, the orientation sensor attacks are not limited by the distance, and can achieve a success rate of around 90% for 8-character passwords entered with both the headset and controller.

The rest of the paper is organized as follows. In Section II, we introduce the Samsung Gear VR system and its input

978-1-5386-7117-7/19/$31.00 ©2019 IEEE

devices. Section III elaborates on the threat model and basic idea of the side channel attacks against Gear VR. We then present the the computer vision-based password inference attack in Section IV and the motion senor-based attack in Section V. Extensive experiments are performed to evaluate the attacks and the results are presented in Section VI. We review related work in Section VII and conclude the paper in Section VIII.

II. Introduction to Gear VR

In this section, we introduce the Samsung Gear VR system and its input devices including the headset and controller.

A. An Overview of Gear VR

A head-mounted display (HMD) based VR system consists of a VR engine, input devices, and output devices as shown in Fig. 1. The VR engine aligns coordinates between physical and virtual worlds using motion sensor data from the HMD. Input devices allow users to interact with the virtual environment, including the manipulation interface, position tracker, and gesture interface. The output devices are used for visual, auditory, and haptic feedbacks.

Gear VR is one of the most popular smartphone-based VR systems. It consists of three components: a Samsung smartphone, a VR headset, and a controller. The VR software is installed on the Android system of the Samsung smartphone. The smartphone mounted to a Gear VR headset is the VR engine, and uses its screen and speaker to generate the visual and auditory feedbacks for users. The input devices, the headset and the controller, leverage manipulation interfaces such as the touchpad and buttons and position trackers (i.e. motion sensors) to acquire the user input.

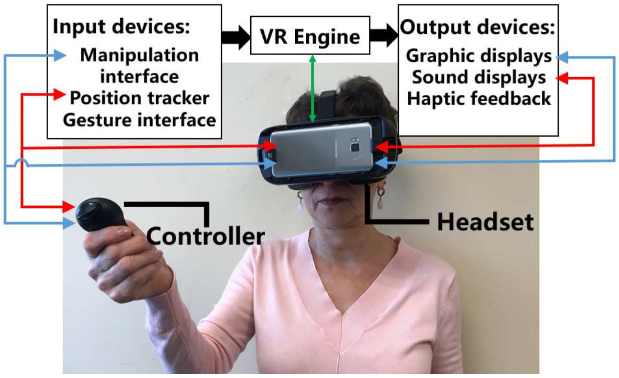


Fig. 1. Components of a virtual reality system

The VR engine builds virtual objects by using the geometric modeling technology, and creates a virtual environment upon a virtual world coordinate system as shown in Fig. 2. It is assumed that the user's eye is at the origin of the virtual world coordinate system. When the user puts on the headset, the virtual world coordinate system is aligned with the physical world and its XZ-plane is set in parallel with the earth's horizon. When the user turns her head thus the headset around, the virtual engine determines the orientation of the headset and presents the corresponding virtual scene to the user.

B. Input Devices

Samsung Gear VR allows users to interact with virtual objects through ray pointing techniques, which are commonly used to control the cursor movement and point at objects in a virtual environment. The intersection of a ray with a virtual object can be used to show where a user is pointing to. The ray pointing techniques are implemented for both the VR headset and the controller.

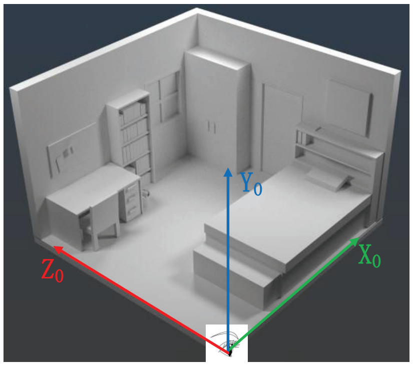


Fig. 2. Virtual world coordinate system

1) Headset: A user can control a cursor in the virtual environment by rotating the VR headset in three axes as shown in Fig. 3. Motion sensors such as accelerometer, gyroscope, and magnetometer are embedded into the Gear VR headset to track the user's head motion [17] while some lightweight smartphone-based VR systems such as Google Cardboard only rely on a smartphone's motion sensors to track the head motion. A user inputs using a touchpad, a home button and a back button on the headset. A Samsung smartphone is connected to the headset via a Micro USB or USB Type-C port. In the virtual environment, the headset casts a virtual ray along the user's gaze direction (i.e., the opposite direction of Z_h axis of the headset) as illustrated in Fig. 4. The user can control the orientation of the headset in only two axes, Yaw (x axis) and Pitch (y axis), to move the cursor. The motion sensors can be used to track the head motion in these two dimensions and derive the orientation of the headset. To enter a key in the virtual environment, the user controls the headset to move the cursor to a target key on a keyboard as shown in Fig. 5 and then taps the touchpad on the headset so as to input the intended key.

2) Controller: A user can also control the cursor via a VR controller. A VR controller contains motion sensors (i.e. accelerometer, gyroscope, and magnetometer), a touchpad and control buttons (i.e. a trigger, a home button, a back button, and volume buttons). The controller is connected to the Samsung smartphone via Bluetooth. Fig. 6 illustrates the VR controller coordinate system. In the virtual environment, a ray is cast by the controller along the opposite direction of the controller's Z_c axis. The intersection of the ray with a virtual object is the position of the cursor as illustrated in Fig. 7. The user can adjust the orientation of the controller in only two axes, Yaw (x axis) and Pitch (y axis), to control the cursor movement. When the cursor is moved to an intended key on a virtual keyboard as shown in Fig. 8, the user can click on the touchpad of the controller and enter the key.

III. Overview of Side Channel Attacks

In this section, we present the threat model of our side channel attacks and the basic idea of these attacks.

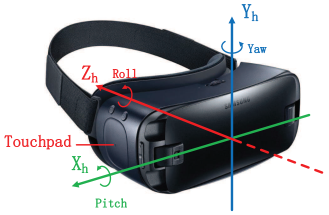


Fig. 3. Gear VR headset coordinate system

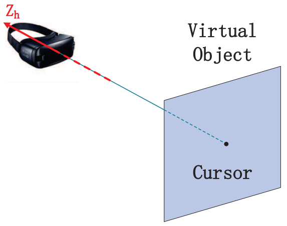


Fig. 4. Cursor control using a VR headset

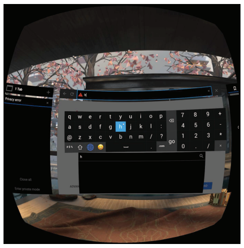


Fig. 5. A left-eye view created for headset input

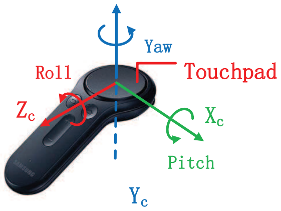


Fig. 6. VR controller coordinate system

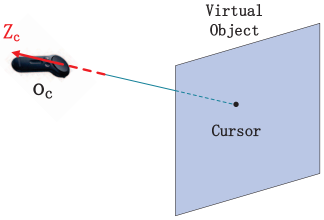


Fig. 7. Cursor control using a VR controller

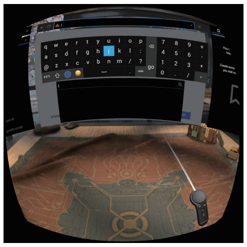
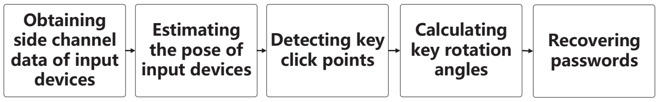


Fig. 8. A left-eye view created for controller input



Obtaining side channel data of input devices → Estimating the pose of input devices → Detecting key click points → Calculating key rotation angles → Recovering passwords

Fig. 9. Workflow of our attacks

A. Threat Model

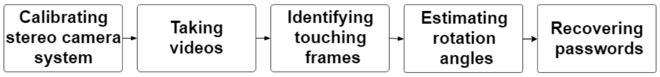
We assume that VR devices will be ubiquitously used [6], [13] Users will use those devices and perform daily activities, such as entering passwords for the login of gaming, emails, mobile banking accounts and other purposes. However, attackers will be around. It is assumed that the virtual keyboard layout can be learned in advance by reverse-engineering the VR app and examining the keyboard layout.

In the vision-based side channel attack, we assume that an attacker is able to use a stereo camera to take videos of a user inputting, for example her password, on a Samsung Gear VR through the headset. Since the view of the victim in the physical world is completely blocked by the headset, the attack can be stealthily conducted at a close distance.

In the motion sensor-based side channel attack, it is assumed that the victim has been tricked to install a malicious app on her smartphone. Then the malware can obtain data from motion sensors such as accelerometer, gyroscope, and magnetometer. Actually any app can perform the attack since accessing such senor data does not require extra permission on the Android system.

B. Basic Idea

Fig. 9 illustrates the workflow of the side channel attacks. Without loss of generality, we take the popular Gear VR as an example to present our attacks in this paper. To control the cursor movement in the virtual world, a user rotates input devices such as the headset and the controller in the yaw and pitch dimensions as shown in Fig. 3 and Fig. 6. The built-in senors (accelerometer, gyroscope, and magnetometer) are leveraged to perform rotational tracking so as to retrieve the pose of these input devices. The motion of the input devices in the yaw and pitch dimensions creates side channels. In the vision-based side channel attack, we exploit a stereo camera to record the rotation of the headset and use various stereo computer vision techniques to estimate the pose of the headset. In the motion sensor-based side channel attack, we leverage a malicious app installed inside on the smartphone to read the orientation angles from the built-in senors of the headset and the controller. From the sequence of orientation angles, we can determine where a key click occurs and thus know the index of the clicked key in terms of the orientation angle time series. We find that the rotation angle from a key to another one is fixed. Therefore, we can measure orientation angles of all the keys on a reference keyboard and further derive *rotation* angles between keys. Since the last key is often the *go* key as shown in Fig. 5 in the virtual environment of Gear VR, we can calculate the rotation angles between the clicked key and the *go* key using the orientation angles. Finally, we can infer each key in the password by comparing the similarity between the empirical rotation angles derived from the side channel attacks and the reference rotation angles derived from the reference keyboard. If there is no *go* key or the user does not use the *go* key, we design a brute force attack in which the trajectory of the ray is fit onto the keyboard from upper left to low right. The trajectory within the region of the keyboard creates a password candidate.



Calibrating stereo camera system → Taking videos → Identifying touching frames → Estimating rotation angles → Recovering passwords

Fig. 10. Workflow of computer vision-based attack

IV. Computer Vision-based Attack

In this section, we first present the workflow of our vision attack against Gear VR and then details of each step.

A. Workflow of Computer Vision-based Attack

Fig. 10 shows the workflow of the computer vision-based attack.

Step 1: Calibrating the stereo camera system. We calibrate a stereo camera system and derive its parameters. These parameters are the key specification of the camera and are used to build a 3D coordinate system for the stereo camera. These parameters are also used to compute the real-world coordinates of the points in a video.

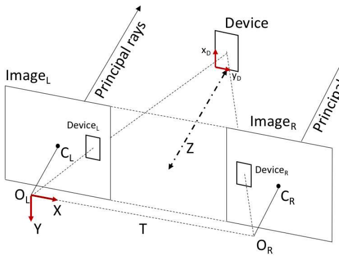


Fig. 11. Image formation process in the stereo camera system

Step 2: Taking videos. We take a 3D video of the victim inputting passwords through a Gear VR headset.

Step 3: Identifying touching frames. We preprocess the video by trimming it and then identify the touching frames. We track multiple feature points of the VR headset in the video frames so as to identify the touching frames, in which the fingertip taps the touchpad of the VR headset.

Step 4: Estimating rotation angles. By analyzing the spatial relationship between the feature points in different touching frames, we estimate the headset's rotation matrices between the first touching frames. The rotation matrices can then be used to calculate the rotation angles.

Step 5: Recovering passwords. Rotation angles can be employed to infer clicked keys. Since the reference keyboard layout is learned in advance, we can derive the reference rotation angles between the *go* key and any other keys. We compare the rotation angles between the *go* key and the other clicked keys with the reference ones so as to infer the key position on the keyboard. Finally, we can recover the whole password with each key located. A brute force attack can also be used to infer a clicked key if there is no such *go* key.

B. Step 1: Calibrating Stereo Camera System

To retrieve a target's coordinates in a 3D coordinate system of a stereo camera, we need to have an undistorted, aligned, and measured stereo camera system as illustrated in Fig. 11. To this end, we perform stereo camera calibration for our camera, ZED [14], to estimate the camera's parameters. We use the ZED camera to take several photos of a chessboard placed on a flat glass plate from different distances and poses. Since the accuracy of the calibration is affected by the quantity and quality of the chessboard photos, at least ten photos are carefully taken at a particular distance in our experiments. We employ a calibration algorithm [3] with these photos to derive the parameters of the stereo camera system, including a camera intrinsic matrix , camera distortion coefficients , and a geometrical relationship between the left and right cameras. The geometrical relationship can be represented by a rotation matrix $\mathbf{R}$ and a translation vector $\bar{\mathbf{T}}$. The intrinsic matrix includes the camera focal length, principal point offset, and axis skew of the camera. The distortion coefficients refer to the radial and tangential distortions of the camera.

Fig. 11 shows the image formation process in the stereo camera system. The projection center and the principal point of the left camera are denoted as O_L and C_L. Comparatively, for the right camera they are respectively denoted as O_R and C_R. $\bar{\mathbf{T}}$ symbolizes the translation relationship between the two cameras. The projections of $Device$ on the left and the right camera image planes are denoted as $Device_L$ and $Device_R$, respectively. After the calibration of the stereo camera system, we can build the 3D coordinate system, with the origin set at the left camera's lens center O_L.

C. Step 2: Taking Videos

The attacker uses the ZED stereo camera to take a video of the victim inputting through a Gear VR headset. Since the victim is submerged in the VR world, the headset blinds the user from seeing anything in physical world. Therefore, the attack can be stealthily performed.

The distance, resolution, frame rate and filming angle are the top four factors that affect the outcome of the attack. A ZED camera is equipped with two wide angle lenses without an optical zooming function. In order to acquire high-quality images of the victim's hand and the headset, the attacker shall carefully choose the distance between the camera and the victim. High resolution images are required to achieve accurate keystroke inference. Since we need to acquire feature points of the VR headset and the touching finger, the victim's finger movement should be clearly captured. An appropriate video sampling rate of the camera system should be carefully selected [9], [20]. We use the ZED camera to record videos of 1920×1080 pixels at 30 frames per second (FPS).

D. Step 3: Identifying Touching Frames

1) Rectifying stereo images: We process the stereo videos to derive undistorted and aligned left and right images. To reduce workload and speed up calculation for later steps, we trim the video clip and keep the part related with the inputting process. We then perform stereo rectification using Bouguet's algorithm [3]. After rectification, the video frames of left and right cameras can be aligned and we can derive a reprojection matrix as follows,

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & 1/T_x & 0 \end{bmatrix}, \qquad (1)$$

where $(c_x,\ c_y)$ is the coordinate of the principal point, f is the focal length of the left camera, and T_x is the translation parameter of the x axis.

2) Tracking headset: To estimate the VR headset's pose, we need to track the headset movement during the whole inputting process. To this end, we first use a boundary box in images from the left camera to specify the area of interest on the headset. Then we apply the optical flow technique [16] to extract and track multiple feature points in the boundary box. The optical flow tracks the points by estimating the similarity of a small area around points of interest.

To improve the accuracy of the headset tracking, we use the K-means clustering algorithm to automatically select effective feature points on the headset. Some feature points in the boundary box are not on the headset since the optical flow algorithm may extract and track the feature points located in the background or on the victim's head. To address this issue, we calculate the average 2D movement velocity of feature points in several sequential frames. Then the K-means clustering algorithm is performed on the sequence of velocity data with $K = 2$ and we only keep the cluster of feature points with a higher velocity.

Since the victim's head may move a lot in a wide range in order to target the virtual ray at a key on the keyboard and input it, we may lose most of the feature points due to the limitation of the optical flow technique and cannot use the remaining few feature points for later estimation of the headset's pose. In our experiments, we find that it needs at least four feature points distributed around the four corners of the smartphone. $S_{u,l}$, $S_{u,r}$, $S_{l,l}$ and $S_{l,r}$ are the sets of upper-left corner, upper-right corner, lower-left and lower-right corner feature points. A threshold is used to determine if a feature point is around a corner. If any of the four corner feature point sets has no feature point any more in a frame during the headset tracking process, we restart the feature point tracking process discussed above from that specific frame.

3) Locating touching frames: To determine the occurrence of a key click, we analyze the touching finger movement pattern and then locate the touching frames in which the victim taps the touchpad on the right side of the headset. To enter a key, the inputting finger moves towards the touchpad, taps it and then moves back. At this point, the headset is relatively still. We define the movement direction toward the headset as positive and the opposite direction as negative. Thus, a touching frame is the one where the inputting finger's movement direction changes from positive to negative.

We track the touching finger using the optical flow in the 2D video frames of left camera during the inputting process. We track multiple points on the inputting finger and calculate their average speed vector $\boldsymbol{V_f}$ to enhance robustness. We also derive the speed vector $\boldsymbol{V_h}$ of a feature point (derived in Section IV-D2) on the headset. We then derive the relative movement vector $\boldsymbol{V_r} = \boldsymbol{V_f} - \boldsymbol{V_h}$ of the touching finger. Since the finger touches the headset almost in the horizontal direction in 2D video frames, we choose a touching frame by the following rule: in a specific frame, if $\boldsymbol{V_r}$ changes from positive to negative in the x axis and the headset movement speed $|\boldsymbol{V_h}|$ is below its average movement speed through the inputting process, that frame is a touching frame.

E. Step 4: Estimating Rotation Angles

In this step, we calculate 3D coordinates of feature points of the VR headset, and estimate the headset's pose in every touching frame. The pose includes 6 degrees-of-freedom (DOF), including rotation and translation along three axes.

1) Calculating 3D coordinates of the VR headset: We calculate the 3D coordinates of feature points of the VR headset in the stereo camera coordinate system as shown in Fig. 11 for every touching frame. To this end, we need to derive the 2D coordinates of the corresponding points in $Device_L$ and $Device_R$ and their disparity d, where d is the horizontal coordinate difference of corresponding points. Recall that we derive feature points in $Device_L$. Then the corresponding points in $Device_R$ can be located by a sub-pixel accuracy template matching algorithm [12].

After obtaining a pair of corresponding points in the left and right images, we can derive the disparity d. According to the 2D coordinates of the feature points in the left frame and the reprojection matrix $\mathbf{Q}$ in Section IV-D, we can calculate the 3D coordinates of the feature points as follows,

$$Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}, \tag{2}$$

where (x, y) is the 2D coordinate of the point in $Device_L$, and $d = x - x_R$ is the disparity of this point and its corresponding point (x_R, y_R) in $Device_R$. W is the scale factor. Then the 3D coordinate of the point can be calculated by $(X/W, Y/W, Z/W)$. In this way, we can derive the 3D coordinates of all the feature points in the attacking camera's coordinate system. Denote F_i as a set of 3D coordinates of all the feature points in the i^{th} touching frame.

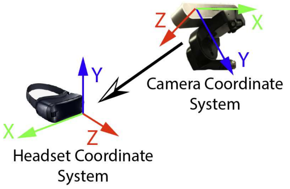


Fig. 12. Converting camera coordinate system to headset coordinate system

2) Converting coordinate system: To estimate the pose of the head, we first establish a coordinate system for the headset and then convert the 3D coordinates of feature points from the camera coordinate system to the headset coordinate system. As shown in the newly designed headset coordinate system in Fig. 12, we set the origin point of the new coordinate system at the intersection of the lower and left edge of the smartphone in the first touching frame, with the x axis aligning with the lower edge of the phone and the y axis aligning with the left edge of the phone. We convert all 3D coordinates of the feature points in every touching frame to this headset coordinate system. Denote F_i' as the converted 3D coordinate of a feature point F_i in the i^{th} touching frame in the headset coordinate system. The spatial relationship of the headset in different frames can be presented as follows,

$$F_1' = \boldsymbol{R} * F_i' + \boldsymbol{t}, \tag{3}$$

where F_1' is a feature point in the first frame, $\boldsymbol{R}$ and $\boldsymbol{t}$ are the rotation and translation relationship between F_i' and F_1'.

Since the attacker's camera is mounted on a fixed tripod, the camera does not move during the video recording process. We use the camera to take a photo of a chessboard placed on the horizontal plane. Note that a digital protractor is used to ensure that the chessboard is horizontally placed, thus the chessboard can represent the horizontal plane in this case. Then, by analyzing this photo, we can calculate a rotation matrix between the headset coordinate system and the horizontal plane, denoting it as R_H.

3) Estimating rotation angles of the VR headset: After the coordinate system conversion, we estimate the rotation angles between the first clicked key and any other clicked key by calculating the spatial relationship between F_1' and F_i'. The pose difference between F_1' and F_i' can be represented by a rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{t}$, as shown in Equation (3). In our context, we just need the rotation matrix $\boldsymbol{R}$. Denote $C_{F_1'}$ and $C_{F_i'}$ as the centroid of all feature points in the first and the i^{th} frames. We can derive the $C_{F_i'}$ as follows,

$$\begin{gathered} F_i'[j] = \begin{bmatrix} x_j \\ y_j \\ z_j \end{bmatrix}, \\ C_{F_i'} = \frac{1}{n} \sum_{j=1}^{n} F_i'[j] \end{gathered} \tag{4}$$

where $F_i'[j]$ is the j^{th} feature point in the i^{th} frame, (x, y, z) is the 3D coordinate of $F_i'[j]$ in the headset coordinate system, and n is the total number of the feature points in the frame.

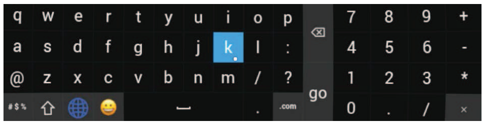


Fig. 13. Keyboard's layout in Oculus Browser

After we acquire the centroid of both sets of feature points, we calculate the rotation matrix $\boldsymbol{R}$ between these two poses. The singular value decomposition (SVD) is used to obtain the rotation matrix. We put both $C_{F_i'}$ and $C_{F_1'}$ at the origin point so as to remove the translation component t. Then the rotation matrix can be solved as follows [4], [2],

$$\begin{aligned} H &= \sum_{j=1}^{n}(F_1'[j] - C_{F_1'})(F_i'[j] - C_{F_i'})^T \\ [U,S,V] &= SVD(H) \\ R &= U^T \end{aligned} . \tag{5}$$

In this way, we can derive a sequence of rotation matrices in the headset coordinate system. We then use R_H to calculate the rotation matrix between headset coordinate system to the horizontal plane R_P. By multiplying R_P to R, we align all rotation matrices to the horizontal plane.

Denote N as the number of clicks on the keyboard and $(P_1, P_2, \ldots, P_N)$ as the sequence of key click events that corresponds to the touching frames, where P_N is the *go* key. Then we can derive the rotation angles between P_1 and P_i in the pitch and yaw by

$$R_i \Rightarrow (\Delta\alpha_i, \Delta\beta_i), \tag{6}$$

where R_i is the rotation matrix calculated using F_i and F_1, and $\Delta\alpha_i$ and $\Delta\beta_i$ are rotation angles of the pitch and yaw in the x and y axes between P_i and P_1. Then the rotation angles between P_k and P_N can be computed by

$$\begin{cases} \Delta D_x(P_k, P_N) = \Delta\alpha_k - \Delta\alpha_N \\ \Delta D_y(P_k, P_N) = \Delta\beta_k - \Delta\beta_N \end{cases} \tag{7}$$

In this way, we can derive the rotation angles between the *go* key and the other clicked keys.

F. Step 5. Recovering Passwords

We propose a password inference method to recover the user's password using the rotation angles of the clicked keys. Fig. 13 illustrates the keyboard in the Oculus Browser. Since the layout of the virtual keyboard can be learned in advance, we first measure the reference rotation angles between the *go* key and any other key. Denote K_i and K_e as the i^{th} key and the *go* key. Denote $\Delta D_x(K_i, K_g)$ and $\Delta D_y(K_i, K_g)$ as the reference rotation angles between K_i and K_g along the x and y axes of the input device.

After deriving the empirical rotation angles between the *go* key and the other clicked keys (i.e., P_k and P_N) using the rotation data, we infer the key P_k by comparing the reference rotation angles between the *go* key and any key K_i with the empirical ones between P_N and P_k. Since the last key (i.e., P_N) is always the *go* key for completing the password input, we infer the previous tapped keys in terms of calculated rotation angles. We define the similarity between P_k and any reference key K_i as follows,

$$\begin{aligned} S(P_k, K_i) = \ & |\Delta D_x(P_k, P_N) - \Delta D_x(K_i, K_e)| \\ & + |\Delta D_y(P_k, P_N) - \Delta D_y(K_i, K_e)| \end{aligned} . \tag{8}$$

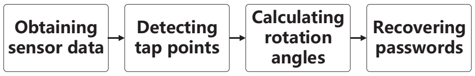


Fig. 14. Workflow of motion sensor-based attack

Then we can derive a candidate key list for each key click P_k based on the similarity score. We choose the top 3 candidates for each P_k and derive 3^{N-1} candidate passwords. We accumulate the similarity of each key in every candidate password and sort the accumulated similarity scores in an ascending order. Therefore, the smallest similarity score corresponds to the best candidate password.

The *success rate* is defined as the probability of recovering entered passwords. Since our attacks produce multiple candidates for a password, we define two kinds of success rate: Top 1 success rate is the probability that the best password candidate is the entered password; Top 3 success rate is the probability that one of the top 3 best password candidates is the user entered password. Top 3 success rate is a reasonable metric because most of the password systems allow a user to input a password 3 times before locking the account.

G. Brute Force Attack

We can perform a brute force password enumeration attack in the case that a *go* key is not clicked. For example, a user may input her password and then click a login button in an app. We consider a sequence of rotation angles between clicked keys as a key clicking trajectory. A brute force attack can be performed by fitting trajectory onto the virtual keyboard. Since the starting position of the trajectory, i.e., the first key of the password, is unknown, we try to fit the trajectory onto the keyboard from upper left to low right of the keyboard so as to derive all possible password candidates. If the password candidate list contains the correct password, this trajectory is referred to as a hit trajectory. To evaluate the brute force attack, we define two metrics, i.e., the *hit rate* and the *average number of candidates* generated by a hit trajectory. The hit rate is the number of hit trajectories divided by the total number of trajectories.

V. Motion Sensor-based Attack

In this section, we first briefly introduce the workflow of the motion sensor-based attack and then provide more details.

A. Workflow of Motion Sensor-based Attack

Fig. 14 illustrates the workflow of the smartphone sensor-based attack. The attack workflow consists of four steps as follows.

Step 1: Obtaining sensor data. A malware is installed on the user's smartphone to record the motion sensor data points (i.e., the orientation data of the headset or the controller in the yaw and pitch dimensions).

Step 2: Detecting click points. We study the user's input pattern and then analyze the orientation data of the headset or the controller so as to detect the key clicks in the orientation data time series. Therefore, we can obtain the orientation angles of clicked keys in the yaw and pitch dimensions.

Step 3. Calculating rotation angles: After deriving the orientation data of the key click points, we obtain the rotation angles by computing the changes of orientation angles between the *go* key and the other clicked keys. The process is similar to estimating rotation angles of the VR headset in the vision attack introduced in Section IV-E.

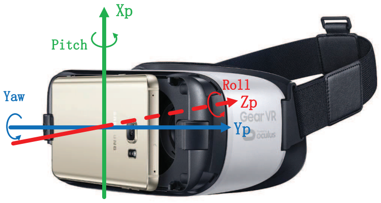


Fig. 15. Smartphone coordinate system

Step 4. Recovering passwords: We exploit the key click points and rotation angles to infer the clicked keys. The strategies used for recovering passwords are similar to those in Step 5 of the vision attack and please refer to Section IV-F.

We introduce Steps 1 to 2 in detail below.

B. Step 1. Obtaining Sensor Data

1) Obtaining sensor data from the VR headset: We exploit the Oculus mobile SDK to read the orientation angles of the Gear VR headset in the yaw and pitch dimensions respectively. The orientation angles are calculated using the data from motion sensors including the accelerometer, gyroscope, and magnetometer in the headset. Fig. 3 illustrates the sensor coordinate system of the headset. When the user opens a VR app and inputs her password, the malware on the smartphone continuously records the orientation angles along the X_h and Y_h axes. Since the virtual ray is cast along the reverse direction of the Z_h axis, the rotation of the Z_h axis cannot change the ray direction. The user can rotate the headset along the X_h and/or Y_h axes of the headset to control the cursor movement on the keyboard. Therefore, we record the yaw, pitch and timestamp of such sensor data. Fig. 15 shows the smartphone's sensor coordinate system.

2) Obtaining sensor data from the VR controller: The malicious app on the smartphone continuously records orientation sensor data from the controller when the user enters her password. Since the virtual ray is along the direction of Z_c axis of the controller as shown in Fig. 6, the user can move the cursor by rotating the controller along the X_c and/or Y_c axes. Therefore, the cursor's positions can be inferred by using the pitch and yaw. Accordingly, we record the yaw, pitch and timestamp of such sensor data, which can be obtained through the Oculus mobile SDK.

C. Step 2. Detecting Click Points

We analyze a user's input patterns and process the orientation data to detect the key click points in the sequence of sensor data. To click a key on the virtual keyboard, the user should first rotate the input device (i.e., the headset or the controller) to move the cursor towards the intended key. Once the cursor is on the key, the user stops moving the input device and presses the touchpad on the input device to enter the key. During the key clicking process, the user keeps the input device steady. As a result, the orientation angular changes of the input device in the yaw and pitch dimensions are almost zero when the user clicks. Based on such observation, we design a sliding time window strategy to detect the key click points. The empirical size of the time window is 400 ms. Denote W as the sliding window, and α_i and β_i as the i^{th} readings of pitch and yaw in the sliding window respectively, where $i \in W$. We aggregate the orientation angular changes in the sliding window by

$$A(W) = \sum_{i \in W} (|\alpha_{i+1} - \alpha_i| + |\beta_{i+1} - \beta_i|) \quad . \tag{9}$$

If $A(W)$ is less than a threshold T_A, the window contains a clicking action. The central point of the window is chosen as the click point.

We process the orientation data in yaw and pitch dimensions of the input device in *reverse* order in order to discover all the key click points. Denote N as the number of clicks on the keyboard and $(P_1, P_2, \ldots, P_N)$ as the sequence of the click points, where P_N is the click position of the *go* key. If the last clicked key is the *go* key on the virtual keyboard, we can process the sensor data in reverse order to locate the first key click, i.e., P_{N-1}, in terms of Equation (9). According to our empirical data, the time interval between two sequential key clicks is at least 1.2 seconds. Therefore, we skip the sensor readings within this time interval and move on to detect the next click points until we find all the click points.

VI. Evaluation

In this section, we introduce the experimental design and results to evaluate our attacks against Gear VR. We also discuss limitations of the side channel attacks at the end of this section.

A. Experimental Design

We attempt to recover random passwords that contain letters and numbers. The password length can be 4, 6 or 8. A Samsung Galaxy S8 is attached to the Gear VR headset. In the computer vision-based attack, a ZED camera is used to record a 3D video of a victim inputting passwords on the Gear VR headset. We do not use the ZED camera against the VR controller since we find it very hard to derive feature points of the controller, let alone track the controller.

In the motion sensor-based attacks, a Gear VR controller is connected to the smartphone by Bluetooth. We develop a VR app to run in the background of the Android system. The orientation data can be directly derived from the headset and the controller using the Oculus mobile SDK [10] and Android SDK, respectively. Recall the accelerometer, gyroscope, and magnetometer are installed in the headset and the controller. The orientation angles retrieved via the SDKs are deliberately calculated using the fused data from these three sensors in the headset and the controller as the VR system requires the accurate and stable tracking of the head or the hand movement via the headset or the controller [7], respectively. For comparison, we also employ the sensors on the smartphone to retrieve the orientation angles using the Android SDK.

We consider the following factors that may affect the success rates of our attacks: users, distance between the camera and victim, and sampling rate of the sensors.

Users: Since different people have different body size, head movement patterns and inputting habits, it is necessary to evaluate the robustness of our attacks in terms of users. In our experiments, two groups of 10 users respectively conducted inputting for vision-based or motion sensor-based attacks. Users were told to use a Gear VR in their preferred way. For the vision-based attack, there were 3 females and 7 males and their average age is 28. For the motion-based attacks, there were 4 females and 6 males and their average age is 23. We asked each user to input 10 random passwords. For each kind of attack, we have 100 password inputs.

Distance: The distance between the stereo camera and the user is taken into account as it affects the vision-based attack. In order to evaluate the impact of the distance on the success rate of attacks, we position the camera in front of the user at a distance from 1 meter to 2 meters.

Sampling rate: This affects motion sensor-based attacks. Sampling rates of the Oculus mobile SDK and Android SDK are different. For the Oculus mobile SDK, the sampling rate is around 60 Hz. We use a sampling rate of 100 Hz on the Samsung smartphone through the Android SDK.

B. Computer Vision-based Attack

TABLE I. SUCCESS RATES OF THE COMPUTER VISION-BASED ATTACK

Password Length	4	6	8
Headset (Top1)	58%	52%	46%
Headset (Top3)	72%	67%	63%

Table I shows the success rates of the computer vision-based attack against Gear VR. To set our baseline, we perform attacks against 10 different users at a distance of 1.5 meters. As shown in the results, the top 3 success rate for 4-character passwords is 72% and the top 3 success rate for 8-character passwords is 63%. The short distance of 1.5m shows the limitation of the vision-based attack.

We also perform attacks at different distances for 8-character passwords, and Fig. 16 shows the results. It can be observed that the success rate drops when the distance increases. This is because as the distance increases, the headset's image size in the frame decreases, and the performance of feature point extraction and object tracking drops.

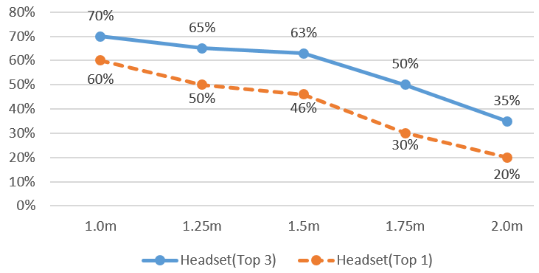


Fig. 16. Success rates of the computer vision-based attack v.s. distance

C. Motion Sensor-based Attacks

Table II gives the top 1 and top 3 success rates of our motion sensor-based attacks using the orientation data from the Oculus mobile SDK. It can be observed that the motion sensor-based attack is effective. The top 3 success rate can reach around 90% for 8-character passwords. The reason why the success rate cannot reach 100% is given as follows. During the key clicking process, a user normally tries to keep the headset static before clicking a key. This is helpful for the detection of a key click point. However, a user may click a key when the controller is moving. This negatively affects the detection of key click points. In our experiments, the success rates of key click detection in the attack against the headset for passwords with length 4, 6, and 8 are 95%, 93%, and 92% respectively. The success rates of key click detection in the attack against the controller are 91%, 90%, and 89% respectively.

Table II also shows the results of the attack using the senors of the smartphone to derive the orientation angles of the smartphone so as to infer the password. Recalled the smartphone is attached to the Gear VR headset. We use the Android SDK to read these orientation angles [1]. However, the well-known gyroscope drift issue of the sensor significantly affects the accuracy of the orientation angle estimation. Since no drift correction algorithm is applied in the Android SDK, the top 1 success rate for 8-character passwords is 30%, while the top 3 success rate is 56%.

TABLE II. SUCCESS RATES OF THE MOTION SENSOR-BASED ATTACKS

	Password Length	4	6	8
Headset sensors	Headset (Top 1)	92%	89%	85%
	Headset (Top 3)	95%	93%	92%
Controller sensors	Controller (Top 1)	84%	80%	71%
	Controller (Top 3)	91%	90%	89%
Smartphone sensors	Headset (Top 1)	42%	36%	30%
	Headset (Top 3)	67%	62%	56%

Table III gives the experimental results of the brute force attack introduced in Section IV-G. The hit rate of the brute force attack is around 90% in the case of 8-character passwords while the attacker may have to try around 10 times. The attack against the Gear VR headset performs better than the attack against the controller. Due to the errors introduced by the key click point detection, the hit rate cannot reach 100% as we discussed before. Moreover, the average number of password candidates per trajectory in the attack against the headset is less than that of the attack against the controller.

TABLE III. HIT RATES AND AVERAGE NUMBER OF CANDIDATES OF THE BRUTE FORCE ATTACK

	Password Length	4	6	8
Headset	Hit rate	95%	93%	91%
	Candidates	16.55	11.65	9.87
Controller	Hit rate	91%	90%	89%
	Candidates	20.91	14.58	11.63

D. Limitations of Side Channel Attacks

We now discuss limitations of the side channel attacks in this paper. We assume a known and fixed keyboard layout in both vision-based and motion sensor-based attacks and infer a clicked key from the rotation angle of the virtual ray between two keys. For mobile banking web pages, the position of username and password input boxes is fixed and the keyboard layout is fixed too. Some apps such as Samsung explorer use a fixed keyboard layout. The attacks in this paper can be directly applied. However, VR apps may relocate the keyboard based on the input box's position and the keyboard's layout changes too while the change might be subtle. Therefore, the rotation angle between two keys changes. If the input box's position and the keyboard layout are unknown, this will affect the accuracy of the key inference if a wrong keyboard layout is used as the reference keyboard in the attacks. For such apps, we may learn how the keyboard's layout changes as its position changes and the person moves the HMD.

VII. Related Work

Given the page limit, we only review papers closely related to VR security. To the best of our knowledge, we are the first to explore side channel attacks against the head-mounted displays (HMD) based VR systems. We exploit the 3D stereo vision-based attack and motion sensor-based attack to derive the orientation of the pointing ray of the VR headset or VR controller so that the orientation can be utilized to infer the clicked keys.

New emerging VR techniques have gradually attracted the attention of attackers. For example, an attacker can exploit the network traffic emitted from the VR system [19] to infer the victim's activities in social applications. However, few research efforts have been made to investigate side channel attacks against VR. Among them, George *et al.* [5] study the security of authentication methods in HMD-based VR systems by evaluating the resistance to shoulder surfing attacks via human eyes, and find that the PIN and pattern-based authentication methods in VR can be more resistant to the attacker than the methods applied to the smartphone in the physical world. Our work shows that these traditional mobile authentication techniques are not secure under the side channel attacks in VR. The VR technique can also be utilized as a tool to hack the face authentication system [18].

Li et al. present a 3D vision attack against smartphones and tablets [8] while there are various 2D vision attacks [15], [20]. They track feature points on a touching hand and infer touched keys by fitting the hand movement topology onto a keyboard. The work in this paper is very different from the techniques in [8] because of the dynamics and properties of the VR headset or VR controller during an the inputting process.

Our motion sensor-based attack is also different from the existing ones [11] because our goal is to infer the orientation of the virtual pointing ray in the virtual environment of VR.

VIII. Conclusion

In this paper, we introduce novel computer vision-based and motion sensor-based side channel attacks to infer the user input such as passwords via a pointing device such as a Samsung Gear VR headset and controller. In the vision attack, a stereo camera is used to record a video of the user inputting with a VR headset and derive the orientation angles of the virtual ray in the virtual environment. Because of the limits of the stereo camera and computer vision techniques, we also introduce the motion sensor-based attack, in which the senors built in the pointing devices and smartphones are exploited to obtain the orientation angles of the virtual ray in the virtual environment. With the orientation angle time series, we can derive the empirical rotation angles and compare them with the reference ones to infer the clicked keys. Extensive real-world experiments were conducted. We can achieve a top 3 success rate of around 90% in motion senor-based attacks while the computer vision-based attack has a top 3 success rate of 63% at 1.5m for 8-character passwords. The advance of stereos cameras will improve the attack distance.

The usability issue often prevents the use of new authentication methods. For example, while a randomized keyboard may defeat the side channel attacks in this paper, we didn't find any apps equipped with a randomized keyboard in the VR environment because of its poor usability. Most bank apps in US use a conventional keyboard. The attacks in the paper will have a long-term impact and usable authentication methods for VR are desired meanwhile.

Acknowledgement

This material is based partially upon work supported by National Key R&D Program of China 2018YFB0803400 and 2017YFB1003000, by the US National Science Foundation (NSF) under Grants 1642124, 1547428 and 1350145, and by the National Natural Science Foundation of China (NSFC) under Grants 61572130, 61532013, and 61632008, by Jiangsu Provincial Key Laboratory of Network and Information Security under grants BM2003201, by Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under grants 93K-9 and by Collaborative Innovation Center of Novel Software Technology and Industrialization. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF or NSFC.

References

[1] Android Open Source Project. Sensor types. https://source.android.com/devices/sensors/sensor-types, 2017.

[2] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.

[3] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, Inc., 2nd edition, 2013.

[4] D. W. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3-d rigid body transformations: A comparison of four major algorithms. *Mach. Vision Appl.*, 9(5-6):272–290, Mar. 1997.

[5] C. George, M. Khamis, E. von Zezschwitz, M. Burger, H. Schmidt, F. Alt, and H. Hussmann. Seamless and secure vr: Adapting and evaluating established authentication systems for virtual reality. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2017.

[6] A. Kharpal. VR will be ḿore ubiquitouśthan smartphones: Oculus. https://www.cnbc.com/2015/11/03/virtual-reality-will-be-more-ubiquitous-than-smartphones-oculus.html, Nov 2015.

[7] S. M. LaValle, A. Yershova, M. Katsev, and M. Antonov. Head tracking for the oculus rift. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[8] Z. Li, Q. Yue, C. Sano, W. Yu, and X. Fu. 3d vision attack against authentication. In *Proceedings of IEEE International Conference on Communications (ICC)*, 2017.

[9] Z. Ling, J. Luo, Q. Chen, Q. Yue, M. Yang, W. Yu, and X. Fu. Secure fingertip mouse for mobile devices. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016.

[10] Oculus. Mobile SDK Getting Started Guide. https://developer.oculus.com/documentation/mobilesdk/latest/concepts/book-intro/, 2018.

[11] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. Accessory: Keystroke inference using accelerometers on smartphones. In *Proceedings of The Thirteenth Workshop on Mobile Computing Systems and Applications (HotMobile)*. ACM, February 2012.

[12] E. Z. Psarakis and G. D. Evangelidis. An enhanced correlation-based method for stereo correspondence with sub-pixel accuracy. In *Proceedings of IEEE ICCV*, 2005.

[13] T. Simonite. Intel and Microsoft are teaming up to make virtual reality ubiquitous. https://www.technologyreview.com/s/602189/intel-and-microsoft-are-teaming-up-to-make-virtual-reality-ubiquitous/, August 2016.

[14] Stereolabs. ZED - depth sensing and camera tracking. https://www.stereolabs.com/zed/specs/, 2017.

[15] J. Sun, X. Jin, Y. Chen, J. Zhang, R. Zhang, and Y. Zhang. Visible: Video-assisted keystroke inference from tablet backside motion. In *Proceedings of the 23rd ISOC Network and Distributed System Security Symposium (NDSS)*, 2016.

[16] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[17] Wikipedia. Samsung Gear VR. https://en.wikipedia.org/wiki/Samsung_Gear_VR, 2018.

[18] Y. Xu, J.-M. Frahm, and F. Monrose. Virtual u: Defeating face liveness detection by building virtual models from your public photos. In *Proceedings of the 25th USENIX Security Symposium (Security)*, 2016.

[19] A. Yarramreddy, P. Gromkowski, and I. Baggili. Forensic analysis of immersive virtual reality social applications: A primary account. In *Proceedings of the 12th International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE)*, 2018.

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