3D Vision Attack against Authentication

Zupei Li*, Qinggang Yue*, Chuta Sano*, Wei Yu†, Xinwen Fu*
*Department of Computer Science
University of Massachusetts Lowell, MA, USA
Email: {zli1, qye, schuta, xinwenfu}@cs.uml.edu
†Department of Computer & Information Sciences
Towson University, MD, USA
Email: wyu@towson.edu

***Abstract*—In this paper, we introduce a computer vision-based attack using stereo cameras against authentication approaches for touch-enabled devices. In the attack, an attacker uses a stereo camera (such as one on the HTC Evo 3D smartphone) and takes a video of a victim entering passwords on the touch screen of the victim's mobile device. We focus on challenging scenarios where the victim holds the device up and the attacker cannot see the victim's fingertip or the device screen. Since the stereo camera provides depth and distance information of objects in video frames, we can build a 3D scene to analyze the victim's hand movement and automatically recover the victim's passcode. The 3D vision attack is stealthy in daily settings like a classroom or a coffee shop since the attacker does not need to take a suspicious angle and see the touch screen of the victim. Without loss of generality, we use graphical passwords as an example and perform extensive experiments to demonstrate the effectiveness of the attack. The success rate of the 3D vision attack reaches 90% when the camera is across a table from a victim in a typical gathering scene.**

I. Introduction

As hardware and software advance, stereo cameras have been gaining more attention on smart devices. CNET claimed "The future of smartphones is in dual cameras" in February 2016. In September 2016, Apple released iPhone 7 plus with dual cameras, which is capable of obtaining depth of field. Before iPhone 7 plus, HTC, LG and Sharp released their smart phones with stereo cameras in 2011. These stereo cameras can be leveraged to implement various 3D special effects such as 3D videos and a taste of DSLR-style photography.

However, stereo cameras may be abused. In this paper, we introduce a novel 3D computer vision-based attack against graphical passwords on touch-enabled devices. Our attack takes a realistic and generic threat model: a stereo camera captures the video of the graphical password inputting process, but cannot capture the device's screen, as shown in Figure 1. The basic idea is to reconstruct the finger movement and touch screen in the 3D space, map the finger trajectory onto the touch screen plane and then fit the trajectory onto the keypad.

The major contribution of this paper is summarized as follows. To the best of our knowledge, we are the first to attack graphical passwords in scenarios where the inputting *fingertip* is occluded. We are the first to use stereo camera systems to attack touch-enabled devices. To validate this attack, we have performed extensive experiments with different attack devices against different victim target devices. The attack devices include a self-built Logitech C920 stereo camera system [5] and an HTC EVO 3D phone [3]. The victim target devices include a Nexus 7 tablet and a Nexus 6P smartphone. When the distance *between the stereo camera and the victim device* is 1 meter, both the Logitech stereo camera system and HTC EVO 3D can achieve a success rate of 86% or better against both the tablet and smartphone. The Logitech stereo camera system can reach a success rate of 80% at 1.5 meters and 60% at 2.0 meters. Please note: the face-to-face distance between the attacker and victim is longer than the distance between the attacking camera and the victim device. The distance we consider resembles the scenarios of classrooms, conferences, cafe shops and other gatherings where we always see people holding up their phones. Therefore, the 3D vision attack is realistic. It is also generic, does not need unrealistic training and can be applied to various other scenarios.

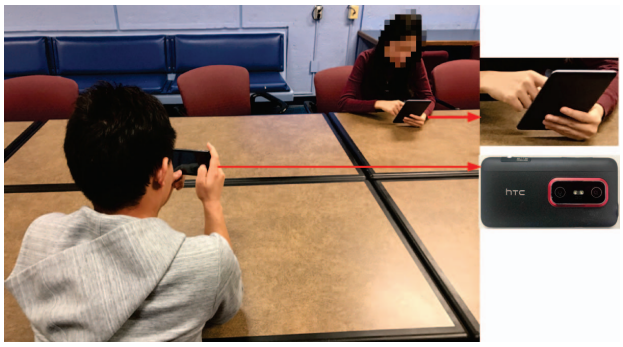

Fig. 1. Experiment Scene

We can divide vision based attacks against touch inputting into four groups. In the first group, the text is visible, although maybe blurry, in a recorded video [1]. In the other three groups, the text is not visible. In the second group, an attacker is able to capture the inputting fingertip and the popup or magnification of keys in the video [6]. In the third group, there is no pop-up in the video while the inputting fingertip is visible [9]–[11]. In the fourth group, the threat model assumes the inputting fingertip may not be visible, but parts of the hand are visible in captured videos [7]. Our attack is in the fourth group of attacks. Compared with related work, the 3D vision attack strategy is generic. The training based strategy in [7] cannot be directly used to attack graphical passwords since the finger dynamics is different. For example, the touching finger is never lifted up during the inputting process.

The rest of this paper is organized as follows. In Section II, we present the stereo camera-based attack, including the threat model, the basic idea, and the step by step workflow of our system. In Section III, we provide the experiment design and results to demonstrate the feasibility of the 3D vision attack. We conclude this paper in Section IV.

II. Attack Process

In this section, we first introduce the threat model. We then present the basic idea of the investigated stereo camera attack and its workflow. At last we introduce each step in detail.

A. Threat Model

In the 3D vision attack, an attacker is able to use a stereo camera and take videos of users inputting their graphical passwords. Although we use graphical passwords as an example to demonstrate the 3D vision attack, our attack is generic and can be applied in other scenarios, for example, while a victim inputs mobile banking account or online shopping account passwords. Since the body of a victim often blocks the view of the attacker's camera, an attacker may have to take the video in front of the victim. Because a user often holds up her device while inputting on the touch screen and the device blocks the view of the fingertip, the inputting fingertip may not be visible in the video. Our attack is designed for these challenging scenarios. This type of attack is stealthy given the fact that holding up a device is a common phenomenon. The attacker can just hold up her phone with a stereo camera and record videos. We assume the inputting fingertip is visible at the start or end or some point of the inputting process in a recorded video.

B. Basic Idea

The basic idea of the 3D vision attack is to use a stereo camera, take a 3D video and reconstruct the 3D trajectory of the inputting hand and fingertip. We then design algorithms fitting the trajectory onto a reference keyboard in order to recover the inputs, considering the limited size of the software keyboard. In this study, we use the graphical password as an example to demonstrate the idea of the 3D vision attack although it is very generic.

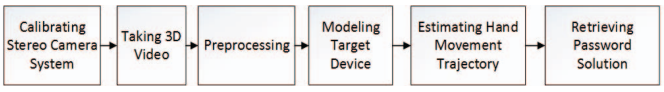

Fig. 2. Workflow of Stereo Camera Attack

Figure 2 gives the workflow, which consists of 6 steps. We now discuss these 6 steps in detail below.

C. Step 1: Calibrating Stereo Camera System

To build the 3D world coordinate system and reconstruct the 3D model of the inputting device and password inputting process as indicated by Figure 3, we need to know the parameters of the stereo camera system, including the camera intrinsic matrix **I**, the camera distortion coefficients **D**, and the geometrical relationship between two left and right cameras

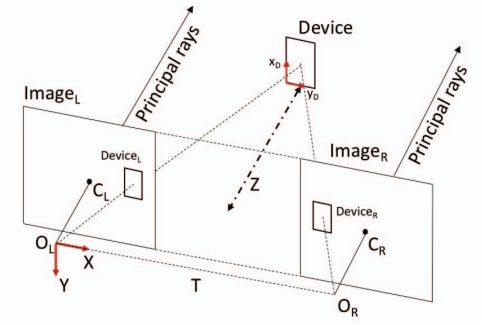

Fig. 3. Stereo Camera System

[2]. Such relationship is represented by a rotation matrix **R** and translation vector $\bar{\mathbf{T}}$. The intrinsic matrix **I** includes camera's focal length, the principal point offset and the axis skew of the camera. The distortion coefficient **D** contains the parameters that describe the camera's radial distortion and tangential distortion.

We perform stereo calibration to obtain these parameters introduced above. We use the camera system to take several photos of a chessboard with side length of 23.5mm from different poses. The calibration employs the corresponding points of the chessboard corners in all the photos for the calculation. The accuracy of the calibration is affected by the quantity and quality of the chessboard photos. In general, more than 10 photos are needed at different poses. In order to obtain accurate and stable results, we disable the camera auto-focus function and set the focus range of the two cameras to infinity.

Figure 3 shows the image formation process of the stereo camera system. O_L and O_R are the projection centers of two cameras. $\bar{\mathbf{T}}$ is the translation relationship between the two cameras. C_L and C_R are the principal points of two cameras. $Device_L$ and $Device_R$ are the object's (in this case, $Device$) in the left and right images $Image_L$ and $Image_R$ taken by the two cameras. After the calibration of the stereo camera system, we can build the 3D coordinate system, with the origin at the left camera's lens center. As shown in Figure 3, the origin is at O_L.

D. Step 2: Taking Stereo Videos

In this step, the attacker uses the stereo camera and takes videos of a victim performing inputs. There are various factors affecting the quality of the video thus the success of the attack, including the distance between the attacking camera and the target device, the environment lighting and the attacking angle. The distance between the attacking camera and the victim device is a key factor that affects the accuracy, because most of the stereo camera systems equipped on smart devices are built with two wide angle cameras with very short camera focal length. These cameras generally do not have the optical zooming function. Therefore when the adversary is far away from the victim, the victim's hand and device in the image will be very small. This affects the 3D reconstruction accuracy and thus the attack performance.

The frame rate (frames per second, denoted as FPS) affects the result of the attack too. For a graphical password, users

can usually finish the input process in one or two seconds. According to the Nyquist sampling theory, the sampling rate (frame rate) of the attack must be high enough to capture the movement of the victim's finger/forearm [4]. Particularly, for stereo camera systems, there are actually two cameras working simultaneously. If the recording resolution of the two cameras keeps the original resolution of each camera, the load on the data bus and the need of storage will increase. For the devices we have, it is not likely that both the original frame rate as well as the image resolution can be kept in the stereo camera mode. The resolution is often compromised to make the frame rate high enough to get a decent sampling rate for capturing the movement details. As hardware and software advance, we expect improving FPS and resolution of future stereo cameras on smart devices and the 3D vision attack will be more powerful in the near future.

The attacker also needs to adjust the shooting angle and make the device's back and the inputting hand (or part of the hand) in the Field of View (FOV) of both cameras. Therefore, we can perform the 3D reconstruction of touch-inputting on a device.

E. Step 3: Preprocessing

First, a raw video from Step 2 is often a long video clip with unnecessary content. We crop the video and keep only the part when the victim is inputting passwords. Cropping the video will reduce the workload of later steps.

Second, we apply rectification to align the videos. Stereo rectification mathematically eliminates the rotation between two cameras and aligns two cameras to one view plane. Axises of left and right cameras will be aligned. We use Bouguet's algorithm [2] with the calibration results obtained in Step 1 to rectify video frames. The rectification produces a reprojection matrix $\mathbf{Q}$ [2],

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

where c_x and c_y are the coordinates of the principal point, f is the focal length of the left camera, and T_x is the translation parameter of the x axis.

F. Step 4: Modeling Target Device

To reconstruct the 3D scene of the password inputting process on the device, the 3D model of the target device should be built first. In this step, we first calculate the 3D coordinates of the target device and then derive its keyboard layout in the 3D world.

1) Computing the 3D Coordinates of Device Corners: Since computing the 3D coordinate of every pixel is time consuming and infeasible sometimes, we only compute the 3D coordinates of the four corners of the device and will derive the keyboard layout using the physical location relationship between the keys and the device corners. To calculate the 3D coordinate of a specific point, the reconstruction algorithm needs the 2D coordinates of the corresponding points in the left and right images and their disparity d, which is the horizontal x coordinate difference.

We detect the device's corner points in the left image and then find their matching points in the right image by template matching algorithms. To find the corner in the left image, we first detect the four edge lines of the device and compute the intersection of those lines. The corners are the intersection of the four edges. We then apply the template matching algorithm to find the corresponding points in the right image. Our algorithm achieves the sub-pixel accuracy, which is necessary for deriving accurate 3D coordinates. It works as follows. Since we know the geometrical relationship of the two cameras, we can estimate the position of the corresponding points in the right image and obtain a searching window. Given the point in the left image and the searching window, we first enlarge the two areas by the bi-cubic 2-D interpolation algorithm. Then we compute the normalized 2D cross-correlation in the searching window of the right image. The position where the maximum correlation is achieved is the location of interest.

After getting the corresponding pair of points, we can derive the disparity (d) of a point pair and calculate the 3D coordinate of the point through the following equation:

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

where (x, y) is the point's 2D coordinate in $Image_L$, and $d = x - x_R$ is the disparity of this point and its corresponding point (x_R, y_R) in $Image_R$. $(X/W, Y/W, Z/W)$ is the point's 3D coordinate. With Equation (2), we can derive the 3D coordinates of the four device corners, denoted as a set Cr_{ori}.

2) Deriving the 3D Keyboard Layout: To accurately model the target device's geometric characteristics, we measure the physical device and build a reference 3D model of the target device and its keyboard layout. The model contains 4 corner points of the device, denoted as a set Cr_{ref}, and the keyboard layout. The surface of the device (and keyboard) aligns with the XOY plane. Figure 4 shows the reference keyboard model for Nexus 7.

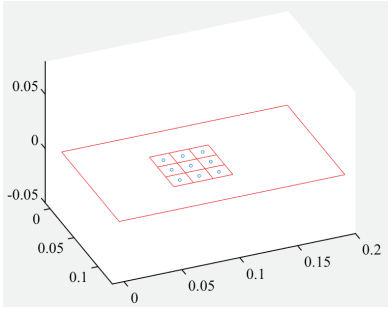

Fig. 4. Reference Keyboard for Nexus 7

The reference keyboard model will be used to correct the derived coordinates of the four corners of the device from a video because of various errors such as those from

computer vision algorithms. We first fit the four corners onto the same plane, which has the minimum average distance to the four corner points. We then perform the 3D point warping between Cr_{ori} and Cr_{ref}. Therefore, we calculate the keyboard position P_{key} in the 3D coordinate system.

G. Step 5: Estimating Hand Movement Trajectory

Under our threat model, the victim's inputting fingertip is not visible in the captured video, as shown in Figure 1. However, we can study the geometric relationship between the inputting fingertip movement and the movement of other parts of the hand, and infer the possible inputting fingertip moving trajectory.

To estimate the hand movement, we first track the movement of feature points on the hand by the optical flow [8] in the 2D video frames. The optical flow tracks the points by estimating the similarity of a small area around points of interest. However, due to the camera angle and the movement of the hand, feature points may be lost in a video since lighting changes and visible parts of the hand may become invisible in the video because of the movement. We pick up feature points that are persistent through the video, The points marked by green crosses in Figure 5 are the feature point we use to track the victim's hand during the inputting process. We choose the most stable one of these feature points based on the accumulative optical flow scores.

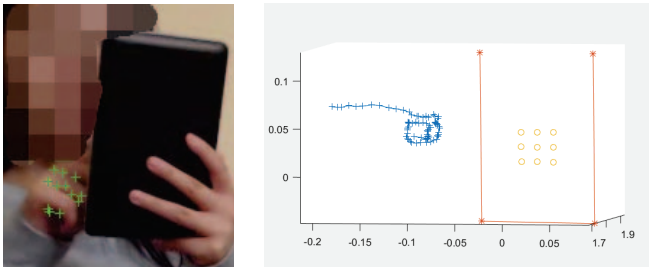

Fig. 5. Tracking Feature Point on Hand and it's 3D Trajectory

After getting the 2D motion trajectory, we would derive their 3D coordinates by Equation (2) from Step 4. This 3D trajectory is called the preliminary trajectory J_{pre}. Figure 5 shows the hand movement in the 3D world from one example in our experiments.

H. Step 6: Finding Password Candidates

In this step we analyze the movement of different parts of the hand. We design algorithms to estimate the inputting fingertip movement trajectory from the hand movement trajectory J_{pre}. Then we derive the graphical password candidates.

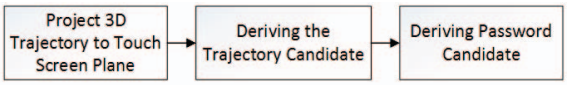

Fig. 6. Workflow of Finding Password Candidates

1) Projecting the 3D trajectory to the Touch Screen Plane: To reduce the complexity, we project the trajectory of the chosen feature point onto the device plane. Since the shape of the hand is relatively fixed during the touching process, we may estimate the trajectory of the touching fingertip from the projected trajectory of the feature point of the hand.

Fig. 7. Spacial Relation Between Inputting Fingertip and Hand

Since we assume the inputting fingertip is visible at some points of the video (e.g. at the start or end of the inputting process) as shown in the left figure of Fig. 7. It can be observed that during the inputting process, the user's hand almost keeps the same gesture. Therefore, we can derive the geometric relationship between the inputting fingertip and the feature point of the hand, as shown in Fig. 7. If we assume that a user's hand keeps the same gesture during the inputting process, this translation relationship keeps the same during the inputting process. The motion trajectory of the inputting fingertip can be inferred by the following equation:

$$J_{finger} = J_{pre} + \bar{\mathbf{T}}, \tag{3}$$

where $\bar{\mathbf{T}} = Tp_{finger} - Tp_{origin}$ is the translation vector, calculated from a frame where the fingertip is visible as shown in Fig. 7. J_{pre} is the feature point of the inputting hand. The right figure of Fig. 7 also shows we use $\bar{\mathbf{T}}$ to estimate the first touched point. We project J_{finger} on device screen plane, defined the trajectory as J_{proj}.

2) Deriving the Candidate Shapes of the Fingertip Trajectory: In this step, we analyze the motion relationship between the inputting fingertip and feature point of the hand, and derive the possible password candidates from J_{proj} derived above.

Fig. 8. Bending effect in inputting process

We find that humans tend to bend their fingertip as they input, as shown in Figure 8. Bending makes the trajectory of the feature point different from the actual trajectory of the inputting fingertip. After a careful study of the inputting process on the touch screen and the hand movement, we have the following observations:

- Due to the physical human characteristics, the inputting fingertip creates a larger trajectory than other parts of the hand. Apparently, the wrist creates the smallest trajectory. Therefore, the trajectory of the feature point is different from the trajectory of the fingertip.

- When people touch and input on the touch screen, their palm is roughly in parallel with the touch screen surface. The inputting fingertip may bend vertically toward the touch screen, but would not swipe horizontally when the wrist does not rest on the touch screen. Therefore, we assume that the fingertip's trajectory is vertically enlarged compared with the feature point's trajectory. Since we cannot see the fingertip during the inputting process, the change of the fingertip gesture introduces extra errors.

We design a compensation algorithm to correct errors caused by the difference between the trajectory of the feature point and the trajectory of the fingertip in Algorithm 1. In Algorithm 1, we enumerate all the possible trajectory heights ($JHeight$), in term of number of rows of keys on the keyboard. Recall that vertically the fingertip's trajectory is enlarged compared with the feature point's trajectory. We enlarge different parts of the trajectory based on an empirical formula $GenDev(.)$ of enlarging coefficients. The input of $GenDev(.)$ are inputting hand position P_{hand}, keyboard center C_{key} and the enumerated trajectory height $JHeight$. After we got the $Deviations$, we use the $amplifier$ function to enlarge J_{proj} to get our possible trajectory shape set J_{can}. J_{can} is the output of the Algorithm 1.

Algorithm 1: Compensation Algorithm

Input : $J_{proj}, P_{Hand}, C_{key}$
Output: J_{can}

$J_{can} = []$;
for $i = 0 : 2$ **do**
  $JHeight = KeyInterval * i$;
  $Deviations = GenDev(P_{Hand}, C_{key}, JHeight)$;
  $Result = Amplifier(Deviations, J_{proj})$;
  $J_{can} = J_{can} + Result$;
end

3) Deriving Password Candidate: J_{can} from last step is the set of possible fingertip trajectory shapes. We now derive the possible positions of the fingertip trajectory. For Android graphical passwords, the strokes must pass through at least 4 keys and each key can be used only once. The smallest password trajectory is a 2 by 2 square containing a set of 4 keys. In such a case, there are 4 possible trajectory position candidates. This means in the worst case we need to try 4 times for one trajectory. If the trajectory occupies almost the whole keyboard, there is only one possible position for the trajectory. The size of the trajectory limits the number of possible candidates. To determine a key that the trajectory passes through, we check if the trajectory passes through a circle whose center is the key. The radius of the circle is 1/3 of the distance between two adjacent keys. The radius is derived from our experience using Android graphical passwords.

Figure 9 is one example of enumerating the possible trajectory positions. In this example, there are two possible trajectory shapes in J_{can}, marked as $J_{can}I$ and $J_{can}II$. For $J_{can}I$, we move the trajectory vertically and see if it fits within the keyboard. As a result, we have 2 possible estimated positions, marked as A and B. For $J_{can}II$, there is only one possible estimated position C due to the size of the trajectory.

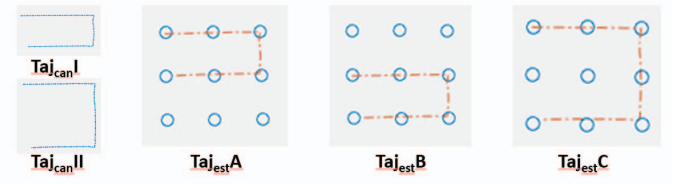

Fig. 9. Possible Trajectories and Solutions

Now we have derived all the graphical password trajectory candidates, we need to rank them. As we discussed in Section II-H1, since the inputting fingertip is partially visible in the video, the estimated location of the fingertip can help rank the password candidates since the trajectory is most likely around the fingertip position.

III. Evaluation

This section introduces the experiment setup and results.

A. Evaluation Setup

We use two stereo camera systems and two target devices in our experiments. The two different camera systems are a stereo camera system built by 2 Logitech C920 webcams and a HTC EVO 3D smart phone equipped with a stereo camera. Our target devices are a large Asus Nexus 7 tablet and a small Huawei Nexus 6P smartphone. The graphical passwords were randomly generated.

The Logitech webcam is supported by openCV. We use a computer to drive two cameras and take videos simultaneously. We disable the auto focus function to make sure that the camera parameters do not change while a video is taken. For the HTC EVO 3D, we use the stock camera app recording 3D videos. The phone stores the videos in the mp4 format, where both left and right cameras images are saved side-by-side. An entire image has 1280 $\times$ 720 pixels. This means the size of the image taken by one camera is 640 $\times$ 720.

In addition to different cameras and different target devices, we also consider other factors in our experiment design such as users, distance between the camera and target.

- **Users**: Since different people have different hand size, finger shapes and inputting gesture habits, it is necessary to study the robustness of the 3D vision attack. We recruited three male and two female participants in our experiments. The average age of the participants is 27. For each data point, each person performs three graphical password inputs so that we have 15 video clips. In the experiments, users were told to use the phone in their own manner in order for us to observe different natural hand gestures from different users.
- **Distance**: To test how the distance affects the attack accuracy, we position the stereo camera system in front of the victim from different distances.

B. Results

We define a successful attack as follows: Recall the attacker derives the password candidates from a video using our 3D vision attack. If any of the top 3 password candidates match the real password, it is a success. The success rate is the number of successful attacks over the number of tested passwords.

Figure 10 shows the effectiveness of the 3D vision attack.

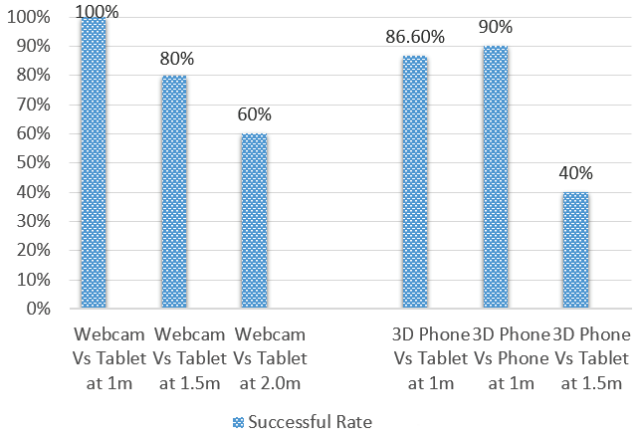

Fig. 10. Success Rate Comparison

To set our base line, we use the webcam stereo camera system and HTC EVO 3D phone to record videos of 5 different users from 1 meter away. It can be observed from Figure 10 that the video quality difference and the interpupillary difference between these 2 camera systems affect the success rate. For both stereo camera systems, we can get a success rate of about 90%.

To validate the attack against different target devices, we use the HTC EVO 3D phone to attack the tablet and smartphone. We find that the success rates are approximately the same for these 2 different target devices. This is because although the device size is different, the actual keyboard size on these 2 devices is similar.

We use both Logitech webcam and HTC EVO 3D to perform the attack upon ASUS Nexus 7 tablet from different distances. For both camera systems, the success rate reduces as the distance increases. For the HTC EVO 3D, it can be observed that when the distance increases, the success rate decreases very much. At the distance of 1.5m the success rate is lower than 50%. This is because at such a distance the target in a video is small and blurry and it is hard to match feature points in left and right camera images.

Figure 11 shows the success rate in terms of the allowed password input attempts. It can be observed that as the number of attempts increases, the success rate increases.

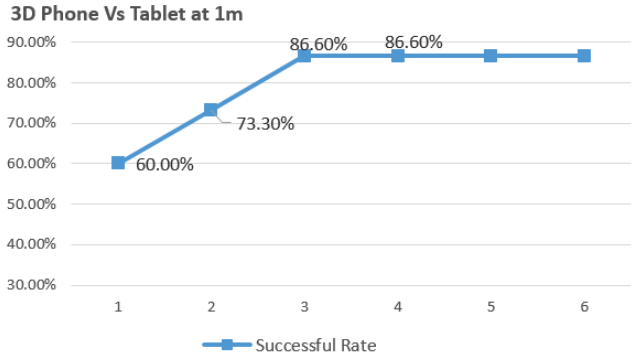

Fig. 11. Attempts Vs Success Rate

IV. Conclusion

In this paper, we present a side channel attack using stereoscopic cameras against authentication strategies on mobile devices. By taking a 3D video of a victim inputting passwords on a device, we can build a 3D model of the inputting hand and the target device. We analyze the geometrical relationship between the inputting fingertip and the visible parts of the hand in the video, and estimate the inputting fingertip movement from the movement of visible parts of the hand. We design algorithms fitting the fingertip trajectory to a reference keyboard and derive the password candidates. We use graphical passwords as an example to demonstrate the effectiveness of the 3D vision attack. Our experiments show that if a fingertip is visible at some points of the video, the success rate can reach 86.6% or better in all investigated cases of attacking stereo cameras including against target devices when the attacking camera is 1 meter from the target device. At 1.5 meters, the Logitech stereo camera system can reach a success rate of 80%. This is the first 3D vision attack against authentication.

References

[1] M. Backes, M. Dürmuth, and D. Unruh. Compromising reflections-or-how to read lcd monitors around the corner. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pages 158–169, 2008.

[2] G. R. Bradski and A. Kaehler. *Learning opencv, 1st edition*. O'Reilly Media, Inc., first edition, 2008.

[3] GSMArena. Htc evo 3d. http://www.gsmarena.com/htc_evo_3d-3901.php, 2011.

[4] Z. Ling. Secure fingertip mouse for mobile devices. In *IEEE: Infocom 2016*, 2016.

[5] Logitech. Logitech c920 hd pro webcam. http://www.logitech.com/en-us/product/hd-pro-webcam-c920.

[6] R. Raguram, A. M. White, D. Goswami, F. Monrose, and J.-M. Frahm. ispy: Automatic reconstruction of typed input from compromising reflections. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 527–536, 2011.

[7] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha. Beware, your hands reveal your secrets! In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 904–917, 2014.

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

[9] 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 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1403–1414, 2014.

[10] Q. Yue, Z. Ling, X. Fu, B. Liu, W. Yu, and W. Zhao. My google glass sees your passwords! In *Black Hat USA*, 2014.

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