Sensor-Assisted Facial Recognition: An Enhanced Biometric Authentication System for Smartphones

Shaxun Chen
Department of Computer Science, University of California Davis, CA, 95616 USA
Phone: 001-530-574-5866
sxch@ucdavis.edu

Amit Pande
Department of Computer Science, University of California Davis, CA, 95616 USA
Phone: 001-530-752-0870
pande@ucdavis.edu

Prasant Mohapatra
Department of Computer Science, University of California Davis, CA, 95616 USA
Phone: 001-530-754-8016
prasant@cs.ucdavis.edu

ABSTRACT

Facial recognition is a popular biometric authentication technique, but it is rarely used in practice for device unlock or website / app login in smartphones, although most of them are equipped with a front-facing camera. Security issues (e.g. 2D media attack and virtual camera attack) and ease of use are two important factors that impede the prevalence of facial authentication in mobile devices. In this paper, we propose a new sensor-assisted facial authentication method to overcome these limitations. Our system uses motion and light sensors to defend against 2D media attacks and virtual camera attacks without the penalty of authentication speed. We conduct experiments to validate our method. Results show 95-97% detection rate and 2-3% false alarm rate over 450 trials in real-settings, indicating high security obtained by the scheme ten times faster than existing 3D facial authentications (3 seconds compared to 30 seconds).

Categories and Subject Descriptors

Security and privacy>software and application security>domain-specific security and privacy architectures

Security and privacy>Security services>authentication

General Terms

Algorithms, Security, Verification.

Keywords

Biometric, mobile devices, user authentication, mobile sensors

I. INTRODUCTION

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Mobisys'14
Copyright 2014 ACM …

Biometric authentication has many advantages over the traditional credential-based authentication mechanism. It is widely considered to be more secure, because it is based on "who the user is" and biometric information is difficult to forge or spoof. On the contrary, credential-based authentication relies on "what the user knows", which can be lost or stolen and more likely to result in identity theft. If dictionary words are used as password, the risk becomes even higher. In addition, biometric authentication is much easier to use. Users do not need to remember a list of passwords for various websites and apps. Of course, devices or browsers can remember the username and password for users. While eliminating users' memory burden, it introduces additional security threats. For example, when users leave their smartphones or tablets on the desk and leave the office for a while, others can easily get access to their social networks or even bank accounts if the password is memorized. Mobile industry also has paid more and more attention to biometric authentications. For example, Apple launched its latest iPhone with fingerprint readers.

Among various biometric authentication methods, facial recognition is a popular technique that has been continuously improving for the past decade. The recognition accuracy is high for practical use [1] [2]. It is recently reported that facial recognition has already been used for commercial payment systems which requires very high accuracy [16]. On the other hand, most of today's smartphones and tablets are equipped with a front facing camera, and the resolution is now typically higher than one mega pixels, which is very handy and able to capture users' face in high quality. Based on these facts, it seems that facial recognition can be widely used for device unlock and website / app login in smartphones.

However, in practice, facial authentication is rarely used in smartphones even if the device has a front facing camera. Android provides face recognition function (to unlock the device) starting from version 4.0, but not many users are using it. For the login of social networks, forums, online services and other apps, credential-based authentication is still dominating other methods. Since biometric authentication has great advantages, what is the cause of

the current status? Apart from the privacy concerns, the following two factors are very important, which impede the prevalence of facial authentication in mobile devices.

First, there is a trade-off between security and ease of use. The simple 2D face recognition, such as the one used in Android 4.0, can be easily fooled by a photograph of the user (referred to as *photo attack*), which is not difficult to obtain in social networks. An improved version requires the user to blink their eyes during the authentication (e.g. Android 4.1 and [3]), but it still can be circumvented by photo editing [4] or playing a clip of video (referred to as *video attack*) [5]. Apparently, these schemes are not secure enough for serious authentication service. More sophisticated 3D facial authentication techniques have been developed to achieve higher security [6] [7]. A practical example is Toshiba Face Recognition Utility used in recent Toshiba laptops. It requires that users turn their heads towards four directions according to a sequence of arrows shown on the screen. In this way, the authentication program is able to differentiate a real 3D face from a flat photo. However, the whole authentication process takes approximately 30 seconds, which is too long and compromises the ease of use (one of the important advantages of biometric authentication). Since it is more complex than entering a password, why would users choose it? In short, current facial authentication schemes cannot achieve *2D-media-attack*-safety and the ease of use simultaneously (*photo attack* and *video attack* together are referred to *2D-media attack*).

Second, the availability of *virtual cameras*. Virtual camera is a category of software which adds a layer between the real physical camera and the operating system. Virtual Webcam, ManyCam, Magic Camera, etc. are all examples of such software. Currently most of them are developed for PC or tablet platforms, but it is easy to migrate to smartphones. The purpose of these software is to add dynamic effects to the webcam, making the video look more beautiful and live chat more interesting. However, they have now become so powerful that they can not only modify the face, hair and backgrounds, but also stream a pre-recorded video, making the operating system believe it is captured by the physical webcam in real time [8]. Therefore, despite their original purpose, virtual camera software in fact puts a serious threat on the security of facial authentication. While the device unlock lies in the system level and has more control of the hardware (so it is less likely to suffer from *virtual camera attacks*), website and app logins are very susceptible to this type of attack if they choose to use facial authentication.

We want to point out that the security issues we mentioned above are not the weakness of biometric information itself; instead, they stem from the methodology people currently use.

In this paper, our goal is to develop a new facial authentication method which is not only more secure but also easy and quick to use. This method should be safe for *2D media attacks* and *virtual camera attacks*; at the same time, the authentication speed needs to be comparable to or faster than credential-based methods (and much faster than the existing 3D facial authentications). If such a method is achievable, it will make facial authentication more practical and useful in smartphones. It will have the potential to become a practically better solution than credential-based authentication methods, and change the current state of human-device interaction.

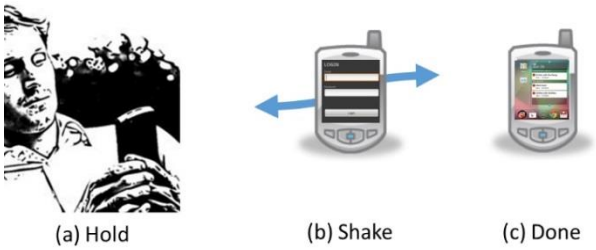


Figure 1. Proposed sensor-assisted user authentication scheme

To achieve this goal, we propose a sensor-assisted facial authentication system. In our method, motion sensors are employed to infer the position and orientation of the front camera. User head movement is replaced by a small movement of the cellphone, which is able to ensure a real 3D face, and results in a much easier operation and faster authentication. Figure 1 gives an overview of proposed scheme, which is simple and intuitive. In addition, the shake of the video and the shake of the smartphone are extracted separately and then compared to each other, in order to defend against *virtual camera attacks*. To the best of our knowledge, this is the first work to enhance the security and usability of facial authentication by utilizing motion sensors of smartphones. The main contributions of this work are as follows:

1. We propose a sensor-assisted facial authentication system for smartphone users. A Nose Angle Detection algorithm is proposed to counter *2D media attacks*.
2. We propose a Motion-Vector Correlation algorithm to counter *virtual camera attacks*.
3. We implement the proposed algorithms on a Galaxy Nexus smartphone with Android 4.2.2 on top of existing face recognition application. Our system is able to achieve very high detection rate for both *2D media attacks* and *virtual camera attacks*. The average authentication time of our method is approximately 2 seconds, which is about ten times faster than the existing 3D facial authentication methods.

The remainder of this paper is organized as follows. Section II discusses related work. Section III outlines the overall picture and describes the attack model of our work. Section IV introduces our method that defends against *2D*

media attack while still conserving the ease of use. Section V presents our technique dealing with the threats of virtual cameras. Section VI evaluates our method and Section VII discusses related issues. Section VIII concludes the paper.

II. RELATED WORK

Face recognition can be divided into two categories: verification and identification [2]. Face verification is a one to one match that compares a face to a template, whose identity is being claimed. Face identification, instead, is a one to *N* problem that compares a face to all the templates in a face database to determine the identity of the face being queried. Apparently, facial authentication is closely related to the former category.

2D media attack has been a well-known but difficult problem for a long time. The existing efforts can be categorized into three groups. *Liveness detection* tries to capture spontaneous eye blinks or lip movements [3] [9]. While it is useful for photo attacks, it cannot deal with recorded videos. Besides eye-blinks, the authors in [9] also use scene context, which matches background image to a template of background to check for forged video playback. It is effective against imposters but not safe against replay attacks. Moreover, the assumption of fixed camera is not valid for smartphones. *Texture analysis* relies on the different texture patterns of a photo or a real object. It can be further divided into two subcategories. The first focus on the printing failures or overall image blur of the printed photo [10]. However, this method will fail if the photo is of high quality. The second detects micro-textures presented on the printer paper [11], but it requires the input has an extremely high resolution, which is not applicable for front facing cameras on smartphones. *Motion analysis* explores one or multiple images, trying to find clues generated by the planar counterfeits. For example, Tan et al. [12] investigated the Lambertian reflectance information, in order to differentiate 3D faces from 2D images. Bao et al. [13] compared the difference of the optical flow fields under various conditions for the same purpose. These methods require high-quality inputs and the computational complexity is very high. Optical flow computations, for example, are 10000 times slower than regular pixel level operations on a video frame [24]. Besides, their performance largely depends on the complexity of the ambient illumination.

3D face recognition has been widely studied in the recent years [14] [15]. It compares the input to a pre-built 3D head template instead of 2D templates. If the face capturing in each authentication trial is also 3D, this approach is robust to *2D media attacks*. However, 3D face recognition is still in its infancy. Sandbach et al. [21] presented a survey on 3D face expression recognition and concluded that the high frame rate requirements and frame resolution limit its practical usage. More importantly, the 3D capturing process is much more time consuming than 2D methods or entering a password, therefore it is not suitable for device unlock or app login. We will further discuss it in the next section.

There have been some studies on extracting non-intentional motions from videos. For MPEG and MPEG2, motion vectors can be directly used to estimate the motion between frames, but it is not a generally applicable method for arbitrary video formats. Deshaker [17] calculates the shift that can achieve maximum likelihood between frames in a top-down fashion. Vella et al. [18] separated foreground from background before movement estimation. Xu et al. [19] analyzed video motions based on extracted feature points. All these works aimed at improving video quality instead of security purpose, and they were not optimized for facial authentication videos which have their own unique characteristics.

Biggio et al. [23] proposed a multi-modal technique where outputs of fingerprint and face recognition techniques are fused together. [22, 24] presented a magnetic user authentication system to avoid both credentials and images to authenticate a user. It requires users to hold a magnetic pen to draw 3D signature in air which are then authenticated using the magnetic sensor in the smartphone. Unlike those approaches that rely on independent processing of different sensors, our approach relies on integrated processing of accelerometers and camera readings.

III. METHOD OVERVIEW

III.A Sensor Assisted Facial Authentication

Today's smartphones are typically equipped with a plethora of sensors, some of which are not fully utilized. Motion sensors (accelerometers and gyroscopes), proximity sensor, digital compass and ambient light sensor have become de facto equipment. Microphone can be viewed as a sound sensor, and sometimes GPS is referred to as the positioning sensor. Traditional facial authentications only use cameras to capture the user's face image / video and then compare it with pre-known templates. In our proposed sensor-assisted facial authentication, we additionally utilize sensors in smartphones, in order to enhance the security or / and improve the performance of the authentication process.

In this paper, in addition to the video camera and existing face recognition schemes, we use motion sensors and ambient light sensor to defend against *2D media attacks* and *virtual camera attacks*. Motion sensors are used to intelligently choose the necessary frames in the video for further processing, while the ambient light sensor triggers screen brightness adjustment to improve the authentication accuracy.

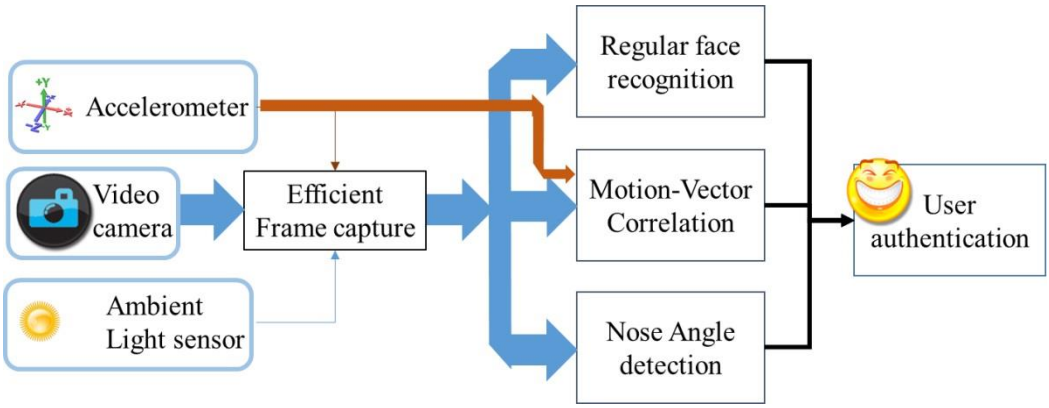


Figure 2. Block diagram of proposed approach. Existing face recognition schemes use 2D frames from smartphone camera to authenticate the user. Our approach builds on top of regular face recognition algorithms to counter 2D media attacks and virtual camera attacks. 2D media attacks are countered by Nose Angle Detection (NAD) algorithm while virtual camera attacks are countered by Motion-Vector Correlation (MVC). The ambient light sensor is used to improve the lightening conditions of face-capture from different angles. The accelerometer sensor is used to select inputs to NAD algorithm and as an input in MVC algorithm

Because these sensors are already built-in for almost all the smartphones, our approach is naturally suitable for smartphone unlock and app login without introducing any extra costs.

III.B Overview of Our Method

As mentioned in Section II, *liveness detection* and *texture analysis* have critical limitations, while *motion analysis* is not reliable under complex illumination. 3D facial authentication is robust to *2D media attacks*. However, they are much more time-consuming and difficult to use. The typical implementation requires that the motion of the user's head is synchronized with system instructions (e.g. the arrows shown on the screen in Toshiba Face Recognition Utility), which puts the burden on the user side. In the experiment (see Section VID), we find that a single trial in Toshiba Face Recognition Utility takes more than 20 seconds, and even for a genuine user, it often needs multiple trials to successfully log in. This fact significantly hurts the usability of 3D facial authentication, for it is more troublesome than simply using a password.

Our proposed approach is simple, intuitive and easy to use. It only requires the user to pick up the phone, move it horizontally for a short distance in front of the face, and it is done. In Section VI, we will show that the average time cost for authentication is less than two seconds, which is significantly faster than the existing 3D recognition methods and comparable to the credential-based method. One of the differences between our method and the existing 3D capturing is that for the former, the synchronization process mentioned above is not needed, which significantly eliminates users' burden. The basic intuition behind our approach is that by utilizing smartphones' motion sensors and object recognition techniques, we are able to infer the relative position between the camera and the user's face, so that the synchronization can be performed automatically and quickly.

Motion sensor readings are also used to compare the shake of the cellphone with the shake of the video been recorded during the authentication process. In this way, our method can also defend against *virtual camera attacks*, and more importantly, no extra operation is imposed on users. Since our method is of high security and very easy to use, it is a promising technique that can bring the smartphone users with more pleasant authentication experience.

Our work does not touch upon regular face recognition techniques that focus on recognition of facial features to identify and authenticate a person (from adversaries), which are orthogonal to our work (see 4th paragraph of Section VII). The main contribution of our system is nose angle detection algorithm to counter *2D media attacks* and motion-vector correlation algorithm to counter *virtual camera attack*s. Figure 2 gives a block diagram explaining the main working logic of our scheme which will be detailed in the following sections. Section IV discusses nose angle detection scheme, and Section V presents motion vector correlation algorithm, respectively.

III.C Attack Model for 2D Media Attack

In the context of facial authentication, *2D media attack* refers to the attacks that use planar photos or video clips containing user's face to cheat the authentication system, making it believe it is a real user's face. When using photos, such an attack is also referred to as *photo attack* or *print attack* in some literatures.

In our attack model, attackers can use either photographs or videos containing user's face to spoof the au-

thentication system, and we assume the quality of these photos or videos can be sufficiently high. In the authentication process, the users' faces or planar counterfeits are captured by the front-facing camera of smartphones, whose typical resolution is approximately 1 to 2 mega pixels. There is no specific requirement on the illumination conditions when the attack happens.

III.D Attack Model for Virtual Camera Attack

As mentioned in Section I, *virtual camera attack* refers to the attack that virtual camera software streams a clip of pre-recorded video containing a genuine user's face, circumventing the authentication system by making it believe the video is captured in real time.

In the attack model, attackers have the full knowledge of the authentication requirements (e.g. eye-blinks, face expressions, the movement of the head or the movement of the cellphone, etc.). In addition, the attacker may be able to obtain the video in which the genuine user is performing required actions (to log in). In the attack, the pre-recorded video is streamed via virtual camera software to cheat the authentication system. Here we assume the attack's target is websites or smartphone apps, whose authentication system typically locates at the server side.

We also assume the attacker uses the market-available virtual camera tools or lightly modified version of such tools. Their ability is to stream a recorded video, making the operating system believe it is a real-time video captured by a hardware camera in the smartphone. The attacker does not have the power to forge the motion sensor readings of the phone (see last paragraph of Section VII).

IV. NOSE ANGLE DETECTION

In this section, we explain how our proposed method defends against 2D media attacks without penalizing the authentication speed.

IV.A Differentiate 3D Face from 2D Counterfeits

Assume we have a sequence of images (or video frames) captured by a front-facing camera during the horizontal move in front of the face. In this subsection, we present how to differentiate real 3D faces from 2D counterfeits based on these images. Theoretically two images are good for the detection (one from the left side of the face and the other from right), but more images can improve the accuracy, as well as prevent the attacker from changing the photo in the middle (clever attackers may use different photos to perform the attack). We leave the timing of the image capturing to the next subsection.

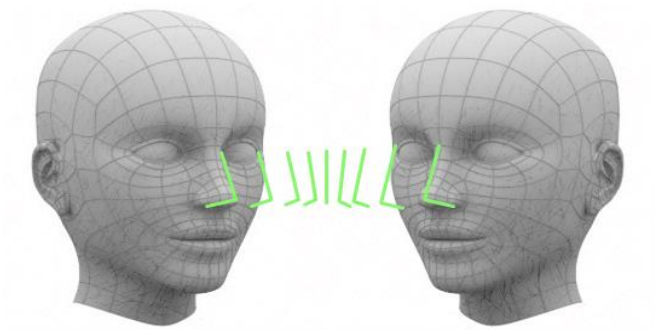


Figure 3. Change of the nose in a sequence of frames

Given an image or video frame P containing user's face, we first transform it to a grayscale image, noted as P′. This operation is trivial; we do it because some libraries only take grayscale inputs. Let the histogram of P′ be H′, we perform histogram equalization on H′ to enhance the contrast. Assume (x, y) is a pixel in P′ and its gray level is i, let:

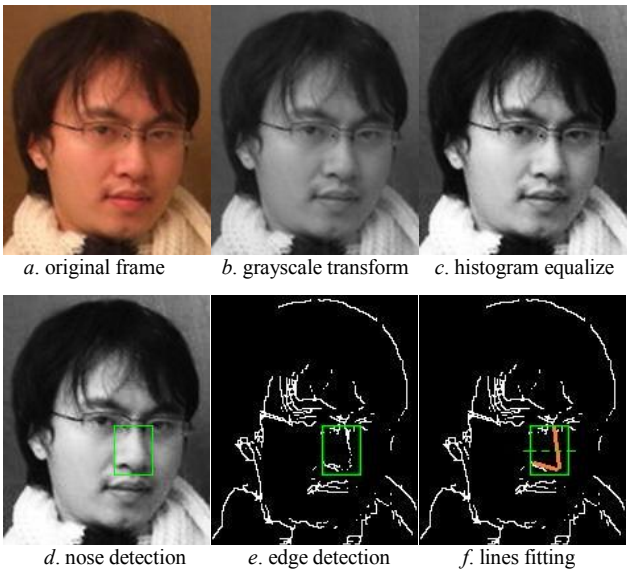
$$p(i) = \frac{n_i}{n} \ (1 \le i \le L)$$

where n is the number of pixels of the image, n_i is the number of occurrences of gray level I, and L is the total number of gray levels. Viewing p as a probability density function, the corresponding cumulative distribution function is:

$$cdf(i) = \sum_{j=0}^{i} p(j) \qquad (1)$$

The gray level of the pixel (x, y) in the new image (after histogram equalization) is:

$$cdf(i) \cdot (\max\{i\} - \min\{i\}) + \min\{i\} \qquad (2)$$



a. original frame *b*. grayscale transform *c*. histogram equalize

d. nose detection *e*. edge detection *f*. lines fitting

Figure 4. Nose border detection

Here i is the gray level of (x, y) in P′. $\max\{i\}$ ($\min\{i\}$) is the largest (smallest) gray level existed in P′. Note the new image we get as P″, and all of the following processing is performed on P″.

In the next step, we extract the nose region from P″. We put emphasis on the nose because it tends to show significant differences between a real face and a planar photo in our method. Given a series of images (captured by the front-facing camera) of a planar face photo, even if they are taken from different angles, the noses in these images are of the same shape and can match each other after necessary rotation, translation and scale. Instead, given a real 3D face, the noses in multiple images from different angles cannot simply match each other this way. The images taken from left and right will show different sides of the nose. Figure 3 illustrates the change of the nose (of a real 3D face) captured by a front-facing camera when the smartphone moves horizontally from left to right. The solid lines depict the outlines of the nose. We would like to emphasize that in Figure 3, the face on the left and the one on the right are not mirrors, but they present either side of the face respectively.

There are two approaches to detect the nose. The first is using face detection tools to figure out the face region, and then infer the approximate position of the nose based on biology knowledge. The second approach detects the nose directly by utilizing Haar-like features. Haar-like features are useful for real-time detection of face and facial regions such as eyes or nose. They are used in popular Viola-Jones face detection algorithm [20] and implemented in tools such as *Haar Cascades* in OpenCV. We use the second approach for its higher accuracy and faster implementation. Figure 4 illustrates each step starting from the original frame (Figure 4a, 4b, and 4c illustrate P, P′ and P″ respectively for a sample image). Figure 4d is the output of nose detection algorithm.

After obtaining the region of the nose in P″, we calculate the nose edges. We test various edge detection operators, such as *prewitt detector*, *marr detector*, *canny detector*, etc., and their performances are compared in Section VIB. The detected edges are presented in Figure 4e (in this example, *prewitt operator* is used), and here we only focus on the edges within the nose region which is given by the previous step.

Next, we use two straight lines to perform curve fitting on the nose edge. MMSE (minimum mean square error) is employed for the estimation. Assume a line is expressed as $ax + b$, we have:

$$MSE = \sum_i (ax_i + b - y_i)^2 \qquad (3)$$

where (x_i, y_i) are the points in the nose edge (i.e. the edges within the green rectangular in Figure 4e). For a single straight line, we can easily calculate the values of a and b that minimize *MSE*. However, for the fitting problem with two uncorrelated lines, the computational complexity becomes much higher. We use a heuristic to reduce this complexity. First, we mask the lower half of the nose area and fit the edge within the upper half with a single line (using Equation 3), noted as l_1. Second, we unmask the lower half, erase all the points that are close to l_1 (within a threshold) or its extension, and then fit the rest of the points by the other line (again, using Equation 3), noted as l_2. The result is illustrated in Figure 4f. Despite a heuristic approach, it shows good accuracy in experiments (see Section VIB) and largely reduces the computational overhead.

Apparently, l_1 and l_2 (or their extensions) will form an angle. If it is a real face, the orientation of this angle will reverse when the camera passes the midline of the face (as the green lines shown in Figure 3). However, if it is a planar photo, this orientation change does not happen. In this way, we are able to differentiate a real face from a 2D counterfeit. The detection accuracy of our method is presented in Section VI.

IV.B Cooperation of Motion Sensor and Camera

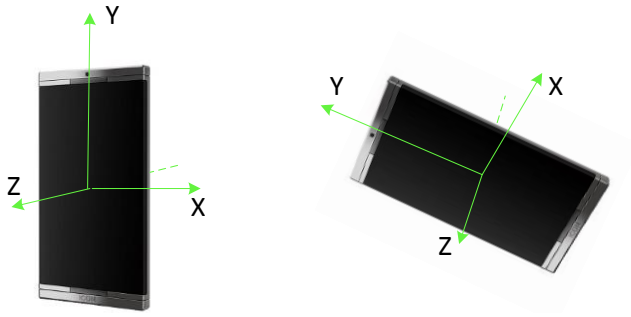
In the previous subsection, we talked about how to tell the difference between a genuine user's face and a 2D counterfeit (photos or videos). However, the advantage of our method not only lies in the ability of detecting such attacks, but also it is easy to use and much faster than the existing 3D face authentication methods. To achieve this goal, the key is to utilize smartphones' motion sensors and capture users' face image cleverly. The authentication process of our method is described as follows.

1) As soon as the authentication starts (time noted as T_0), the front-facing camera begins to capture video and perform face detection (detect the existence of human face, OpenCV *haarcascade* is used). Once the face area is greater than or equal to a threshold (in experiment, we set the default value to 40% area of the video frame), the front-facing camera starts recording video (time noted as T_s). The area of the face is approximated by the area of the face detection output (ususally a rectangular). At the same time (T_s), our system starts sampling the readings of accelerometers. If no face is detected, the authentication system will show a prompt and ask the user to move the phone closer to the face.

2) During the video recording, once the face area is smaller than a threshold (by default, 30% of the frame area) or the face can no longer be detected, the video recording stops. If no horizontal move is detected, a prompt will show on the screen asking user to move the phone horizontally in front of the face. If the horizontal move stopped for a time period longer than a threshold, the video recording also stops. The accelerometer sampling always terminates at the same time when video stops recording (time noted as T_e).

3) During the horizontal move, if the light sensor detects that the ambient illumination is very low, the system will change the majority of screen area to brighter colors and turn the screen brightness up, which helps increase the illumination, make the nose outline clearer, and thus improve the accuracy of edge detection.

4) Our system analyzes the accelerometer readings (from T_s to T_e), based on which it calculates the time when the phone is at its left most and right most position during the horizontal move, noted as t_l and t_r, respectively. Then, the video frames captured around time t_l and t_r are used for 2D media attack detection, in which the method introduced in Section IVA is applied. More details are presented as follows.



a b

Figure 5. Three axes of the accelerometer and orientation in smartphones

The accelerometer in smartphones typically has three axes, as shown in Figure 5 (z axis is orthogonal to the plane of the phone screen). In order to perform a successful facial authentication, the virtual line linking two eyes should be approximately parallel with the width dimension of the front facing camera. Therefore, the horizontal move required by our method is mostly related with the acceleration along the x axis. As we know, accelerometers measures the acceleration applied to the devices. An important fact is that when a cellphone is stationary, its accelerometer reads a magnitude of about 9.81 m/s^2, pointing to the direction of gravity; when the phone is in free fall, the accelerometer reads 0. If we use the smartphone when standing or sitting, it is most likely that the gravity only affects y axis (Figure 5a). However, if a user plays with his phone on bed or lying on sofa, gravity can influence x axis readings (Figure 5b). Therefore, we first need to filter out gravity to get the real acceleration of the phone. Let $\vec{G}_{t_i}$represent the estimate of gravity. Note the accelerometer readings as $\vec{R}$ (this vector has three components: x, y, and z), we calculate:

$$\vec{G}_{t_i} = \alpha \vec{R}_{t_i} + (1-\alpha)\vec{G}_{t_{i-1}} \quad (4)$$

and note the real acceleration as $\vec{A}$, we have:

$$\vec{A}_{t_i} = \vec{R}_{t_i} - \vec{G}_{t_i} \quad (5)$$

In Equation 4 and 5, t_i ($i \in Z^+$) are the time points that the sensor readings are collected. The basic idea is using exponentially-weighted moving average as a low-pass filter, which isolates the force of gravity. Then the remaining high frequency part is thereby the real acceleration of the device. α in Equation 4 is a smoothing factor, which defined as:

$$\alpha = \frac{\Delta t}{\mathrm{T} + \Delta t} \quad (6)$$

here Δt is the sensor sampling interval (i.e. $t_i - t_{i-1}$), and T is a time constant. By default, we let T = 5Δt.

Having obtained the real acceleration of the device, we can calculate its displacement during the horizontal move. It can be safely assumed that the velocity of the cellphone at T_s is 0 or close to 0, because the horizontal move has not started yet. So we have:

$$s(t) = \iint_{T_s}^{t} A^x(t)\, d^2t \quad (7)$$

where $A^x(t)$ is the x-component of $\vec{A}$, that is, the real acceleration of the phone along the x axis. $s(t)$ is the relative displacement from the original position (position at T_s). Due to the discrete nature of sensor readings, Equation 7 equals:

$$s(i) = (\Delta t)^2 \sum_i \sum_i A^x_{t_i} \quad (8)$$

The smaller Δt is, the more accurate $s(i)$ we get. By default, we use Δt = 0.01s. Then we calculate the min and max values of $s(i)$ based on Equation 8 such that $T_s \le t_i \le T_s$. They stand for the left-most and right-most position during the horizontal move respectively. Note the values of t_i as t_l and t_r where $s(i)$ reaches its minimum and maximum.

We examine the recorded video around time t_l, and pick up three frames at t_l – 0.1s, t_l, and t_l + 0.1s (if the video is 30 fps, they are approximately 3 frames away from each other). These three frames are processed using our method discussed in Section IVA, and the frame with minimum *MSE* is retained (refer to Equation 3). The same operation is performed on the frames around time t_r. Finally, two retained frames are compared to each other on the orientation of the angle between l_1 and l_2 (see Section IVA).

So far we have discussed how to quickly differentiate a real 3D face from a 2D photo (or video) with the help of motion sensors. Apart from the frame processing mentioned above, we perform anomaly detection on each frame between t_l and t_r. Face detection is applied on these frames to detect the absence of user's face or duplicated faces, in order to prevent the attacker from changing photos in the middle. Although this operation has to be performed on all the frames between t_l and t_r, the cost is much lower than the 2D counterfeits detection discussed above. Clever attackers will be further discussed in Section VII.

Besides, the frame around time (t_l + t_r) / 2 is used as

the input for routine face recognition (matching with a template whose identity is being claimed); but for this task, we can employ the existing methods and it is orthogonal to our work.

V. MOTION VECTOR CORRELATION

In Section IV, we present our method that defends against 2D media attacks. Now we move on to the detection of virtual camera attacks, which is another important threat to the facial authentication.

The attack model of the virtual camera attack is described in Section IIID. Since smartphones are hand-held devices, non-intentional shakes are inevitable for facial authentication videos captured by the front-facing camera. The basic idea of our method is to extract these shakes from the video, and compare with the shakes detected by the motion sensors. If it is a match, we can infer that the video is captured in real time; otherwise, it is very likely to be a pre-recorded video streamed via virtual camera software.

An important rule is that only small-scale random shakes (non-intentional hand shaking) are used for the matching, because large-scale motion trajectory can be easily emulated. Our virtual camera detection is performed by reusing the video and sensor readings recorded during T_s and T_e (see Section IVB); no extra operation is required from users.

Now we present our method for video motion extraction. Given two consecutive frames, P_i and P_{i-1}, we first reduce them in scale (resize to $m \times n$). Assuming the frame aspect ratio is 3:4, by default, we let $m = 24$ and $n = 32$. The resized frames are noted as $\overline{P}_i$ and $\overline{P}_{i-1}$ respectively. Then we apply panning (vertical and horizontal), rotation and zooming on $\overline{P}_i$, and the frame after processing is noted as $\widetilde{P}_i$. Our goal is to find the $\widetilde{P}_i$ that best matches $\overline{P}_{i-1}$. In our method, we set a range for the panning, rotation and zoom operations. For example, the panning should be no more than 5 pixels (in each direction) and rotation is less than 10 degrees. The limitation is introduced for two reasons. First, we are only interested in small-scale shakes. The time interval between two consecutive frames is only around 0.03 second. If the difference between them is large, it is very likely that what we detect is large-scale motion instead of a small shake. Second, since we need to try all the combinations of panning, rotation and zoom to find the best match, the search space is considerably large. Putting these physical restrictions on the search space accelerates our method without any negative implications on results.

The degree of match between $\widetilde{P}_i$ and $\overline{P}_{i-1}$ is measured by correlation coefficient. Compared with absolute value matching, it better tolerates the ambient light change and camera ISO / aperture auto adjustments. Correlation coefficient of $\widetilde{P}_i$ and $\overline{P}_{i-1}$ is defined as follows:

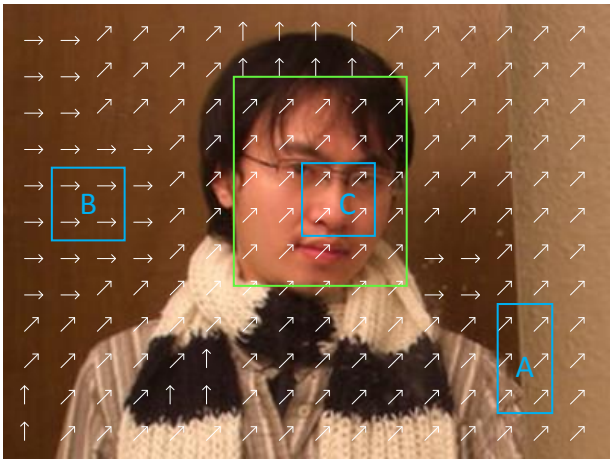


Figure 6. Small shake extraction from video

$$r = \frac{\sum_{0 \le x < m, 0 \le y < n} (p_{x,y} - \bar{p})(q_{x,y} - \bar{q})}{\sqrt{\sum_{0 \le x < m, 0 \le y < n} (p_{x,y} - \bar{p})^2} \sqrt{\sum_{0 \le x < m, 0 \le y < n} (q_{x,y} - \bar{q})^2}}$$

where $p_{x,y}$ is the value of pixel (x, y) in $\overline{P}_{i-1}$, and $q_{x,y}$ is the value of pixel (x, y) in $\widetilde{P}_i$. $\bar{p}$ is the mean value for all $p_{x,y}$ while $\bar{q}$ is the average of $q_{x,y}$. For each $\widetilde{P}_i$, we calculate r, and the one with highest r is memorized (noted as $\widehat{P}_i$). The operation performed between $\widetilde{P}_i$ and $\widehat{P}_i$ is noted as OP_i.

If the largest r is smaller than a threshold (0.7 in our settings), it means very likely the motion between P_i and P_{i-1} is so large that any minor shift will not let them match. In this case, the result is discarded and we move on to the next frame pair (P_{i+1} and P_i). Otherwise, the following steps are performed.

We divide P_i into small blocks. For each block, we calculate a shift to make it best match P_{i-1} (in their original resolutions) by using the same approach as for $\widehat{P}_i$. The only difference is that previously we apply panning, rotation and zooming, but now only vertical and horizontal panning is allowed. In other words, now OP_i is fine-tuned in the block level. The results of the best shift for each block is illustrated on a sample frame in Figure 6 (the arrows stand for the adjustment added to OP_i for each block; OP_i itself is not included). If the majority of these adjustments are the same (or very close), we use this value (or the average) as the shake of P_i, noted as $\vec{K}_i$. Otherwise, we assign a priority to each block. The blocks in the background but with details have the highest priority (such as region A in Figure 6), the blocks in the foreground rank the next (e.g. region C), and the background blocks with few details have the lowest priority (e.g. region B). We choose the average adjustment of the blocks with the highest priority as $\vec{K}_i$. Foreground and background are differentiated by the previous face detection output (blocks within the face region is considered as

foreground, otherwise background), and details are judged by the diagonal subband of the wavelet transform of P_i (diagonal subband values are close to zero if few details exist). If the panning component of OP_i is smaller than a threshold, we add this component to $\vec{K}_i$ and note the result as $\overrightarrow{\mathcal{K}_i}$. Otherwise, $\overrightarrow{\mathcal{K}_i}$ is the same as $\vec{K}_i$. Similarly, we calculate $\overrightarrow{\mathcal{K}}$ for each frame during T_s and T_e.

Compared with the standard motion vector extraction method, our method includes a number of modifications to improve the performance. Authentication videos have their unique characteristics. For example, the foreground of an authentication video is always a human or human's face; these videos usually do not contain fast moving objects, etc. By utilizing these characteristics, our method presented above can extract motions faster than the standard method without negative implications on results.

In parallel to the motion extraction from video frames, we extract shakes from accelerometer readings, where the method is similar as in Section IVB. The shake of the phone ($\vec{A}$) is calculated by Equation 5 (see Section IVB). In order to remove large-scale motions, we redefine α by letting T = $2\Delta t$ in Equation 6. As T decreases, α increases, and the cutoff frequency of the filter becomes higher. The value of T is determined by the experiments (please refer to Section VIC). Since $\overrightarrow{\mathcal{K}}$ is a two dimensional vector that only includes the shifts parallel to the camera sensor plane, we remove the z-component of $\vec{A}$, making it consist of merely the comparable information as in $\overrightarrow{\mathcal{K}}$. The trimmed $\vec{A}$ is noted as $\vec{\mathcal{A}}$.

$\overrightarrow{\mathcal{K}}$ and $\vec{\mathcal{A}}$ are then aligned according to the time scale. As mentioned previously, some elements of $\overrightarrow{\mathcal{K}}$ could be dropped in our method, so we also remove the corresponding elements in $\vec{\mathcal{A}}$ if necessary. Correlation coefficient is employed to measure the similarity between $\overrightarrow{\mathcal{K}}$ and $\vec{\mathcal{A}}$:

$$\rho = \frac{E[(\vec{\mathcal{A}} - E(\vec{\mathcal{A}}))(\overrightarrow{\mathcal{K}} - E(\overrightarrow{\mathcal{K}}))]}{\delta_{\vec{\mathcal{A}}}\delta_{\overrightarrow{\mathcal{K}}}} \quad (9)$$

where E is expectation and δ stands for standard deviation. The closer $|\rho|$ is to one, the better $\overrightarrow{\mathcal{K}}$ and $\vec{\mathcal{A}}$ matches. Otherwise, if $|\rho|$ is close to zero, a virtual camera attack is assumed. The threshold is determined by experiments (please refer to Section VIC).

VI. EVALUATIONS

We implement our method and conduct real-world experiments to evaluate it. We first talk about the implementation details and experiment settings. Then, we evaluate the accuracy of our 2D media attack detection scheme and virtual camera attack detection approach respectively. After that, the authentication speed of our method, exiting 3D facial authentication and credential-based authentication methods are compared.

VI.A System Implementation and Experiment Preparation

Our system is implemented in a Samsung Galaxy Nexus smartphone. The operating system is Android 4.2.2 and the front-facing camera has 1.3 mega pixels. The video captured in facial authentication is 480×720@24fps, and in the succeeding processing we chop them to 480×640.

We use *Haar Cascades* in OpenCV for face and facial region detection. haarcascade_frontalface_alt2.xml and haarcascade_mcs_nose.xml are employed to detect face and nose region respectively. JavaCV is used as a wrapper to call the native functions.

The main objective of our system is defending against 2D media attacks and virtual camera attacks without introducing high costs. Face identification techniques are orthogonal to our work; any authentication system using 2D face recognition can benefit from our method. But for the completeness, we do include a PCA (principal component analysis) based facial identification module, which is also implemented using OpenCV. The video frames around time $(t_l + t_r) / 2$ are used (see Section IVB) as the input for template matching.

We now compare the face detection accuracy (using the module mentioned above) when the normal operation, hand-picked *best* frame, and automatically extracted frame (by our method) are used, respectively. The purpose of this experiment is to check whether our method is able to pick the appropriate frame for template matching, since face recognition algorithms are usually sensitive to the orientation between the camera and the user.

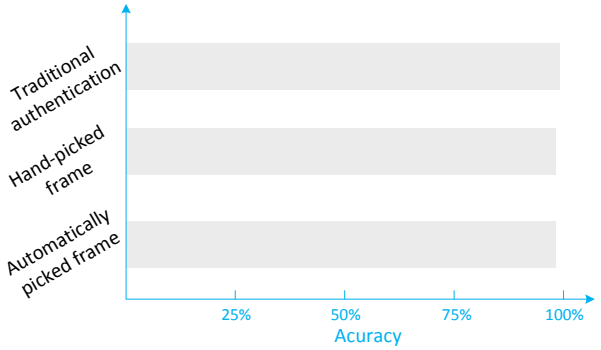


Figure 7. Performance of automatic frame selection

In Figure 7, traditional authentication is the case where a user holds the smartphone, keeps it still, and the smartphone takes a picture of the user's face and performs the recognition. Other two cases use the video collected by our method (horizontally move the phone in front of the user's face); the difference lies in that, "hand-picked frame" refers to the approach in which we manually pick the *best* frame (where the face squarely opposite to the camera) from the video, while "automatically picked

frame" fully utilizes our method and the frame is picked by calculating $(t_l + t_r) / 2$. For each case, we plot the accuracy of 180 trials, where data are from 9 volunteers (volunteers will be detailed in Section VIB). The results show that our method is good enough in automatic frame selection (for succeeding template matching).

VI.B Accuracy of 2D Media Attack Detection

In this subsection, we evaluate the 2D media attack detection accuracy of our system.

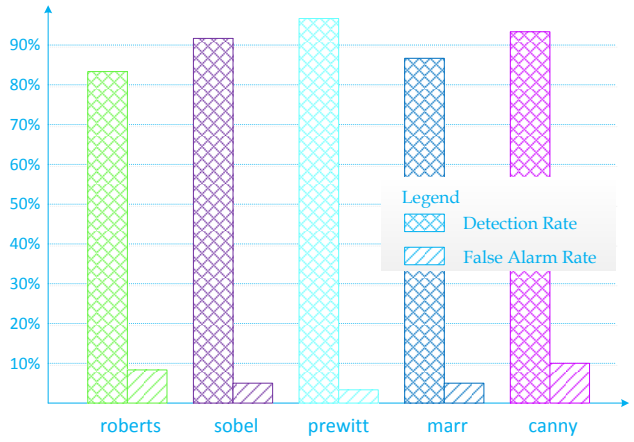


Figure 8. 2D media attack detection accuracy

We have 9 volunteers, each of them performing 20 trials of facial authentication using our system. Besides, we take two pictures and one clip of video for each volunteer, all of which are of high quality and with the head centered. Volunteers include 8 males and 1 female. 6 of them are in their twenties while the other 3 are in their thirties. They are all friends of authors but are not inside or related to our research lab. They participated in experiments voluntarily without compensation and were not aware of the purpose of experiments. For the facial authentication trial using our system, each volunteer was asked to hold the Galaxy Nexus in front of his / her face, click a "start" button on the screen, move the phone horizontally for a short distance, and then click the "complete" button. Besides two buttons mentioned above, the screen shows the real time scene captured by the front facing camera as well as a large green square. We asked the volunteers to locate their face approximately inside this square when moving the phone.

During the attack, pictures are printed on white paper in high quality (in case of the video attack, videos are played in another device), facing the smartphone that is used for authentication. Distance is carefully adjusted, making the size of the face on the planar media the same as the real one. We give each photo (video) ten trials. Therefore, in total we have 180 genuine user trials and 270 attacks (180 photo attacks and 90 video attacks). Half of them are performed indoor and the other half outdoor. For each test, the smartphone is moved horizontally for a short distance in front of the face (real faces or 2D counterfeits). The experiment result is shown in Figure 8.

Here detection rate refers to the number of detected attacks divided by the total number of attacks, while false alarms stand for genuine users' trials which are reported as attacks by our system. Therefore, a good detection system is expected to have a high detection rate and a low false alarm rate. In Figure 8, *x* axis exhibits 5 different edge detectors (for nose border extraction, refer to Section IVA). For each detector, we try different thresholds and the best one is used in the final settings. Figure 8 shows that *prewitt* and *canny* detector have the best detection rate (97% and 93%), while *prewitt* and *marr* have the best false alarm rate (3% and 5%). Combining the results, we choose *prewitt* detector for our method. Hereinafter, we use this detector in the rest of experiments unless otherwise specified.

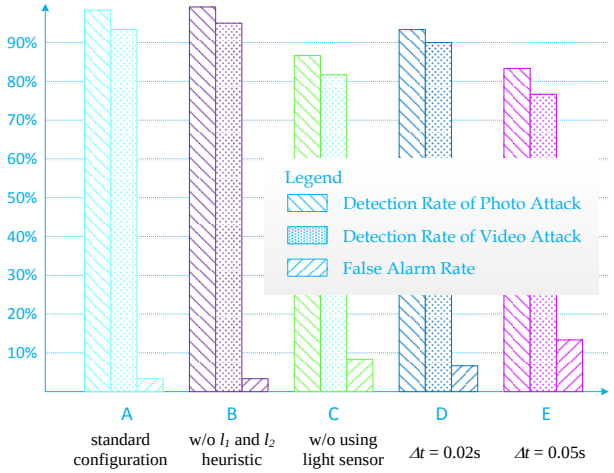


Figure 9. Detection accuracy under different configurations

In Figure 9, we show the detection accuracy of the photo attack and video attack separately. We also twist some configurations of our method to see how they affect the performance. In *x* axis, A is the standard configuration of our method. B uses brutal force search instead of the heuristic (see Section IVA) to find l_1 and l_2 (two lines used to fit nose edges). The brute force method has negligible performance gain as compared to the increased computational cost (15 times slower). In C, we disable the brightness auto adjustment described in *3)* of Section IVB, and a performance drop is observed. Because only half of our tests are performed indoor, the actual influence might be even stronger than shown in Figure 9. For D and E, we change the motion sensor sampling rate. It can be seen that increased sampling time leads to lower performance. For all the tests, an observation is that video attacks are slightly harder to detect than photo attacks.

Next, we compare the performance of our method to the 3D authentication method as well as the Android face unlock (in version 4.2.2) under 2D media attacks. For the 3D method, we use Toshiba Face Recognition Utility. The results are shown in Figure 10. The result shows that the detection rate of the 3D method is only marginally better than ours for both photo attack and video attack. However, our scheme has a lower false alarm rate than 3D authentication method (up to 10% difference). More important, the detection time of the 3D method is significantly longer than ours, which will be shown later.

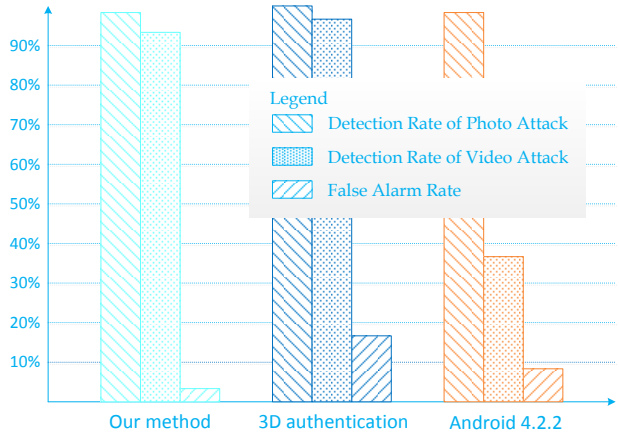


Figure 10. Comparing the detection accuracy with state-of-the-art appraoches

As to the face unlock alternative embedded in the Android system, since it includes an "eye blink" detection mechanism starting from version 4.1, it does a decent job in photo attacks, but it cannot effectively detect video attacks, since many videos naturally contains spontaneous eye blinks. In sum, our method can achieve a detection rate of 97% in 2D media attack detection, and its false alarm rate is as low as 3%. The speed of our method will be evaluated and compared in Section VID.

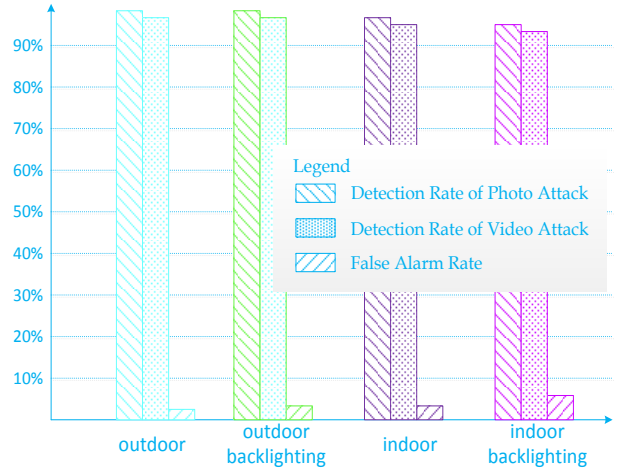


Figure 11. Detection accuracy under different illuminance

We now compare the accuracy of our 2D media attack detection algorithm under different illumination conditions. The indoor environment has illuminance of 10 - 50 lux, while the outdoor tests are performed under illuminance that varies from 50 to 200 lux. The averaged results for this set of experiments are shown in Figure 11. We can see from the results that the performance of our method is not greatly affected by the variation of illumination conditions (in contrast, as mentioned in Section II, the performance of motion analysis methods largely depends on the illumination). We also find that if we disable the light sensor and our automatic-screen-brightness-increase (Section IV-B) mechanism, in the indoor backlighting case, the false alarm rate will increase approximately 10 percent.

VI.C Accuracy of Virtual Camera Attack Detection

Now we evaluate our system's detection accuracy of virtual camera attacks. In our implementation, 2D media attacks can be detected either in the smartphone or at the server side (in order to apply to device unlock and website / app login respectively). However, the virtual camera attack detection can only be performed at the server side. A laptop (Toshiba Z930 with Intel i7-3667U CPU) is acting as the server, and the authentication video and sensor readings are streamed to it from the smartphone in real time.

We implement our system this way for two reasons. First, as mentioned in Section I, device unlock lies in the system level and has more control of hardware, so it is less likely to suffer from virtual camera attacks. On the other hand, for website / app login, which are susceptible to this attack, the authentication is always performed at the server side. Second, the computation overhead of virtual camera attack detection is relatively high. Implementation in the server can achieve better performance.

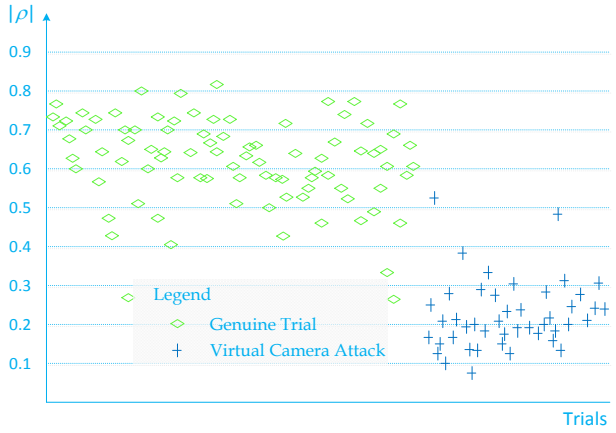


Figure 12. Threshold between attacks and genuine trials

During an attack, a pre-recorded video (with a genuine user's face in it) is streamed to the server instead of the live video captured by the front-facing camera. Meanwhile, the same horizontal move of the smartphone is also performed. The dataset we use is the same as the previous experiments. We have 180 genuine users' trials and 90 pre-recorded videos. They are divided equally into two sets, referred to as D1 and D2 respectively (each contains 90 genuine trials and 45 attacks).

First we use dataset D1 to determine the threshold for $|\rho|$ (see Section V; if $|\rho|$ is smaller than the threshold, an attack is assumed). $|\rho|$ of 90 genuine trials and 45 attacks in D1 are calculated using Equation 9 (T is fixed at $3\Delta t$) and presented in Figure 12. From the result, we can see that the genuine trials and the attacks are separated fairly clearly. $|\rho|$ of most attacks are smaller than 0.4 and genuine trails are the opposite. Based on this observation, we set the threshold of $|\rho|$ at 0.4, which is used as the default value of our method thereinafter.

Next, we vary the value of T (see Sections IVB and V) and see how it affects the detection accuracy. As mentioned before, in our method, large-scale motions are filtered out from both sensor readings and video motion vectors, but the amount that is removed from two sides (sensor readings and video motion vectors) should be approximately the same. Intuitively speaking, adjusting the value of T is the operation that twists the sensitivity of the sensor filter and makes two amounts to be removed approximately the same. The test is performed on dataset D2. The detection rate (solid line) and false alarm rate (dashed line) are plotted in Figure 13.

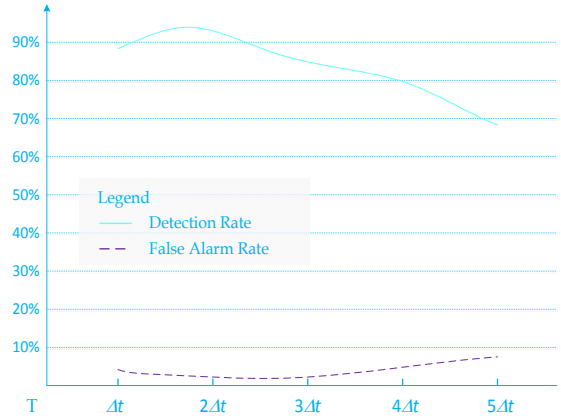


Figure 13. Detection accuracy of virtual camera attacks

The experiment results show that when T is around $2\Delta t$, the best performance can be achieved. So we use T = $2\Delta t$ as the default value in our method for virtual camera attack detection. From the experiment, we can see that our method is able to achieve a detection rate of 95% while the false alarm rate is as low as 2% in virtual camera attack detection.

VI.D Authentication Speed and Usability

In this subsection, we compare the authentication time of our method, 3D facial authentications, and credential based authentications.

In addition to the 180 trials in our system mentioned above, our volunteers perform 90 trials on Toshiba Face Recognition Utility (to log into a Toshiba laptop) using 3D facial authentication, as well as 90 trials to login a website by entering the username and password. For all three schemes, time is counted until the authentication result is sent back. To reduce the influence of network latency, we use local servers in the LAN. The time costs of three schemes are averaged and plotted in Figure 14 (gray bars). We can see that speed-wise, our system is comparable to the credential-based system, and 9 times faster than the 3D facial authentication method.

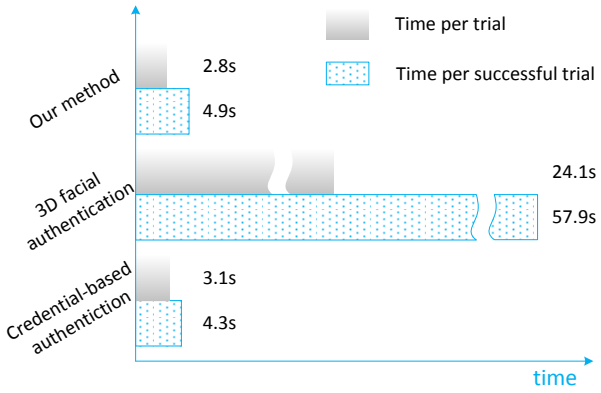


Figure 14. Authentication time comparison

All the trials used for calculation in the last experiment are in fact successful trials. In some cases, a trial can fail even for a genuine user. For example, the user may fail to synchronize the head movement with the instructions given by the 3D scheme; user's face may be too far away from the front camera; the required horizontal move is not detected by our method; or the user enters the wrong password by mistake, etc. In the following test, we include all the trials and calculate the time cost per a successful trial. The result is plotted as dotted bars in Figure 14. Our method is still very close to the credential-based scheme in speed, and its advantage over the 3D facial authentication becomes even larger (13 times faster), which exhibits good usability of our method.

VII. DISCUSSION

We claim that our method can detect 2D media attacks, including photo attacks and video attacks. In Android 4.2, users are required to blink their eyes during the face unlock, which can be easily spoofed by a video containing eye blinks. Our method is safe for such videos. However, clever attackers may think about using more sophisticated

videos. Since our system requires a short move of the smartphone from left to right in front of the face, how about using a video in which the genuine user slowly turns his / her head from right to left?

This method sounds good but in fact it can hardly compromise our system. The reason is as follows. First, the feasibility of the video attack against the Android face unlock is based on the fact that videos containing eye blinks are relatively easy to obtain (e.g. through social networks). Blinking is a spontaneous or voluntary action of human eyes. A video with a clear user's face in it has high probability to contain eye blinks as well. On the contrary, finding a video in which the user turns head slowly is much more difficult.

Second, as mentioned above, our method also collects sensor readings during the horizontal move of the smartphone. Keeping the phone still and only relying on the head turn in the video can be easily detected by our system. The attacker must move the phone horizontally and synchronize the smartphone movement with the head turn in the video (making it look like the phone is moving, but the user is not turning head), which is very difficult.

In this work, we have shown a prototype implementation of our system on the top of *haarcascade* face recognition utility available in openCV. On one hand, the recognition accuracy of this default face recognition scheme can limit the overall accuracy of our system. Iris recognition or other advanced facial feature recognition schemes can be used instead of *haarcascade* to achieve higher detection rate and lower false alarm rate for our authentication system. On the other hand, different recognition schemes mentioned above are orthogonal to our research. The method discussed in this paper can still be used to counter against 2D media attacks and virtual camera attacks when the face recognition schemes other than *haarcascade* are used.

As mentioned in Section IIID, the proposed method is based on the assumption that motion sensor readings from the smartphone that performs facial authentication are not compromised. If an attacker is able to forge the sensor readings and manipulate the video (i.e., use a pre-recorded video as if it is captured by the physical front-facing camera of the smartphone) at the same time, our system can be compromised. In this paper, we focus on defending against the virtual camera attacks which take advantage of virtual camera software or lightly modified versions of such software, instead of an omnipotent Trojan which can compromise the OS, manipulate the video and forge the sensor readings simultaneously. The former is the tools that are available on market, while the latter does not exist for now. Of course, an attacker may think of using virtual camera software as well as a Trojan that can modify the sensor readings at the same time. However, it requires that the virtual camera and Trojan cooperate together and synchronize perfectly, which is difficult and significantly raises the bar for breaking into the authentication system.

VIII. CONCLUSION

In this paper, we proposed a novel facial authentication method for smartphones, which can defend against 2D media attacks and virtual camera attacks without penalizing the authentication speed. Motion sensors inside smartphones are employed to achieve this purpose.

We conduct extensive experiments to evaluate our method. The result shows that the attack detection rate of our method is very high, while its speed is comparable to credential-based methods, and over ten times faster than the existing 3D facial authentications. Our work overcomes the most important limitations of current facial authentication techniques, making it more practical and useful for smartphones. It has the potential to change the current state of smartphone authentications.

IX. REFERENCES

[1] R. Jenkins, A. Burton, *"100% accuracy in automatic face recognition,"* Science, 319 (5862): 435, January 2008.

[2] A.F. Abate, M. Nappi, D. Riccio, G. Sabatino, *"2D and 3D face recognition: a survey,"* Pattern Recognition Letters, 28(14): 1885-1906, 2007.

[3] G. Pan, L. Sun, Z. Wu, and S. Lao, *"Eyeblink-based anti-spoofing in face recognition from a generic webcamera,"* IEEE 11th International Conference on Computer Vision, 2007.

[4] www.androidauthority.com/android-jelly-bean-face-unlock-blink-hacking -105556/

[5] www.youtube.com/watch?v=UKIZBbvloO8

[6] C. Queirolo, L. Silva, O. Bellon, and M. Pamplona, "*3D face recognition using simulated annealing and the surface interpenetration measure,"* IEEE Trans. on Pattern Analysis and Machine Intelligence. 32(2): 206-219, 2010.

[7] V. Blanz and T. Vetter, *"Face recognition based on fitting a 3D morphable model,"* IEEE Trans. on Pattern Analysis and Machine Intelligence. 25(9): 1063-1074, 2003.

[8] http://www.manycam.com/

[9] G. Pan, L. Sun, Z. Wu, and Y. Wang, *"Monocular camera-based face liveness detection by combining eye-blink and scene context,"* Journal of Telecommunication Systems, 2009.

[10] J. Li, Y. Wang, T. Tan, and A.K. Jain, *"Live face detection based on the analysis of fourier spectra,"* Biometric Technology for Human Identification, 2004.

[11] J. Bai, T. Ng, X. Gao, and Y. Shi, "*Is physics-based liveness detection truly possible with a single image*?" IEEE International Symposium on Circuits and Systems. 2010.

[12] X. Tan, Y. Li, J. Liu, and L. Jiang, *"Face liveness detection from a single image with sparse low rank bilinear discriminative model,"* Computer Vision ECCV, vol. 6316, pp. 504-517, 2010.

[13] W. Bao, H. Li, N. Li, and W. Jiang, *"A liveness detection method for face recognition based on optical field*," IEEE International Conference on Image Analysis and Signal Processing, 2009.

[14] Y. Wang, J. Liu, and X. Tang, *"Robust 3D face recognition by local shape difference boosting,"* IEEE Trans. on Pattern Analysis and Machine Intelligence, 32(10): 1858-1870, 2010.

[15] B. Amberg, R. Knothe, and T. Vetter, *"Expression invariant 3D face recognition with a morphable model,"* 8th IEEE International Conference on Automatic Face and Gesture Recognition, 2008.

[16] http://uniqul.com/news/17-launch

[17] http://www.guthspot.se/video/deshaker.htm

[18] F. Vella, A. Castorina, M. Mancuso, and G. Messina. *"Digital image stabilization by adaptive block motion vectors filtering"*. IEEE Transactions on Consumer Electronics, 48(3): 796-801, 2002.

[19] Y. Xu, and S. Qin, *"A new approach to video stabilization with iterative smoothing,"* IEEE International Conference on Signal Processing, 2010.

[20] P. Viola and M. Jones, *"Robust real-time face detection,"* International Journal of Computer Vision, (57)2: 137-154, 2004.

[21] G. Sandbach, S. Zaferiou, M. Pantic, L. Yin, "*Static and dynamic 3D facial expression recognition: A comprehensive survey."* Image and Vision Computing (2012).

[22] S. Shirazi, A. P. Moghadam, H. Ketabdar, and A. Schmidt. *"Assessing the vulnerability of magnetic gestural authentication to video-based shoulder surfing attacks."* In Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems, pp. 2045-2048. ACM, 2012.

[23] B. Battista, Z. Akthar, G. Fumera, G. L. Marcialis, and F Roli. *"Security evaluation of biometric authentication systems under realistic spoofing attacks."* IET biometrics 1, no. 1 (2012): 11-24.

[24] K. H. Kamer, A. Yüksel, A. Jahnbekam, M. Roshandel, and D. Skirpo. *"Magisign: User identification/authentication based on 3d around device magnetic signatures."* In Proceedings of fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (Ubicomm), pp. 31-34. 2010.