Stealing Passwords by Observing Hands Movement

Diksha Shukla, *Student Member, IEEE*, and Vir V. Phoha, *Senior Member, IEEE*

***Abstract*— The use of mobile phones in public places opens up the possibilities of remote side channel attacks on these devices. We present a video-based side channel attack to decipher passwords on mobile devices. Our method uses short video clips ranging from 5 to 10 s each, which can be taken unobtrusively from a distance and do not require the keyboard or the screen of the phone to be visible. By relating the spatiotemporal movements of the user's hand during typing and an anchor point on any visible part of the phone, we predict the typed password with high accuracy. The results on a dataset of 375 short videos of password entry process on a Samsung Galaxy S4 phone show an exponential reduction in the search space compared to a random guess. For each key-press corresponding to a character in the passwords, our method was able to reduce the search space to an average of 2–3 keys compared to ∼30 keys if one has to guess the key randomly. Thus, this paper reaffirms threats to smartphone users' conventional login in public places and highlights the threats in scenarios such as hiding the screen that otherwise gives the impression of being safe to the users.**

***Index Terms*— Biometrics, authentication, side channel attack, password, smartphone security, hand gestures.**

I. Introduction

WE PROPOSE algorithms and methods to predict passwords by analyzing the underlying dynamics of hand movements when typing on a mobile phone. The components of our system include a video clip that provides anchor points on any visible part of the phone and any visible part of the typing hand. No part of the screen of the phone needs to be visible. We achieve accuracy as high as 94% in predicting the cluster of keys where each cluster represents 5-6 spatially close keys on the keyboard. Using keyboard state transition probability and flight time between keys pressed obtained through analysis of the video frames, we correctly recognized an average of 70% of the characters in a password.

The process works as follows. Since the geometry of the keyboard on brand name phones such as iPhone, Samsung Galaxy S4, and HTC One uses a fixed design, we can locate the keyboard in the video by estimating the location, length, and width of the keyboard and by relating these dimensions in the video with the physical (known) dimensions of the keypad by simple geometric transforms between the two. The cues to deciphering the characters typed include the observations that the hand moves closer to the keyboard when pressing a key and moves away just after that (see Section III-E for finer nuances of this observation).

Fig. 1. An adversary recording the visible hand movements to infer the password being typed on the mobile phone screen. A user is typing his password on the mobile phone screen and the adversary is recording the video using a video recorder.

The prediction of the characters pressed poses specific challenges because any character can represent a sequence of key presses such as shift key or number key to transforming from alphabets to numbers and numbers to special characters. For example– to type the password *Sec@16*, a user will press the sequence of keys as $Shift \rightarrow S \rightarrow e \rightarrow c \rightarrow Number$ key $\rightarrow$ @ $\rightarrow$ 1 $\rightarrow$ 6. $Shift$ key changes the keyboard state from lowercase alphabets to uppercase alphabets and $Number$ key transforms the state from alphabets to numbers.

To the best of our knowledge, ours is the first work that predicts passwords solely on the basis of observed hand movements. Figure 1 shows the scenario, we analyzed in this paper. It shows, a user is entering the password on his mobile phone, and an adversary with a camera is capturing the video from a distance such that mobile phone screen is not visible and hence gives an illusion to the user of being safe. In the same fashion, an adversary can capture the video from a more sophisticated camera with optical zoom facility which enables to capture video from a remote distance. We performed our analysis on the password entry videos on Samsung Galaxy S4 phone and video captured by a range of video recorders starting from HTC One mobile phone camera with no optical zoom to Sony camcorder with high optical zoom capability.

Manuscript received August 13, 2018; revised December 22, 2018 and March 4, 2019; accepted March 27, 2019. Date of publication April 15, 2019; date of current version July 31, 2019. This work was supported in part by the National Science Foundation (NSF) under Grant SaTC-1527795. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Julian Fierrez. *(Corresponding author: Diksha Shukla.)*

The authors are with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244 USA (e-mail: dshukla@syr.edu; vvphoha@syr.edu).

Digital Object Identifier 10.1109/TIFS.2019.2911171

The following are the contributions of the paper:

1) We design an attack that decodes a user's password based on the hand movements recorded in a video clip. The attack can be executed in such a way as to not raise any suspicions to the user, so it poses a serious threat with the new range of sophisticated cameras now introduced in the market (such as smart glasses and smart watches).
2) Using a large dataset of the video recording of user's hand dynamics while they type their password on the mobile phone, we show our attack was able to break an average of over 70% of characters in a password.
3) The users have a strong conviction that covering the screen while entering a password at a public place adequately secures the password against eavesdropping attacks. Our findings provide an evidence against the notion because all the videos used in the study have no part of the screen visible.

Discussion – Follow-Up Potential Adversarial Scenarios

A 70%-character guess of user's passwords provides significant leverage to an attack because it opens several follow up adversarial possibilities. Below we list some of these possibilities.

1) *Reused Credential Attack* - Although research in the past has shown that reusing credentials across applications and websites puts a user at security risk [1], [2], several users still reuse their passwords across applications. An attacker could easily launch an attack on other linked sites for such a user. Research also suggests that a single good credential could lead to a bigger organizational level attack [3], [4] and hence our work exposes the possibility of such an attack by using video based channels to obtain credentials of potential users.
2) *Targeted User Attack* - Since the users type their passwords several times in a day, an adversary could obtain multiple videos of a target user over time and obtain a better estimate over time. Also, for a targeted user, an adversary could combine other personal information about the user to get a better estimate of the password in fewer guesses.
3) *Credential Trading* - Credential trading is another motivation behind the credential thefts. The attacks such as ours can be utilized by adversaries for trading the users passwords. These traded partial (or complete) passwords might lead to a bigger attack [5].

The rest of the paper is organized as follows. We discuss related work in Section II. Section III discusses the data sets and the detailed attack process. We present the attack results in Section IV and broader implications and our conclusions in Section V.

II. Related Work

Recent research show possibility of exploiting video based side channels to steal the smart device user's sensitive information [6]–[15].

A very closely related work to this paper is our previous work on user's Pin entry process on the mobile phones [6]. The attack was shown to work with the video recording of a user's hand movement while not compromising any information from the mobile phone screen display. The attack design in [6] is suitable for numeric Pins, but given the complexity of keyboard for password or text input, the attack would fail due to the close proximity of key locations and change of keypad state from characters to numbers and to special characters. The attack presented in this paper differs in several ways; (1) we consider the keypad state transitions in the attack, (2) we estimate the fingertip movement by using the observed typing hand point location in each frame. This gives us better estimate of the key touched considering the spatially close keys in an alphanumeric keypad, (3) the attack presented in this paper does not rely solely on one point tracked on the typing hand but rather takes multiple points to track and fuses the tracked results to get a better estimate of hand movement, and (4) we build a password language model to get a better prediction in case of lexical patterns in the user's passwords.

Another closely related work to this paper is by Xu *et al.* [7]. Xu *et al.* show the text reconstruction using a low-resolution video recording of the fingertip and the screen on which typing is being done. They show the video recording captured from a long distance such that it was impossible for the user to see the attacker. They show the attack to work even with the recording of the reflection of the typing finger and phone screen.

The attack by Ye *et al.* [8] utilizes video recordings of the users' mobile phone screen to build an attack on the smartphone pattern lock system. Ye et al. show that their attack could reconstruct over 95% of the graphical patterns in the first five attempts. The work by Chen [9] show an automated attack for fast inference of number inputs on the mobile phone screen using video recordings of the users' screen while they type. Another recent attack presented by Balagani *et al.* [10] on ATM pins uses the video recording of screen or the projector. The attack by Balagani et al. extracts timing information from consecutive key presses from the recorded video to infer the key sequence being entered on the keypad.

The attacks presented in [7]–[10] were based on the appearance of the screen while a particular key is being typed. Their attack model uses the information from the mobile phone screen display which makes our work different to theirs. For users who will be able to hide their screen while they type, these attacks will fail. Our attack model does not require any information from screen display while typing is being done. We only need a video recording of hand movement and a part of the back of the mobile phone while typing is being done.

The work by Raguram *et al.* [11] and Backes *et al.* [12] also shows the text reconstruction using reflection. In their work, they use state of the art image processing techniques to reconstruct the text. The fact that they use the information of display of the text in the reflection makes their work clearly very different from ours.

The attack design by Balzarotti *et al.* [13] uses a video recording of user's typing on the *desktop* keyboard. The video recording was done in such way that camera directly points to

the keyboard. Balzarotti *et al.* used a series of computer vision analysis followed by language modeling techniques to infer the text typed on the desktop keyboard. They use the information about which keys are not visible in the video while a particular key is being pressed and puts them as candidate keys to be pressed. Our work is clearly very different to the work by Balzarotti *et al.* Our attack is conducted in such way that it does not need the keyboard of the mobile phone to be visible.

The attack by Maggi *et al.* [14] uses the video recording while the user is typing on the smartphone screen. Their attack takes advantage of display feedback mechanism (the enlarged key display while it is being typed). In their attack set up, the recording was done while the camera was directly pointing to the smartphone screen. They use a classifier trained on the appearance of the enlarged display of characters to determine the characters typed in a stream of video recording. A similar kind of attack has recently been proposed by Yue *et al.* [15]–[17] on password and text entry process. In their work, they rely on the video recording of the smartphone screen and fingertip from a distance using advanced camera-enabled devices such as Google glass. They employ advanced image processing techniques to estimate the touched locations on the screen which in turn maps to the actual key presses.

In all three works by Balzarotti *et al.* and Maggi *et al.* and that of Yue *et al.*, the *cameras directly pointed at the keyboard* to capture the video of user's typing, i.e. direct observation of the appearance of the keyboard or the text typed. In our attack model, we do not use such a fine-grained information from the screen display. It needs only part of the hand and an anchor point on the mobile phone to be visible to launch our attack.

We also discuss in brief two side-channel attacks that despite having been evaluated in a desktop environment [18], [19] help put into context two interesting attributes of our attack. Using a neural network trained with samples collected from the intended victim, Asonov and Agrawal [18] showed that acoustic keyboard emanations could be used to retrieve typed text with close to 80% accuracy. This attack was then later refined by Zhuang *et al.* [19] who recovered up to 96% of typed *English* characters and 90% of random 5-character passwords (using only letters) in just a few guesses. A notable aspect of the attack refinement by Zhuang *et al.* was the elimination of the training process, making the attack applicable to victims for whom no previous recordings of keyboard emanations are available. In an attack against the SSH login mechanism, Song *et al.* [20] showed that the time intervals between SSH packets provide a significant amount of information on what users type during an SSH session. Using an attack tool, build based on Hidden Markov Models, Song *et al.* [20] showed their attack to reduce the SSH password search space by up to 50 times on average compared to exhaustive search.

Like the attack by Zhuang *et al.* [19], our attack does not require any user specific training phase, making it applicable to any video on the fly. Meanwhile, compared to the 50× reduction in the search space seen with the SSH attacks, our attack reduces the password search space by over 99% confirming its lethality relative to some of the known attacks on text input (see Section IV).

There is a series of work which explored the vulnerabilities posed by sensor data on the mobile phone. Cai and Chen [21] and Owusu *et al.* [22] used the accelerometer sensor data to infer the password typed on the smartphone screen. Miluzzo *et al.* [23] and Xu *et al.* [24] fused accelerometer readings with those of gyroscope readings while typing was being done. They also implemented an attack to infer password using the orientation and movement information extracted from mentioned smartphone sensors to determine the keys being pressed. A recent work by Simon and Anderson [25] showed an attack to infer the Pin entered on the smartphone screen. Attack by Simon and Anderson uses the output from front camera of the user's mobile phone and also output from the microphone. Wang *et al.* [26] presented an attack to infer the hand movement of the users' hand using motion sensors in the mobile device. Wang *et al.* show that their attack could accurately infer mm-level distance in more than 90% of the cases. Their attack utilizes these mm-level hand movement distance to decipher the users' pin. Another recent attack was proposed by Tang *et al.* [27] on the inference of number Pins by using accelerometer data. Their attack was shown to infer the users' pin with 70% and 85% accuracy in 10 attempts in user-independent and user-dependent environments respectively. Tang *et al.* claim that their attack could easily be used to attack the pattern lock system and requires minimal training with very few samples in case of user-dependent environments. By using a large number of password entries from 362 volunteer participants and their motion sensors data, Lu *et al.* [28] show that their attack could accurately infer the users' password in first 20 guesses.

Sensors based attack makes an assumption of having access to the user's smartphone sensor data [25], [26]. On the other hand, our method is solely based on the user's hand movement captured while the user type on their smartphone screen. The whole breed of sensor attacks is different from our attack model as our attack can be executed in such a way that does not require any information from user's smartphone.

III. Attack Details

In this section, we introduce the broader idea of the attack model and discuss each step in detail to decipher the password.

Figure 2 shows the workflow and steps of the attack. First, we summarize each step here and then discuss one by one later in detail.

Step 1 - Capture a video of the user's hands movements while he/she types the password on the mobile phones screen. One can easily guess the password typing as the first action by the user after picking the phone in his/her hands. Our method does not require any information from mobile phone screen display such as key popups or typed text to be recorded. However, our method assumes user's hand's movements and part of the back of the mobile phone are visible on the video clip.

Step 2 - Preprocess the recorded video and keep only the part of the clip of password entry process. In this process, attacker observes the recorded video frame by frame and cuts the extra part captured at the start and end of the video and

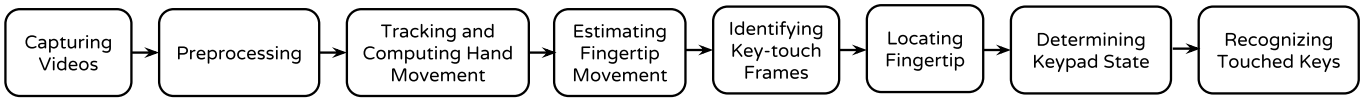


Fig. 2. Work flow of the attack model to decipher the smartphone password.

keeps only the password entry part. This step need not be very precise and the method works even with extra frames both at the start and end of the video clip. However, less number of extra frames results in less processing in further steps.

Step 3 - Employ TLD tracking tool [29], [30] to track an anchor hand point and an anchor point on the visible part of the mobile phone. This step requires the adversary to manually select the anchor points on the user's hand and mobile phone. We discuss criteria to select the anchor points in Section III-C. Having the tracked location of mobile phone-point and hand-point in each frame, the attack model computes the relative location of hand point with respect to the mobile phone anchor point.

Step 4 - Given the movement of hand point and some visible part of user's hand, estimate the shape and size of the hidden part of the user's hand. By using approximations of the measurements, estimate the user's fingertip movement.

Step 5 - Detect the key touch frames in which fingertip touches the mobile phone screen. By computing the image velocity corresponding to fingertip point and finding frames with zero velocity which are part of the sequence of consecutive frames in the video having a pattern of $\ldots, P, Z, \ldots, Z, N, \ldots$ in order where P represents positive, Z denotes zero, and N denotes negative image velocity.

Step 6 - Locate the fingertip in the detected key touch frames. Assuming the last key touch as OK key[1] and given the estimated movement of fingertip and known location of OK key on keypad, use backtracking to locate the fingertip on the mobile phone keypad.

Step 7 - Determine the keypad state based on the previous key touched and probabilities computed from a large password dataset.

Step 8 - Build a probability based password model with added information from determined touched location and keypad state and identify the touched keys. Given the correct locations of tracked points, we can reduce the search space to an average of 2-3 keys per touching frame. We discuss the eight steps in detail below.

A. Step 1 - Capturing Videos

An adversary captures the video of the user typing the password on the mobile phone screen. With the increasing use of mobile phones in public places such as conference halls, meeting rooms, shopping malls, airports, and clubs, etc., adversaries get a lot of opportunities to record a video of the user's typing. Also, adversaries can get access to the videos captured by surveillance cameras installed in the public places.

[1]Our method will need small geometric modifications for the mobile phones in which password entry process does not end with touching a specified key like OK key.

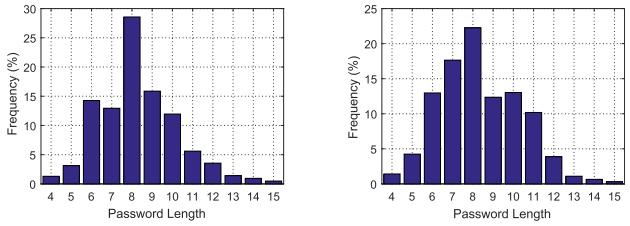


Fig. 3. Password length distribution in the selected password datasets for training and testing the attack performance. (a) Password length distribution in the *password list I, training password dataset*. (b) Password length distribution in the *password list II, testing password dataset.*

One can easily identify the password typing session from a video clip as the first thing a user will type on the mobile phone after picking his/her phone is the password if the mobile phone is secured by one. Our method uses the video clip of password entry process such that the user's hand movement and part of the mobile phone are visible. We collected the videos of the users' password entry process to evaluate the performance of our attack model.

In addition to the video data to evaluate the performance, we used a large password dataset to build the probability-based model. For the clarity of dataset used, we discuss the datasets in detail below.

Datasets: We used two different datasets. UNIQPASS v15 Password Dataset [31], a large password list, to build a probability-based model discussed in Step 7 and Step 8. Another is password entry video data set for computer vision based analysis and test the overall performance of our attack model. The detailed description of each dataset follows.

UNIQPASS v15 Password Dataset: From a large password list, UNIQPASS v15, with over 2 million unique ASCII passwords stored in random order [31], we created two mutually exclusive sublists of randomly selected passwords of length between 4 to 15 characters. These two sublists, *password list I* and *password list II*, have approximately 0.6 million and 0.1 million passwords respectively.

The passwords in both the lists, Password List I and Password List II, are unique and chosen randomly without repetition from UNIQPASS v15 selected password list. The password length distribution in both the lists is shown in Figure 3. The passwords length of eight characters were most frequently occurring in both of our password list: *password list I*, and *password list II*.

Video Dataset: Following approval from our university's Institutional Review Board (IRB), we collected two data sets; a training dataset comprising of 20 password entry videos from 10 volunteers, and a testing dataset comprising of 135 password entry videos from 45 volunteers (i.e., 3 videos per volunteer, with each video recorded using a different camera

TABLE I

SPECIFICATION OF THE CAMERAS USED IN OUR EXPERIMENTS FOR VIDEO RECORDING

Camera Configuration	Properties		
	Frame Rate	Resolution	Zoom
iPhone 6 Plus	240 fps	1280 x 720 p	Digital
Sony	30 fps	1440 x 1080 p	Optical
HTC One	30 fps	1920 x 1080 p	Digital

configuration). We used three cameras to record the videos — HTC One mobile phone camera, a Sony Camcorder, and an iPhone 6 plus camera. None of the volunteer participants were common among training session and testing session of data collection, i.e., the training data was collected from a separate set of volunteer participants. All participants were students, faculty or staff at our university. The volunteers entered passwords on a Samsung Galaxy S4 phone.

Table I summarizes the specifications of the three video recorders used in our experiments. We used the various specification of cameras to simulate a range of adversaries. An iPhone 6 plus camera enabled us to simulate an adversary who uses a high-end mobile phone camera with fairly good resolution and high frame rate of video recording. We used digital zoom capability of iPhone 6 plus camera to capture video from a distance as phone camera was not enabled with an optical zoom feature. A Sony camcorder illustrated the adversary who uses the video recorder with optical zoom feature and high resolution. Our last camera of HTC One mobile phone represented basic mobile phone cameras.

Password entry process and video recording: Users type their password on the mobile phone several times in a day and can type with no or very low cognitive load. Collecting the video recordings of the user's hand movement while they type their real password comes with a set of security risks. Hence, we worked around this problem by having each user to randomly select a password from the password list II and let them practice entering it on our Samsung Galaxy S4 phones until they felt comfortable typing it. On average, this practice session took 10-15 trials per password for each user. Following the practice session, each user then typed the same password during the recorded session. For each of the three camera configurations, each user typed a password that was preceded by a practice session. We performed this data collection experiment in various setup. For example, we recorded some users while they sat and others while they stood. Also, some users came to our lab for video recording, while others were recorded from inside or outside their residential halls. The recording was done from a distance of approximately 4-5 meters from the participants.

Video Dataset to Analyze the Effect of Parameters Settings: To analyze the effect of various parameters settings on our attack model, we recorded 42 videos for each user while the user typed the randomly selected password from password list II as follows: (1) 15 videos– with three different camera configurations and each camera configuration was used at five different distances of recording - each at 2 meters, 4 meters, 6 meters, 8 meters, and 10 meters; (2) 12 videos– with three different camera configurations and each camera configuration was used in four different lighting conditions - 250 lumens, 500 lumens, 750 lumens, and 1000 lumens; (3) 15 videos– with three different camera configurations and each camera configuration was used from five different angles of recording - each from -100deg, -50deg, 0deg +50deg, and +100deg. Angle is measured as the angle between the line of sight of the user's eye and negative of the direction of view of the recording camera. In total, we recorded 210 videos from 5 different users i.e. 42 password entry videos from each user.

B. Step 2 - Video Preprocessing

Given a video recording of a length of 5 to 10 seconds of the password entry process, we first cut out the video segment of interest. One can identify the part of password entry in the video segment, simply by observing the user's interaction with the mobile phone. We assume during the interaction with the mobile phone the first thing user types is the password if their mobile phone is secured by one. We used Windows Movie Maker [32] to select the password entry segment and cut and remove the unwanted part. The video segment of interest contains 200-300 milliseconds of recording both before and after the password entry process. Cutting the unwanted part from video need not be very precise for our attack to work, precision will just save the unnecessary processing in the later steps of attack process. We input the selected part of password entry video into AVS Video editor to extract the frames from the video. The video captured by Sony camcorder and HTC One mobile phone camera generated an average of 120 frames for each video. On the other hand, video captured by iPhone camera generated an average of 1000 frames per video.

C. Step 3 -Hand and Phone Anchor Points Tracking

We used the Tracking Learning Detection (TLD) framework [30] to track the anchor points on the user's hand and on the mobile phone in the captured video. TLD is an open source tool [29], and has three modules: a tracker which locates an object across the subsequent frames in the video, a detector which detects the presence of the object in each subsequent frames and a learner estimates the detectors error and update the template for the object to minimize the detection error for the subsequent frames.

TLD object tracking tool can detect the appearance of the object in the subsequent frames even after several failed detections. Also, it works quite well with a real-time stream of video [29]. We tracked an anchor point on the mobile phone visible throughout the video. This point can be present anywhere on the mobile phone. Also, we tracked five different anchor points on the users hand in the video. We selected such an anchor point on the user's hand that it fulfills the following criteria— 1). The hand anchor points should be visible throughout the password entry process in the video. 2). It should be as close to the mobile phone or the typing finger as possible. We believe that the point which is closer to the typing finger gives a more plausible representation of fingertip movement. We tracked one point on the mobile phone

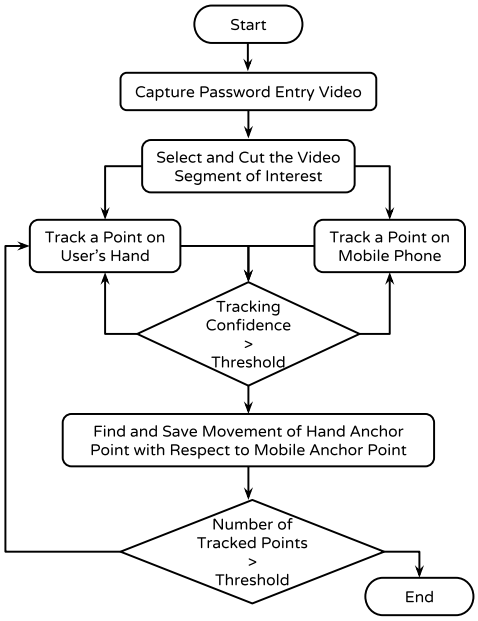


Fig. 4. Flow chart summarizing video preprocessing and hand and phone anchor point tracking steps.

as we observed that the mobile phone anchor point is very different from the rest of the objects in the video and hence tracking algorithm gives very accurate tracking results for the mobile phone point. Although, we select hand anchor point which seems different from other objects in the frame, but tracking algorithm gets confused many times even with a little bit similar looking other points present on the user's hand. So, we tracked multiple points on the hands and merged the results as to minimize the tracking error.

Figure 4 summarizes the preprocessing and tracking steps. For a given frame, if the tracking confidence[2] was less than a set threshold for more than 10% of the frames, we selected another point and reran the entire tracking operation for that particular video. We set a tracking confidence threshold as 0.5 as we found it produces good tracking results in our sample of the training dataset.

We compute the relative movement of hand point $S_{hand\,Point}$ in the frame of reference of tracked mobile phone point simply by subtracting the pixel locations of hand point from the pixel locations of mobile phone anchor point obtained from tracking.

D. Step 4 - Estimating the Fingertip Movement

We observed that the user's finger first moves towards the screen, rests, and then moves away from the touch screen while typing any particular key. We also observed that the rest of the hand also follows the similar behavior of motion (i.e. the direction of motion of other points on the hand is same as the direction of motion of fingertip. Also, the magnitude of movement of any other point on the hand is directly proportional to the magnitude of fingertip movement.). We establish a relation

[2]Tracking framework gives a tracking confidence for each frame in the video as a measure of how accurate the detection of the object is for the corresponding frame. For details on this, the user is referred to [29].

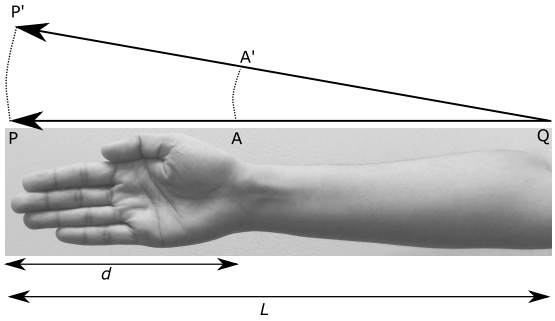


Fig. 5. Illustrating the transformation of hand point movement to fingertip movement. Point A shows the tracked point with movement $S_{hand\,Point} = AA'$ between two given frames. Figure shows the corresponding movement in typing fingertip P as $S_{fingertip} = PP'$.

between the movement of the fingertip and the movement of a visible point. For simplicity, we assume that user will not apply any other external force (such as force applied over the wrist of the user's hand) during the typing process. We build a hand movement model as an approximation of the movements of user's hand while they type their password. Given the movement $S_{hand\,Point} = AA'$ of a tracked point A to A' on the user's hand and d, the distance of the hand point A from the fingertip P (see figure 5), then the movement $S_{fingertip} = PP'$ of fingertip point P corresponding to hand point movement AA' can be estimated by applying equal proportion of sides rule[3] on similar triangle $\triangle PQP'$ and $\triangle AQA'$ as follows—

$$\begin{aligned} S_{fingertip}/S_{hand\,Point} &= PP'/AA' \\ &= PQ/AQ \\ &= L/(L-d) \end{aligned} \tag{1}$$

where L is the length of hand (length from elbow to fingertip) which can be estimated[4] using the visible part of the hand in the video.

E. Step 5 - Identifying Key Touch Frames

Key touch frames are the frames which represent the finger touching the mobile phone screen. Using the estimated movement of the fingertip (see Section III-D), we computed the image velocity[5] corresponding to fingertip point [36]. As per well-known notion of typing process on the mobile phone screen, fingertip moves towards the screen, rests and then moves away from the screen [6], [16]. We first looked for the sequences of consecutive frames in the video having positives, zeros, and negatives image velocity pattern in order. Such a sequence represents the corresponding movements of the fingertip to type a key on the screen. The frames which show zero velocity and are part of selected frames sequence represent *the resting part (or key touch frames)* of the process

[3]In two similar triangles, corresponding sides are all in the same proportion. This rule is also called Thales theorem or Intercept theorem [33].

[4]For an average person, length of the forehand is approximately 1.6 times of the length from wrist to fingertip. For details the reader is referred to [34], [35].

[5]Pixel movement of fingertip point between two consecutive frames in the video segment.

of typing a key. Hence, we selected and referred the set of such frames as candidate key touch frames F_i.

There can be some error associated in the tracking step or in the estimation of fingertip movement for an anchor hand point. Hence, to reduce the error in the estimates, we repeated the whole procedure of inferring candidate key touch frames for five different hand anchor points. The process gave us five non-mutually exclusive set of candidate key touch frames (say F_1, F_2, F_3, F_4, and F_5) each for a hand anchor point (see Section III-C).

To identify the final set of key touch frames, we form groups of all the candidate key touch frames for the target video. Suppose $F_{candidate} = \{f \mid f \in F_i,\ i \in [1, 5]\}$ is the set of all candidate key touch frames, then the groups were created for all the frames $f \in F_{candidate}$ as follows—

1) Distance between two farthest frames in any group must be less than the set threshold Th_{max}.
2) Distance between two closest frames in two different groups must be greater than the set threshold Th_{min}.

Analysis to decide the minimum and maximum threshold is explained under the threshold selection heading.

In this step, the frames which are the member of set $F_{candidate}$ due to error (False Positive) in preceding steps, will form a group with only a few frames. The rationale behind this idea is that if a tracked point misses the detection of a key touch frame f_i (False Negatives), it will be detected corresponding to other tracked points. On the other hand, if a frame f_j was detected as key touch frame due to error (False Positives), it will not be there for other tracked points. Hence, the higher number of frames in a group shows a true detection (True Positives) of the key touch frame.

Hereafter, we will remove those groups of frames in which the number of frames is less than a set threshold Th_{frames} from the set of candidate key touch frames. Also, all the frames in a group represent one key touch event and hence one frame per group will suffice to create the set of key touch frames. Hence, we form a new set of key touch frames $F_{keyTouch}$ containing the median of the remaining groups.

Figure 6 shows an example of grouping where set of frames corresponding to five different tracked hand anchor point are as follows.

$$\begin{aligned} F_1 &= \{12, 14, 39, 42, 63, 63, 79, 102, 117, 121\}, \\ F_2 &= \{13, 14, 39, 43, 62, 26, 77, 103, 104, 118\}, \\ F_3 &= \{12, 13, 40, 61, 65, 102, 104, 119, 120\}, \\ F_4 &= \{13, 28, 39, 65, 63, 65, 102, 104\}, \\ \textit{and}\ F_5 &= \{26, 40, 62, 65, 81, 101, 102, 104, 117, 121\}. \end{aligned}$$

Figure 6a shows seven different group of candidate key touch frames G_1, ..., G_7 where the median of groups G_1, G_3, G_4, G_6, and G_7 represent the final key touch frames i.e. $F_{keyTouch} = \{13, 40, 63, 102, 119\}$. Whereas, frames belonging to groups G_2 and G_5 indicate the error in detection as less than five frames belong to these groups, hence we will remove all the frames from groups G_2 and G_5 for further analysis.

Threshold Selection: We extracted key touch frames manually by observing the videos frame by frame from the training

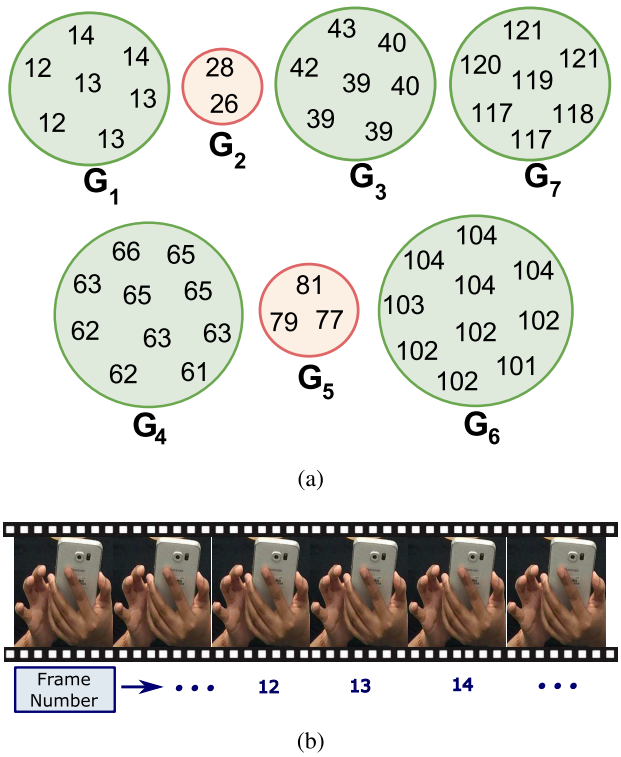


Fig. 6. Figure 6a shows seven different groups of candidate key touch frames. Groups G_2, and G_5 has only three and two candidate frames respectively, hence represent the false detection of candidate key touch frames. Median of groups G_1, G_3, G_4, G_6, and G_7 constitute the set of final key touch frames $F_{keyTouch}$. Figure 6b shows the video frames represented by Frame # in Figure 6a. (a) Grouping of candidate key touch frames $f \in F_i$, $i \in [1, 5]$ to obtain a final set of key touch frame. (b) Frames of a user's recorded video clip. Frame #12, #13, and #14 are selected as key touch frames in group G_1.

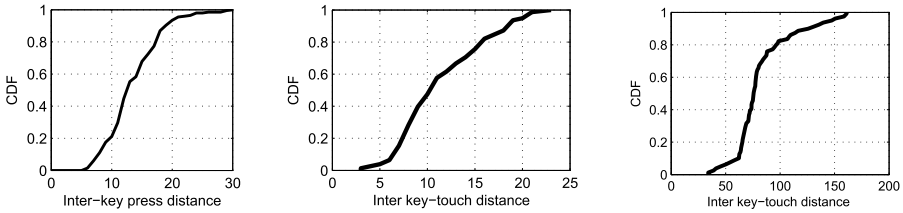
TABLE II

TABLE SUMMARIZING THRESHOLD VALUES SET BY ANALYZING VIDEO TRAINING DATA SET FOR ALL THREE CAMERA CONFIGURATIONS

Camera Configuration	Thresholds		
	Th_{max}	Th_{min}	Th_{frames}
iPhone 6 Plus	150	50	30
Sony	20	6	6
HTC One	20	6	6

dataset of 20 videos (see Section III-A). We calculated the difference between the number of frames between two key touch frames and plotted the cumulative distribution function (CDF) (see Figure 7) of inter-key touch frames for all three types of camera configurations used in our analysis. Figure 7b and Figure 7a show that the distribution of inter-key touch frames for the HTC One phone camera and the Sony Camera is the same. Note that the average frame rate of video recording in the Sony camera and the HTC One phone camera was equal. Figure 7c shows that for more than 95% of the consecutive key touch pairs, inter-key touch frames falls between 50 to 150 frames for videos captured from iPhone 6 plus camera. Similarly, figure 7b and 7a show that more than 95% of the key touch pairs, inter-key touch frames falls between 6 to 20 frames for videos captured from Sony and HTC One phone cameras. We used the above measures to set Th_{min} and Th_{max} corresponding to different camera configuration.

To set the threshold Th_{frames}, we performed an empirical analysis of the videos from our training dataset. For our



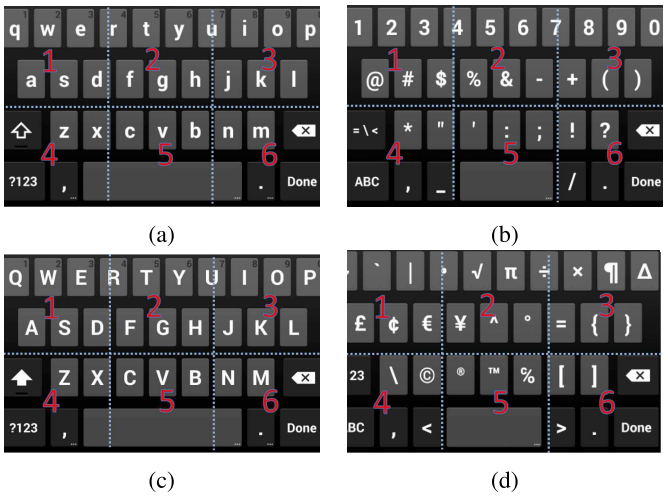
(a) (b) (c)

Fig. 7. cdf plot of inter key touch frames for training videos captured using variety of cameras used in our experiments. (a) HTC One Phone. (b) Sony camcorder. (c) iPhone 6 Plus camera.

TABLE III

ASSIGNMENT OF KEYS TO THE MUTUALLY NON- EXCLUSIVE CLUSTERS FOR EACH STATE OF THE KEYPAD. TABLE SHOWS SIX CLUSTERS $C_i's$ OF KEYS FOR EACH STATE S_i OF THE KEYPAD

Keypad State	Set of keys belonging to different clusters					
	C_1	C_2	C_3	C_4	C_5	C_6
S_1	q, w, e, r, a, s, d	r, t, y, u, f, g, h	u, i, o, p, j, k, l	*shift*, z, x, *number*, *space*	c,v, b, *space*	n, m, *back*, *space*, *Done*
S_2	Q, W, E, R, A, S, D	R, T, Y, U, F, G, H	U, I, O, P, J, K, L	*shift*, Z, X, *number*, *space*	C,V, B, *space*	N, M, *back*, *space*, *Done*
S_3	1, 2, 3, 4, @, #, $	4, 5, 6, 7, %, &, -	7, 8, 9, 0, +, (,)	*shift*, *number*	*space*	*back*, *Done*
S_4	-	-	-	*shift*, *number*	*space*	*back*, *Done*



(a) (b) (c) (d)

Fig. 8. Samsung Galaxy S4 keypad used for alphanumeric password entry. In Figure 8a, state S_1 of keypad with alphabets in lowercase. In Figure 8c, state S_3 of keypad with numeric keys and a particular set of special characters.In Figure 8b, state S_2 of keypad with alphabets in uppercase. In Figure 8d, state S_4 of keypad with another set of special characters. (a) keypad state S_1. (b) keypad state S_2. (c) keypad state S_3. (d) keypad state S_4.

training dataset $Th_{frames} = 6$ for Sony camcorder, and HTC One Phone camera performed well in detecting true key touch frames. Similarly, $Th_{frames} = 30$ for iPhone camera detected true key touch frames accurately. We noticed low false positive rate (FPR) and high true positive rate (TPR) for the set threshold values in our training dataset. Table II summarizes the threshold values for all three cameras used in our experiments.

F. Step 6 - Estimating the Location of Touch on Mobile Phone Screen

We map the fingertip location to the actual location of the touch on the keypad for each frame in the list of final key touch frames $F_{keyTouch}$. In the standard keypad, used for password entry, the proximity between the keys is very high due to which it is very easy to confuse with neighboring keys locations. Hence, we divided the keypad screen into non-mutually exclusive clusters of keys as shown in the Table III. The cluster assignment was done based on the geometrical distance of the keys such that spatially closed keys belong to the same cluster. For example, the keys Q and W are spatially close to each other and hence belong to the same cluster. On the other hand, the key A and B are far from each other on the keypad (we used the standard $QWERTY$ keypad for our analysis (see figure 8)) and hence will not be part of the same cluster.

The clusters assignments of keys is shown in Table III, exhibiting 6 clusters corresponding to each of the keypad state S_1, S_2, and S_3 of the keypad (see Figure 8). For the keypad state S_4, there are only 3 clusters as the keys in the proximity of rest 3 cluster locations were not allowed to be used in a password. Figure 8 shows four different states of the keypad where S_1 represents default state of the keypad with all the keys in lowercase. We, now, estimate the location of the fingertip in a key touch frame as one of the key cluster location. For example, for the video for the password $Sec@16$, we detect the fingertip location as C_1 and C_5 for the first and third key touch frame respectively. Details of estimating the fingertip location as a key cluster location are as follows.

Mapping the fingertip location to specific cluster of keys: By this stage of the attack process, we have a final array of key touch frames $F_{keyTouch}$ and relative pixel location (x_i, y_i) of fingertip corresponding to each key touch frame. $F_{keyTouch}$ is the set of frames $f_1, f_2, \ldots, f_n$ representing n key presses and $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ be the corresponding relative pixel location of fingertip point. We utilized these pixel locations to compute the movement of fingertip from frame f_i to frame f_j in X and Y direction as — $X_{fingerTipMovement_{ij}} = x_i - x_j$ and $Y_{fingerTipMovement_{ij}} = y_i - y_j$ respectively.

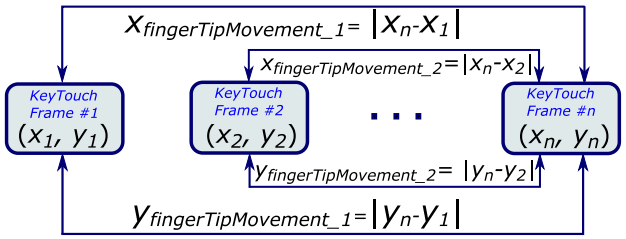


Fig. 9. Graphical representation of Algorithm 1 for cluster assignment corresponding to key touch frames. (X_i, Y_i) shows the estimated pixel location of fingertip point. $X_{fingerTipMovement}$ and $Y_{fingerTipMovement}$ is movement of fingertip for i^{th} frame to last frame. f_n shows the last key touch frame which corresponds to OK key pressed.

We computed the fingertip movement for every pair of key touch frames f_i and $f_n \in F_{keyTouch}$, where f_n is the last frame as shown in Figure 9. The reason for computing the fingertip movement for all the key touch frames with respect to fingertip location in the last key touch frame f_n is the fixed last key touched as $OKkey$ and the known corresponding physical location of $OKkey$ on the keypad for the password entry process. Hence, fingertip location in the last key touch frame serves as a frame of reference for the estimation of movement of the fingertip for a key touch frame and helps estimate the corresponding key cluster.

We followed the following steps to map the key touch frames to the clusters of corresponding key location —

1) First step is to calculate the magnification.[6] We estimated the width (or length) of the mobile phone in the video using the visible part of the phone and calculated the ratio with the actual known width (or length) of the mobile phone keypad.
2) Based on the calculated magnification, we estimate the dimensions of the keypad in the video. Width and length of mobile phone in the video $w_{keypad} = magnification \times actualWidth$ and $l_{keypad} = magnification \times actualLength$. We set the threshold Th_w and Th_l as $w_{keypad}/3$ and $l_{keypad}/2$ respectively as we divided the keypad width into 3 clusters and keypad length into 2 clusters of spatially close keys.
3) Last step is to assign the cluster location to each key touch frame. In this process, we took the advantage of known phenomenon that the last key pressed while entering the password is 'OK key'. The user touches the 'OK key' ($Done$ or $Enter$) as a last key in the password entry process. Hence, we assigned the fingertip location in the last key touch frame as C_6 corresponding to 'OK key' (see Table III). Then, based on the magnitude of fingertip movement in X and Y direction with respect to fingertip location in last key touch frame, we assigned the corresponding cluster of keys. Algorithm 1 describes the basic substeps in the cluster location estimation.

Algorithm 1 compares the fingertip movement in X and Y direction for each key touch frame with Th_w and Th_l to decide the key touch location and assigns the corresponding cluster to the frame. $X_{fingerTipMovement}[i]$ and $Y_{fingerTipMovement}[i]$ are the fingertip movement for i^{th} frame with respect to the last key touch frame in X and Y direction respectively. Th_w and Th_l are the thresholds computed based on the estimated length and breadth of keypad in the video. This step finally results in cluster location C_i for each key touch frame. We proceed to the next step if the cluster location inferred for the frame is C_4, otherwise we jump to Step 8 to recognize the actual key pressed.

[6]The ratio of the size of an object in the video to its actual size.

Algorithm 1 Determining Key Cluster for Each Key Touch Frame.

Input: $F_{keyTouch}[]$
Input: $X_{fingerTipMovement}[]$
Input: $Y_{fingerTipMovement}[]$
Input: Th_w
Input: Th_l
Input: $Clusters[] \leftarrow [C_1, C_2, C_3; C_4, C_5, C_6]$
for $i \leftarrow 1$ **to** $Length(F_{keyTouch})$ **do**
  if $X_{fingerTipMovement}[i] < Th_w$ **then**
    $Cluster_{col} \leftarrow 3$
  else if $X_{fingerTipMovement}[i] < 2 \times Th_w$ **then**
    $Cluster_{col} \leftarrow 2$
  else
    $Cluster_{col} \leftarrow 1$
  end
  if $Y_{fingerTipMovement}[i] < Th_l$ **then**
    $Cluster_{row} \leftarrow 2$
  else
    $Cluster_{row} \leftarrow 1$
  end
  $Cluster[i] \leftarrow Clusters\ [Cluster_{row}, Cluster_{col}]$
end

G. Step 7 - Determining Keypad State

State transition diagram: State of the keypad changes from one state to another state based on the key touched by the user. For example, if user presses $shift$ key in the S_1 state of the keypad (see Figure 8), state of the keypad changes from state S_1 to state S_2. Keypad state S_1 and state S_2 represents the keypad with alphabets in lowercase and uppercase respectively (see Figure 8a and 8c). Similarly, state S_3 and state S_4 represents the keypad with numbers and special characters respectively (see Figure 8b and 8d). We define three types of key touch events and brief description of transition of keypad states as follows—

1) Kp_1 represents the key location touched by the user corresponding to $shift$ key. Kp_1 key helps the user to switch between lowercase and uppercase state of keypad while the keypad is in alphabets states. Also, it switches between numbers and special characters when the keypad is in number/special-character states.
2) Kp_2 represents the key location touched by the user corresponding to $number123$ key. Kp_2 key helps the user to switch between letters and numbers.
3) Kp_3 symbolizes any key touched by the user except the key locations corresponding to Kp_1 and Kp_2.

Figure 10 shows detailed state transition diagram for password entry keypad used in our experiments. To decide a

particular key touch is Kp_1 key or Kp_2 key having previously determined the cluster location as C_4, we compare the previously calculated probabilities using a generic password dataset *Password List I* as follows.

$$P(Key_j = Kp_1) = (P(Kp_1|PreKey) + P(Kp_1|PreState))/2$$
$$P(Key_j = Kp_2) = (P(Kp_2|PreKey) + P(Kp_2|PreState))/2$$
$$P(Key_j = Kp_3) = (P(Kp_3|PreKey) + P(Kp_3|PreState))/2$$

Where, $PreKey$ represents the key pressed in the previous frame and $PreState$ denotes previous state of the keypad before this key-press. $PreKey$ and $PreState$ are distinct because $PreState$ represents the state of the keypad among the four possible states (See Figure 10) whereas $PreKey$ represents the key touched in the previous state, i.e., in $PreState$. The probability of the current key touched being Kp_i depends on the previous key touched and the previous state of the keypad. We say the key Kp_i is pressed in j^th key touch frame if the computed probability $P(Key_j{=}Kp_i)$ is maximum for all $i \in \{1, 2, 3\}$. The keypad state is determined based on the Kp_i and previous state based on state transition diagram shown in Figure 10.

For example, to enter password $sec18$, a user would need to type following key sequence: $s- > e- > c- >$ ***Number(***Kp_2***)***$- > 1- > 8- > OK$. The highlighted key entry is corresponding to key cluster C_4. We compute probabilities $P(Kp_i|PreKey = c)$ and $P(Kp_i|PreState = S_1)$ using our training password dataset to determine that the fourth key touched is $Kp_1, Kp_2, or\ Kp_3$,. In this example $P(currentKey = Kp_i|PreState = S_1)$ would account for all the occurrences of Kp_i after any key from keypad state S_1 whereas, $P(Kp_i|PreKey = c)$ would account for all the cases where key Kp_i is pressed after key c being pressed.

H. Step 8 - Recognizing Touched Keys

In the last stage of our attack process, we have final key touch frames $F_{keyTouch}[]$ and the corresponding cluster assignment $Cluster[]$ from step 6. All the keys in the cluster C_i for the determined keypad state are the candidate keys. Hence, there are more than one candidate key for each key touch frame. For example, for ith key touch frame, if the cluster assignment is C_j, then all the keys in the cluster C_j have some probability to be actual key pressed by the user in that particular frame. We calculated the probabilities for each key to be the actual key touched by user and assigned the weight in the order of probability. We computed and designed our probability model using the *password list I* (see Section III-A). We take very generic case of n keys pressed by the user while entering the password to describe our model. Suppose, we have an array of key touch frames and an array of cluster assignment both of size n. Following are the steps, we used further to infer the key pressed by user.

1) In the password entry process, last key press is OK key. We assign the probability of the i^{th} key touched to be 'OK' as —

$$P(Key_i = OK) = \begin{cases} 1, & \text{if } i = n \quad (2a) \\ 0, & \text{otherwise.} \quad (2b) \end{cases}$$

and

$$W(Key_i = OK) = P(Key_i = OK) \quad (3)$$

2) We calculate the probability $p_i(A_j)$ of a key A_j to be the key touched by the user in the i^{th} key touch frame as —

$$p_i(A_j) = P(Key_k = A_j|Key_{k+1} = A_{j+1}) \quad (4)$$

Where $A_j \in C_i$ and Key_k denotes the key touched in the k^{th} frame and k is any frame number. We selected only top five keys in order of probability as the candidate key presses corresponding to the i^{th} key touch frame.

3) We build a graph using the top five keys corresponding to each key touch frame. Figure 11 shows a graph for an example password input $Sec@16$. If a key appears two or more times in the top five probable keys, we summed up all the probabilities for that particular key. For example, to compute the total weight for the 3^{th} key press in the graph shows character W, E, and S with weights 0.49, 0.36, and 0.15 respectively shown with the characters in the brackets. The weight for character W is the sum of the probabilities $P(Key_k = W|Key_{k+1} = C)$ and $P(Key_k = W|Key_{k+1} = V)$.
4) For any occurrences of cluster C_4 we refer to Step 7 for weight calculations and determining the keypad state.

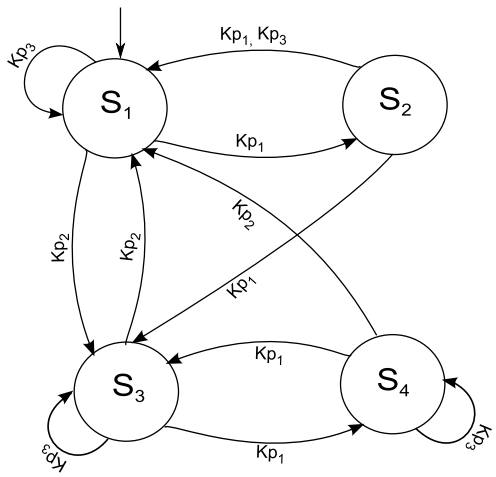


Fig. 10. Keypad state transition diagram. State S_1, S_2, S_3, and S_4 are shown in figure 8.

For the shown example password entry in Figure 11, the first guess will be $Swc@16$ which shows 83% accuracy of characters guessed in the password. We generated top 5 five and top 10 guesses in order of total weight of the guessed password. For example next two guesses for the entered password will be $Swc\$16$ and $Swc\#16$ which shows 67% and 67% of the accuracy of predicting characters in the password.

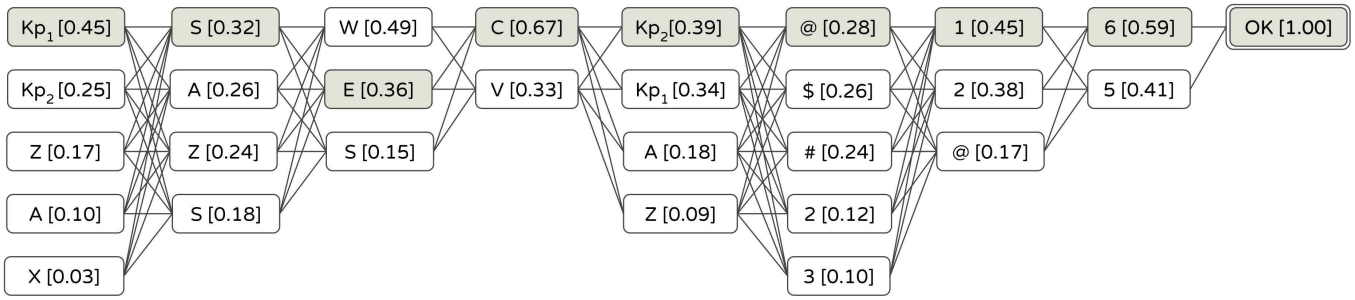


Fig. 11. Probability model graph, based on the top five most probable keys corresponding to each key touch frame. Figure shows the graph for password entry which needs 9 key presses. Last key pressed is OK key with probability 1. Probability to be key pressed is shown under the character label for each node in the graph. We generate the passwords from this graph as to maximize the total weight for the whole password string, given the weight given corresponding to each key label. The graph is corresponding to password entry video analysis for the typed password $Sec@16$ and correct key press sequence for the password entry is shown with gray colored boxes.

TABLE IV

KEY TOUCH FRAME DETECTION CONFUSION MATRIX FOR VIDEO RECORDING USING IPHONE 6 PLUS CAMERA

Observed in the Video	Identified by Our Model	
	Key Touch Frame	No Key Touch Frame
Key Touch Frame	99.7 %	0.3 %
No Key Touch Frame	1.6 %	98.4 %

TABLE V

CLUSTER PREDICTION CONFUSION MATRIX FOR IPHONE 6 PLUS CAMERA

Actual Clusters	Predicted Clusters (Expressed as %)					
	C_1	C_2	C_3	C_4	C_5	C_6
C_1	96.0	1.4	0.0	0.6	0.0	0.0
C_2	0.9	94.0	3.4	0.0	1.7	0.0
C_3	0.0	2.2	96.1	0.0	0.4	1.3
C_4	2.5	0.0	0.0	96.8	0.7	0.0
C_5	0.0	0.0	0.0	0.0	100	0.0
C_6	0.0	0.0	0.0	0.0	0.0	100

IV. PERFORMANCE EVALUATION

A. Key Touch Frame Detection

A major step in our attack model is to detect the frames where a key is being touched. Once we estimate the fingertip movement of the typing hand using the tracked hand point, we detect the key touch frames. We observed each video manually to label every frame in the video as 'key touch frame' or 'no key touch frame' as our ground truth information. We compared our ground truth label with that of identified by our key touch frame detection model. Table IV shows the confusion matrix for detected key touch frames using iPhone 6 Plus camera (best performing camera in our experiments) for video recording. The attack model could identify key touch frames correctly in 99.7% cases with a false positive rate of 1.6% and false negative rate of 0.3%.

B. Key Cluster Prediction

Our attack process first estimates the cluster in which the actual key touched belongs. Hence, it is interesting to observe the accuracy of estimation of the cluster for each key press. Table V and Table VI show the confusion matrix for cluster prediction for iPhone 6 plus camera and Sony camcorder used in our experiments. The prediction accuracies in the tables are expressed as percentages — e.g., the password entry videos recorded by the iPhone 6 plus camera show an average accuracy of, 97.15% for key cluster prediction. For Sony camcorder, we achieved an average accuracy of, 94.42% for key cluster prediction.

TABLE VI

CLUSTER PREDICTION CONFUSION MATRIX FOR SONY CAMCORDER

Actual Clusters	Predicted Clusters (Expressed as %)					
	C_1	C_2	C_3	C_4	C_5	C_6
C_1	92.3	3.1	0.0	4.6	0.0	0.0
C_2	2.9	91.0	3.0	0.0	3.1	0.0
C_3	0.4	2.4	94.5	0.0	1.0	1.7
C_4	3.6	1.4	0.0	93.7	1.3	0.0
C_5	0.0	0.0	0.0	1.7	96.0	2.3
C_6	0.0	0.0	0.0	0.0	1.0	99.0

C. Keypad State Determination - Prediction of $Shift$ and $Number$ Key

Prediction of $Shift$ and $Number$ keys are very critical for our attack process as these two keys are responsible for the change of the state of the mobile phone keypad. Detail of the transition of the keypad state is described in detail in Section III-F. Table VII shows the confusion matrix for the prediction of $Shift$ and $Number$ key corresponding to 1 guess, 5 guesses and 10 guesses for the video recording using iPhone 6 plus camera.

D. Character Prediction Accuracy

Character level accuracy of predicting password gives a consolidated impression of the strength of our attack model. Using the test dataset of 135 password entry videos, we achieved an average prediction accuracy of 70.9% of characters per password in 10 guesses using iPhone 6 plus camera. We were able to accurately predict an average of 68.4% and 66.0% of characters per password in 10 guesses for the videos captured using Sony camcorder and HTC One phone camera respectively. Table VIII shows the percentage of the average number of character predicted per password in one, five, and ten attempts for all three camera recordings used in our experiments.

E. Effect of Various Parameter Settings

Figure 12 shows the effect of lighting conditions, recording distance, and angle of recording on the performance

TABLE VII

CONFUSION MATRIX FOR PREDICTION OF *Shift* KEY AND *Number* KEY FOR IPHONE 6 PLUS CAMERA

Actual Key Pressed	**Predicted Key (Expressed as %)**					
	1st Guess		**5 Guesses**		**10 Guesses**	
	Shift **Key**	*Number* **Key**	*Shift* **Key**	*Number* **Key**	*Shift* **Key**	*Number* **Key**
Shift **Key**	92.0	8.0	96.0	4.0	98.0	2.0
Number **Key**	9.0	91.0	7.0	93.0	1.0	99.0

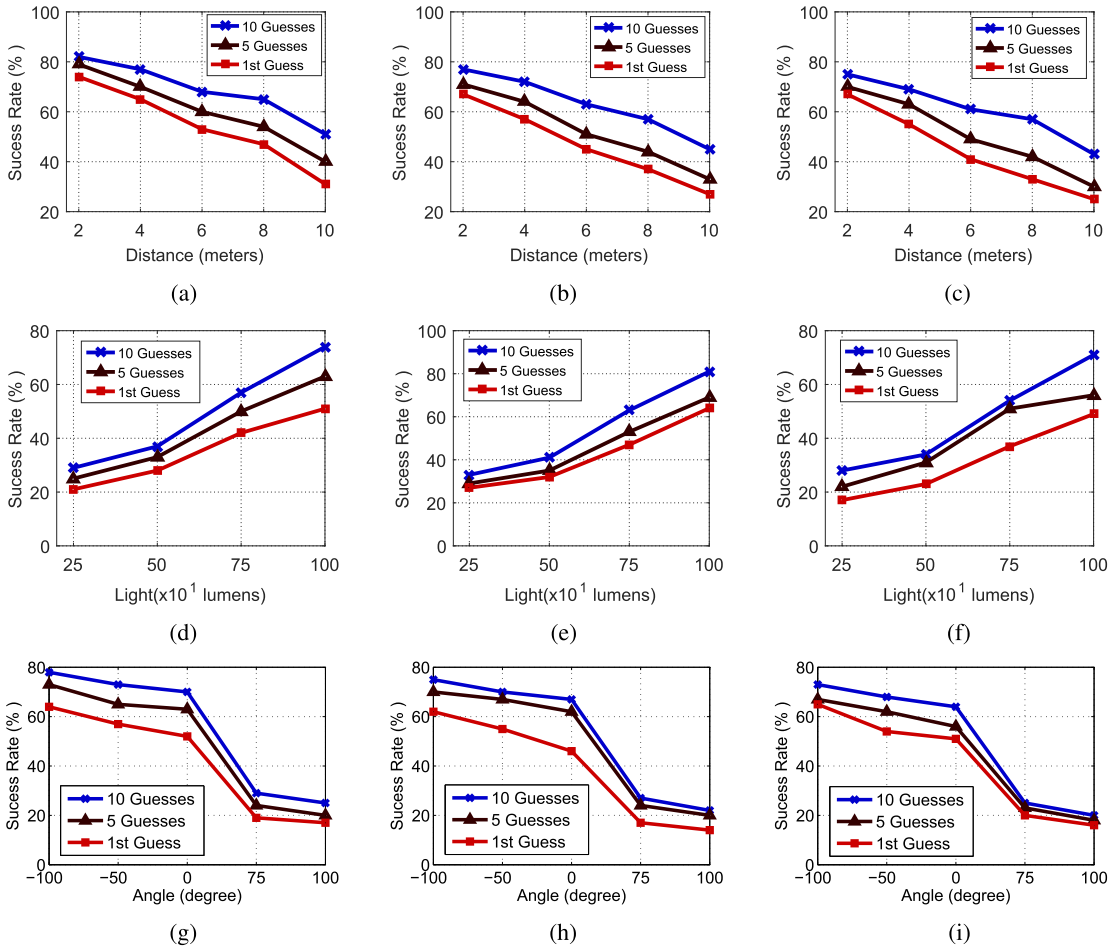


Fig. 12. Plots showing the effect of various parameters settings on the performance of our attack. Y- axis shows the percent of average number of characters per password correctly recognized and X axis shows the various parameter settings. (a) Sony Camcorder. (b) iPhone 6 Plus Camera. (c) HTC One Phone Camera. (d) Sony Camcorder. (e) iPhone 6 Plus Camera. (f) HTC One Phone Camera. (g) Sony Camcorder. (h) iPhone 6 Plus Camera. (i) HTC One Phone Camera.

TABLE VIII

ACCURACY OF PASSWORD PREDICTION EXPRESSED AS PERCENTAGE OF CHARACTERS PER PASSWORD

Camera Configuration	**Accuracy (expressed as %)**		
	1st Guess	**5 Guesses**	**10 Guesses**
iPhone 6 Plus	46.8	62.9	70.9
Sony Camera	42.4	58.7	68.4
HTC One	43.9	55.4	66.0

of our attack. Figure 12a-12c show the effect of recording distance on the performance of our method. Optical zoom capability of the camera was not used to record the videos. The figure shows the prediction accuracy decreases with the distance of the camera from the victim. Figure 12d-12f shows the effect of different lighting conditions on the performance of our attack. The attack works better in better lighting. Also, we observed that no significant results were obtained with video recording in lighting condition less than 250 lumens. Figure 12g-12i shows the effect of angle of recording on the performance of our method. We observed that the prediction accuracy was very low when the videos were recorded from the right-front angle from the user. The reason behind low accuracy is probably the typing behavior of users in our dataset. The majority of the users in our dataset typed the passwords using their right hand and hence any part of the palm of the typing hand was not visible. In such scenarios, method requires tracking any visible point on the forehand which resulted in a less accurate estimate of fingertip motion due to a relatively longer distance from the hand anchor point. The attack works better with video recordings captured from the left-front to center-front from the user.

F. Effect of Screen Size of the Victim's Phone

The results based on the analysis on the videos recorded while the users typed on Samsung S4 phone which has a smaller screen as compared to Samsung S9+ provides a *more stringent* test environment. A brief experimental substantiation is given in Appendix A.

The phones with bigger screen sizes such as Samsung S9+, generally have bigger keypad size. Larger keypad size provides the flexibility of dividing it into more numbers of spatially close key clusters and thus provides more information from the video analysis part of the attack. Each key cluster would contain fewer numbers of keys and thus would result in a smaller search space for keys touched. The small modification to the attack model for smartphones with bigger keypad size results in a significantly smaller search space.

For the screen size of the target phone similar to the phone such as Samsung S4, we performed an extensive analysis to find optimal division of the keypad into key clusters. The keypad divided into 2 rows and 3 columns i.e. 6 clusters gave the best results in our training dataset. In other keypad division settings, we observed an increase in the error either for the typed key identification within the cluster keys or the error for the cluster detection.

We obtained an average increase of 29.6% for the first guess, 16.3% for the first five guesses, and 17.3 % for the first 10 guesses in the character prediction for the passwords typed on the Samsung S9+ phone compared to Samsung S4 (see Appendix A).

G. Discussion – Attack Performance

Because the operating conditions of our attack model are very stringent compared to other existing works, it is hard to compare the performance of our method with them. To analyze the significance of our attack model, first we present a comparative analysis showing the difference and level of difficulty of the scenario studied. We, then, calculate the entropy and information gain and compare the same with random attack. We also present the analysis and show the reduction of search space in the order of magnitude induced by our attack model.

1) ***Comparison of Our Method with Existing Work:*** We compare our attack model in various parameters such as operating conditions, unobtrusiveness, and underline assumptions about the structure of password. Table IX presents the comparative study of our attack model.
2) ***Entropy and Information Gain:*** We estimate the information gain about the character typed revealed by the captured video clip and compare the same with random guess. If we select a character c uniformly at random from the character set Q, and if the attacker does not get any additional information, the entropy of the probability distribution of the character is –

$$H_0[c] = -\sum_i Pr[i] log_2\ Pr[i]$$

where i=p,q, and $Pr[p]$ is the probability that the selected character is correct and $Pr[q]$ is the probability that the selected character is not correct. On a standard keyboard, there are an average of 62 characters including numbers and special character to be selected as one of the character member of a typed password. Hence the probability of a randomly selected character being correct is $Pr[p] = 1/62$. Similarly, there are 61 possibilities which represent the selected character will be incorrect and so the $Pr[q] = 61/62$.

If the attacker get the video clip and learns the character $c \in Q_0$, where Q_0 is the subset of character set Q, the estimated entropy of the probability distribution of character is –

$$H_1[c|(c \in Q_0)] = -\sum_i Pr[i|c \in Q_0] log_2\ Pr[i|c \in Q_0]$$

where i = p,q, and $Pr[p|c \in Q_0]$ is the probability that the selected character is correct given that character is in subset Q_0 and $Pr[q|c \in Q_0]$ is the probability that the selected character is not correct when selected randomly from the subset Q_0. In our attack model $Pr[p|c \in Q_0] = 0.97 \times 1/6$ for iPhone camera. 0.97 is the average probability to predict key cluster correctly (see Table V) and 1/6 is the probability to select a character from the selected key cluster. There are an average of 6 keys in a cluster (see Table III). Similarly, we can calculate the probability for Sony camcorder using Table VI. Average probability to select a cluster correctly is obtained by averaging the diagonal elements in the cluster prediction confusion matrix Table V and Table VI.

The information gain induced by the learning from the video clip is the difference between the two entropies i.e.

$$InfoGain = H_0[c] - H_1[c|(c \in Q_0)]$$

In a similar fashion we compute the probability of predicting the characters in a password correctly using our attack model. Table VIII shows the average number of characters predicted correctly in the users' password using our attack model for different camera settings. Table X presents the entropy for a character in the password in our method and compares the same with the entropy of a character in a random guess. Table also presents the information gain per character using our methods.

3) ***Comparing Performance with an Exhaustive Search:*** Our attack model show over 99% reduction in search space compared to an exhaustive search. We evaluated the performance of our method and found an average of 2-3 candidate keys for each key press in the password. We show the comparison of search space for a password of an average length i.e. password of 7-8 characters long [20], chosen randomly from all letters and number keys. For example, for a password chosen randomly from all lower-case letter keys, upper-case letter keys, and number keys, an attacker would need to try $62^8/2$ candidate passwords on average before he/she finds the correct one. On the other hand, using our attack model, the attacker would only need to try an average of 2-3 candidate keys for each character in the password. This

TABLE IX
COMPARISON OF OUR ATTACK MODEL WITH EXISTING WORKS

Differentiating Criterion	Our Attack Model	Other Existing Works
Visibility of the user's phone screen display	Do not need any information of screen display	[7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17] assume having access to the screen display of user's device
Access to various sensors of user's device	Do not access any sensor data from user's device	[18], [19], [21], [22], [23], [24], [25], [26], [27], [28] assume having access to the various sensors in the user's device such as accelerometer, microphone, gyroscope, etc.
Assumption about the structure of password	Considers random password	Balzarotti et al. [13] predict the regular text and use language modeling. Our earlier work [6] predict the numeric pin which is a very simplistic scenario considering the complexity of keyboard for password entry. [7], [15], [16], [17] predict regular text and passwords and use language modeling techniques with an assumption about the structure of the text or passwords.

TABLE X
ENTROPY AND INFORMATION GAIN INDUCED FROM OUR ATTACK MODEL

Attack Model		Entropy	Gain	
			Over Random Attack	**Over Our Model with Random Guess within a Key Cluster**
iPhone Camera	**Our Model with Random Guess within a Key Cluster**	0.61	0.49 ($\sim$ 400%)	-
	Our Complete Model	0.98	0.86 ($\sim$700%)	0.37 (61%)
Sony Camcorder	**Our Model with Random Guess within a Key Cluster**	0.61	0.49 ($\sim$400%)	-
	Our Complete Model	0.99	0.87 ($\sim$700%)	0.38 (62%)
Random Attack		0.12	-	-

results into an average of $< 3^8$ trials of candidate passwords before finding the correct one. It is evident that our method reduces the search space by over 99% compared to an exhaustive search. Also note that if an attacker only utilizes the information from the videos of password typing, he/she can correctly identify the key clusters with an average accuracy of 97% (see Table V). Since each key cluster consists of an average of 6 keys, an attacker would need to try an average of 6 candidates keys for each character in the password. Hence the attacker would need to try an average of 6^8 passwords before finding the correct one that results in over 92% search space reduction.

H. Discussion – Assumptions, Limitations, and Mitigations

There are certain users who enter their password differently from the scenario studied in the paper. While our results may not apply to these users, the paper exposes a major security threat faced by many users. Studying the performance of the attack with appropriate modifications to the algorithm for scenarios such as users entering the password with both hands and multiple fingers will be part of future work.

The success of the method depends on the following factors:

1) The attack model assumes that the user is using the default keypad for the password entry process. This makes the design and location of the keypad on the users device to be easily determined given the make and model of the phone.
2) An attacker could determine the password entry action and get the video of the process. Password entry process involves a very distinct sequence of actions as a targets first action would be entering his/her password after picking up the phone if the device is secured by one.

A potential defense against our attack model is randomization of user-interface which includes randomization of key locations, the location of the whole keypad, etc. However, randomization-based solutions need to be studied more for usability analysis. Since randomization poses usability challenges and hence the users might not adopt these kinds of solutions.

Another possible defense against the attacks such as ours is biometric-based authentication solutions such as fingerprint, face, and behavioral biometrics. However, biometric solutions are also susceptible to another family of attacks. Behavior-based continuous authentication mechanisms [37]–[39] that verify the users identity throughout a devices usage period seem to be the most robust solutions against the attacks such as ours.

V. CONCLUSION

We have shown the feasibility of predicting passwords through the analysis of hand movements in small video clips. We show that an adversary need not have access to the mobile phone or have the screen of the phone visible to execute an attack. Our attacks can easily be launched in public places without the knowledge of the victim. Thus our results expose

serious security and vulnerability of smartphone usage in public places in scenarios such as hiding the screen, while using the phone, that otherwise gives the impression of being safe to the users. The users have a strong conviction that covering the screen while entering a password at a public place adequately secures the password against eavesdropping attacks. Our findings provide an evidence against the notion. We are currently developing methods such as randomization of shape, size, and location of keys on the keypad, biometric based continuous authentication system, etc. to address the issues exposed by this kind of side channel attack. However, these methods still need to be weighed for other security and privacy risks.

APPENDIX

A. Effect of Screen Size of the Victim's Phone

To study the effect of change in screen size of the phone, we recruited 10 volunteer participants and recorded a total of 20 password entry videos. Each volunteer selected a password and typed it two times while we recorded the video of the users typing. The recorded session was preceded with a practice session where the participant practiced typing the selected password 10 times. In the recorded session, one for password entry was made on Samsung Galaxy S4 and other on Samsung Galaxy S9+ phone. A user entered the same password on both the devices whereas every user choose a different password from our *Password List II*. The attack performance was evaluated in various settings of password entry mobile phone and the division of keypad to key clusters. Table XI shows the attack performance in four different settings for comparative analysis; (1) Samsung S4 with 6 key clusters, (2) Samsung S4 with 12 key clusters, (3) Samsung S9+ with 6 key clusters, and (4) Samsung S9+ with 12 key clusters.

TABLE XI
AVERAGE PERCENTAGE OF CHARACTERS PER PASSWORD INFERRED CORRECTLY FOR DIFFERENT SCREEN SIZES OF THE TARGET PHONES AND ATTACK CONFIGURATION

Target Phone and Attack Configuration	Accuracy (expressed as %)		
	1st Guess	5 Guesses	10 Guesses
Samsung S4 with 6 Key Clusters	42.3	57.8	68.0
Samsung S4 with 12 Key Clusters	38.6	51.5	60.0
Samsung S9+ with 6 Key Clusters	46.0	60.4	74.8
Samsung S9+ with 12 Key Clusters	54.8	67.2	79.8

Our results show that the average accuracy of character prediction increases with an increase in screen size of the target phone (Samsung Galaxy S9+ in our experiments) (See Table XI). We observed the best performance of character inference for the password entry videos on Samsung S4 phone when we divided the keypad to 6 key clusters (i.e., 2 rows and 3 columns). On the other hand, analysis of the videos of password entry on Samsung S9+ phone yielded in best accuracy when we divided the keypad to 12 key clusters (i.e., 4 rows and 3 columns). Table XII shows the gain in the attack performance in terms of an average increase in character prediction accuracy with the best performing attack configurations from Samsung S4 phone (screen size $5.0''$) to Samsung S9+ phone (screen size $6.2''$). In the light of our analysis with Samsung S4 and Samsung S9+ as target phones, we believe that the attack performance would improve with todays bigger screen phones that have bigger keypad size.

TABLE XII
PERCENTAGE INCREASE IN THE ATTACK PERFORMANCE FOR THE CHARACTER INFERENCE WITH AN INCREASE IN THE SCREEN SIZE OF THE VICTIM'S PHONE

Change in Screen Size	Gain in Attack Accuracy (expressed as %)		
	1st Guess	5 Guesses	10 Guesses
$6.2''$ **from** $5.0''$	29.6	16.3	17.3

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their detailed comments and for pointing them to Thales and Intercept Theorems.

REFERENCES

[1] *Usenix Enigma 2016—NSA TAO Chief on Disrupting Nation State Hackers*. Accessed: Dec. 2018. [Online]. Available: https://www.youtube.com/watch?v=bDJb8WOJYdA

[2] *Why You Should Never use the Same Password on Multiple Sites*. Accessed: Dec. 2018. [Online]. Available: https://geekdad.com/2018/12/why-you-should-never-use-the-same-password-on-multiple-sites

[3] *Common Types of Cybersecurity Attacks—A Look at the Various Types of Hacking Techniques*. Accessed: Dec. 2018. [Online]. Available: https://www.rapid7.com/fundamentals/types-of-attacks

[4] *Credential Theft: The Business Impact of Stolen Credentials—What are Cybercriminals Doing With Your Stolen Passwords?* Accessed: Dec. 2018. [Online]. Available: https://www.blueliv.com/blog-news/credential-theft/credential-theft-the-business-impact-of-stolen-credentials

[5] *The Credential Theft Ecosystem*. Accessed: Dec. 2018. [Online]. Available: https://www.blueliv.com/the-credential-theft-ecosystem

[6] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha, "Beware, your hands reveal your secrets!" in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2014, pp. 904–917. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660360

[7] Y. Xu, J. Heinly, A. M. White, F. Monrose, and J.-M. Frahm, "Seeing double: Reconstructing obscured typed input from repeated compromising reflections," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 1063–1074.

[8] G. Ye *et al.*, "A video-based attack for android pattern lock," *ACM Trans. Privacy Secur.*, vol. 21, no. 4, p. 19, 2018.

[9] T. Chen, "Smartphone passcode prediction," *IET Inf. Secur.*, vol. 12, pp. 431–437, Sep. 2018. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-ifs.2017.0606

[10] K. S. Balagani *et al.*, "SILK-TV: Secret information leakage from keystroke timing videos," in *Computer Security*, J. Lopez, J. Zhou, and M. Soriano, Eds. Cham, Switzerland: Springer, 2018, pp. 263–280.

[11] R. Raguram, A. M. White, D. Goswami, F. Monrose, and J.-M. Frahm, "iSpy: Automatic reconstruction of typed input from compromising reflections," in *Proc. 18th ACM Conf. Comput. Commun. Secur. (CCS)*, 2011, pp. 527–536.

[12] M. Backes, M. Dürmuth, and D. Unruh, "Compromising reflections-or-how to read LCD monitors around the corner," in *Proc. IEEE Symp. Secur. Privacy*, May 2008, pp. 158–169.

[13] D. Balzarotti, M. Cova, and G. Vigna, "ClearShot: Eavesdropping on keyboard input from video," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2008, pp. 170–183.

[14] F. Maggi, A. Volpatto, S. Gasparini, G. Boracchi, and S. Zanero, "A fast eavesdropping attack against touchscreens," in *Proc. Inf. Assurance Secur. (IAS)*, Dec. 2011, pp. 320–325.

[15] Q. Yue, Z. Ling, W. Yu, B. Liu, and X. Fu, "Blind recognition of text input on mobile devices via natural language processing," in *Proc. Workshop Privacy-Aware Mobile Comput.*, 2015, pp. 19–24.
[16] Q. Yue, Z. Ling, X. Fu, B. Liu, W. Yu, and W. Zhao, "My Google glass sees your passwords!" in *Proc. Black Hat USA*, 2014. [Online]. Available: https://www.blackhat.com/docs/us-14/materials/us-14-Fu-My-Google-Glass-Sees-Your-Passwords-WP.pdf
[17] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao, "Blind recognition of touched keys on mobile devices," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 1403–1414.
[18] D. Asonov and R. Agrawal, "Keyboard acoustic emanations," in *Proc. IEEE Symp. Secur. Privacy*, May 2004, pp. 3–11.
[19] L. Zhuang, F. Zhou, and J. D. Tygar, "Keyboard acoustic emanations revisited," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 1, pp. 36:1–36:6, Oct. 2009. [Online]. Available: http://doi.acm.org/10.1145/1609956.1609959
[20] D. X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on SSH," in *Proc. 10th Conf. USENIX Secur. Symp. (SSYM)*, vol. 10, 2011, p. 25.
[21] L. Cai and H. Chen, "TouchLogger: Inferring keystrokes on touch screen from smartphone motion," in *Proc. 6th USENIX Conf. Hot Topics Secur. (HotSec)*, 2011, p. 9.
[22] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "ACCessory: Password inference using accelerometers on smartphones," in *Proc. 12th Workshop Mobile Comput. Syst. Appl. (HotMobile)*, 2012, pp. 9:1–9:6.
[23] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, "TapPrints: your finger taps have fingerprints," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2012, pp. 323–336.
[24] Z. Xu, K. Bai, and S. Zhu, "TapLogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors," in *Proc. 5th ACM Conf. Secur. Privacy Wireless Mobile Netw. (WISEC)*, 2012, pp. 113–124.
[25] L. Simon and R. Anderson, "Pin skimmer: Inferring pins through the camera and microphone," in *Proc. 3rd ACM Workshop Secur. Privacy Smartphones Mobile Devices (SPSM)*, 2013, pp. 67–78.
[26] C. Wang, X. Guo, Y. Chen, Y. Wang, and B. Liu, "Personal PIN leakage from wearable devices," *IEEE Trans. Mobile Comput.*, vol. 17, no. 3, pp. 646–660, Mar. 2018.
[27] B. Tang, Z. Wang, R. Wang, L. Zhao, and L. Wang, "Niffler: A context-aware and user-independent side-channel attack system for password inference," *Wireless Commun. Mobile Comput.*, vol. 2018, May 2018, Art. no. 4627108.
[28] C. X. Lu *et al.*, "Snoopy: Sniffing your smartwatch passwords via deep sequence learning," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 1, no. 4, p. 152, 2018.
[29] *TLD*. Accessed: Aug. 2018. [Online]. Available: http://kahlan.eps.surrey.ac.uk/featurespace/tld
[30] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, Jul. 2012.
[31] *Never Ending Security: Uniqpass V15, Large Password List*. Accessed: Nov. 2017. [Online]. Available: https://neverendingsecurity.wordpress.com/2015/04/19/uniqpass-v15-large-password-list/
[32] *Windows Movie Maker*. Accessed: Aug. 2017. [Online]. Available: http://windows.microsoft.com/en-us/windows-live/movie-maker
[33] A. Ostermann and G. Wanner, "Thales pythagoras," in *Geometry by Its History* (Undergraduate Texts in Mathematics). Berlin, Germany: Springer, 2012, pp. 3–26.
[34] B. Bradtmiller, B. Hodge, S. Kristensen, and M. Mucher, "Anthropometric survey of federal aviation administration technical operations personnel—2006-2008: Prime contract DTFAWA-07-C-00059," Report prepared for Federal Aviation Admin., Washington, DC, USA, 2008.
[35] *Human Hand and Foot*. Accessed: Dec. 2018. [Online]. Available: http://www.goldennumber.net/human-hand-foot
[36] R. Szeliski, *Computer Vision: Algorithms and Applications* (Texts in Computer Science). London, U.K.: Springer, 2010. [Online]. Available:https://books.google.com/books?id=8_2RNQEACAAJ
[37] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Cell phone-based biometric identification," in *Proc. 4th IEEE Int. Conf. Biometrics, Theory Appl. Syst. (BTAS)*, Sep. 2010, pp. 1–7.
[38] R. Kumar, P. P. Kundu, D. Shukla, and V. V. Phoha, "Continuous user authentication via unlabeled phone movement patterns," in *Proc. IEEE Int. Joint Conf. Biometrics (IJCB)*, Oct. 2017, pp. 177–184.
[39] V. Patel, R. Chellappa, D. Chandra, and B. Barbello, "Continuous user authentication on mobile devices: Recent progress and remaining challenges," *IEEE Signal Process. Mag.*, vol. 33, no. 4, pp. 49–61, Jul. 2016.

Diksha Shukla received the M.C.A. degree in computer applications from Jawaharlal Nehru University, New Delhi, India, in 2011 and the M.S. degree in mathematics from Louisiana Tech University, USA, in 2014. She is currently pursuing the Ph.D. degree in computer & information science and engineering with Syracuse University, Syracuse, NY, USA. Her research interests include machine learning, computer vision, and cybersecurity. Her research spans applications of these areas to wearable devices, authentication, biometrics, and side channel attacks.

Vir V. Phoha (M'96–SM'02) received the Ph.D. degree in computer science from Texas Tech University, Lubbock, in 1992. He is currently a Professor of electrical engineering and computer science with the College of Engineering and Computer Science, Syracuse University. His research interests include attack-averse authentication, optimized attack formulation, machine learning, anomaly detection, spatial-temporal pattern detection and event recognition, and knowledge discovery and analysis. He is a fellow of the AAAS and SDPS. He is an ACM Distinguished Scientist.