A Video-based Attack for Android Pattern Lock

GUIXIN YE, ZHANYONG TANG*, DINGYI FANG, XIAOJIANG CHEN, Northwest University, China
WILLY WOLFF, Lancaster University, U. K.
ADAM J. AVIV, Naval Academy, U.S.A.
ZHENG WANG*, Lancaster University, U. K.; Xi'an University of Posts & Telecommunications, China

Pattern lock is widely used for identification and authentication on Android devices. This article presents a novel video-based side channel attack that can reconstruct Android locking patterns from video footage filmed using a smartphone. As a departure from previous attacks on pattern lock, this new attack does not require the camera to capture any content displayed on the screen. Instead, it employs a computer vision algorithm to track the fingertip movement trajectory to infer the pattern. Using the geometry information extracted from the tracked fingertip motions, the method can accurately infer a small number of (often one) candidate patterns to be tested by an attacker. We conduct extensive experiments to evaluate our approach using 120 unique patterns collected from 215 independent users. Experimental results show that the proposed attack can reconstruct over 95% of the patterns in five attempts. We discovered that, in contrast to most people's belief, complex patterns do not offer stronger protection under our attacking scenarios. This is demonstrated by the fact that we are able to break all but one complex patterns (with a 97.5% success rate) as opposed to 60% of the simple patterns in the first attempt.

We demonstrate that this video-side channel is a serious concern for not only graphical locking patterns but also PIN-based passwords, as algorithms and analysis developed from the attack can be easily adapted to target PIN-based passwords. As a countermeasure, we propose to change the way the Android locking pattern is constructed and used. We show that our proposal can successfully defeat this video-based attack. We hope the results of this article can encourage the community to revisit the design and practical use of Android pattern lock.

CCS Concepts: • **Security and privacy → Security services**; **Authentication**; **Graphical / visual passwords**; *Authorization*;

Additional Key Words and Phrases: Pattern lock, Fingertip movement, Video-based attack, Sensitive information, Object tracking, Authentication mechanism

ACM Reference Format:
Guixin Ye, Zhanyong Tang*, Dingyi Fang, Xiaojiang Chen, Willy Wolff, Adam J. Aviv, and Zheng Wang*. 2018. A Video-based Attack for Android Pattern Lock. *ACM Trans. Priv. Sec.* 9, 4, Article 39 (March 2018), 31 pages. https://doi.org/10.1145/3230740

Extension of Conference Paper: a preliminary version of this article entitled "Cracking Android Pattern Lock in Five Attempts" by G. Ye *et al.* appeared in The Network and Distributed System Security Symposium (NDSS), 2017 [Ye et al. 2017].

*Corresponding authors: Zhanyong Tang and Zheng Wang.

Authors' addresses: Guixin Ye, Zhanyong Tang*, Dingyi Fang, Xiaojiang Chen, Northwest University, China, gxye@stumail.nwu.edu.cn, {zytang,dyf,xjchen}@nwu.edu.cn; Willy Wolff, Lancaster University, U. K., w.wolff@lancaster.ac.uk; Adam J. Aviv, Naval Academy, U.S.A., aviv@usna.edu; Zheng Wang*, Lancaster University, U. K.; Xi'an University of Posts & Telecommunications, China, z.wang@lancaster.ac.uk.

2471-2566/2018/3-ART39 $15.00
https://doi.org/10.1145/3230740

1 INTRODUCTION

Graphical-based passwords, like the Android pattern lock, are widely used as a protection mechanism to prevent sensitive information leakage from mobile devices. It is preferred by many users over PIN- or text-based passwords, as psychology studies show that the human brain remembers and recalls visual information better than numbers and letters [De Angeli et al. 2005; Standing et al. 1970; Weiss and De Luca 2008]. According to a recent study, 40% of the Android users use patterns to protect their devices instead of a PIN [Bruggen 2014]. Pattern lock is also used by many critical applications for authentication. For example, `Alipay`, the largest third-party online-payment platform with over 450 million users in China, uses pattern lock as part of the login authentication. Considering its widespread usage, a security breach of the pattern lock could lead to serious consequences.

Security experts have demonstrated several ways to launch an attack for pattern lock in the past. These include thermal [Abdelrahman et al. 2017], smudge [Aviv et al. 2010] and WiFi attacks [Zhang et al. 2016]. However, these previous attacks all rely on assumptions that are often too strong to realize in practice, and as a result, the attack is unlikely to be successfully launched. For examples, thermal and smudge attacks can be easily disrupted by other on-screen operations after drawing the PIN or pattern, and they are less effective on lock patterns that contain multiple overlaps; wireless based attacks, on the other hand, require the environment to remain static, because any moving object nearby can interfere the wireless signal.

Recently, video-based analysis is shown to be effective in reconstructing PIN- or text-based passwords. Some of the early successes in this area rely on video footage filmed using a camera directly facing the screen or the keyboard [Balzarotti et al. 2008; Kuhn 2002]. Latest work shows that this limitation can be lifted by exploiting spatial-temporal dynamics of the hands during typing [Shukla et al. 2014]. Despite the success of video-based attacks on PIN- and text-based passwords, no work so far has exploited video-based side-channels to crack pattern lock. To do so, the attacker must deal with the fundamental difference between graphical patterns and PIN-or text-based passwords[1]. He must be able to map the user's fingertip movements to a graphical structures and to translate the graphical structure into locking patterns. To overcome these challenges requires new methods and analysis to be constructed in the new application context of pattern lock.

In this article, we present a novel approach to crack Android pattern lock using video footage that captures the user's fingertip motions when drawing the pattern. Unlike prior work on pattern lock attacks such as the thermal [Abdelrahman et al. 2017] and smudge attacks [Aviv et al. 2010], our approach does not require the video footage or images to be captured by a camera directly faced the screen. Furthermore, the video can be filmed at a distance of two meters away from the user in public places. Such a distance is less likely to raise suspicion compared to shoulder surfing [Rogers 2007] that requires a closer observation distance to have a clear sight of the content displayed on the screen. A recent study has shown that shoulder surfing is not perceived as a serious threat by many mobile users because to launch the attack an adversary has to stay close to the target [Eiband et al. 2017]; and as a result, many users do not take extra protection when entering sensitive information in public spaces. As a departure from prior approaches, our attack is more likely to succeed because the adversary can stay further away from the target and the attack does not rely on any information on the screen.

Approach. To infer the user's locking pattern, the attack uses a computer vision algorithm to track the fingertip motions from the video. Using the geometry information extracted from the fingertip motions, it then maps the tracked fingertip locations to a small number of candidate

[1] A pattern essentially is a graphical shape with continuous points. This is different from a PIN-or text-based password consisting of discrete characters.

patterns to be tested on the target device. We show that an adversary can employ a set of empirical heuristics and algorithms developed in other domains to overcome a range of practical issues to successfully launch the attack. This set of issues include how to translate the video footage from the camera's perspective to the user's, how to identify the start and the end of pattern drawing, how to rank candidate patterns, etc.

Results. We thoroughly evaluate our approach on a large number of locking patterns, including 120 unique patterns collected from independent users. The experiment results show that our approach is highly accurate in inferring candidate patterns and as a result, an attacker can unlock the target device with a success rate of over 95% (up to 97.5%) in five attempts. We demonstrate that, in contrast to many people's belief, complex locking pattern is less secure than a simpler one under our attacking methodology. We also show that the algorithms and analysis developed in this attack can be used to target PIN-based passwords with a high success rate. As a countermeasure, we propose to change the way how a locking pattern is formed and used. We show that by adding some randomness to pattern drawing, our countermeasure can significantly increase the difficulties for launching the video-based attacks.

Contributions. The key contribution of this article are summarized as follows.

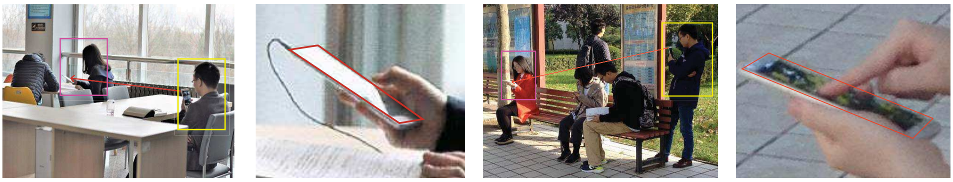
- This is the first work to exploit the video side-channel to automatically reconstruct Android pattern lock using computer vision algorithms. (Section 4).
- We identify **a new vulnerability**. In our attacking scenario, filming can be carried out at a distance of two meters away from the user and the camera does not need to directly face the target device (Section 3). Such a camera setting makes our attack less likely to raise suspicion, but is more likely to succeed when compared to direct observation, e.g. shoulder surfing.
- We discover **a counter-intuitive finding**. The experimental results suggest that complex patterns are more vulnerable under video-based attacks (Section 7.1). This finding debunks many people's conception that more complex patterns lead to stronger protection. Therefore, this work sheds new insights on the practical use of pattern lock.
- We present **a new countermeasure**. We show that by making some small modifications to how the pattern lock is formed, a simple yet effective countermeasure can be proposed to significantly reduce the risk of the presented attack.

2 ANDROID PATTERN LOCK

Pattern lock is a popular authentication mechanism for Android touch-screen devices such as mobile phones, smart watches and tablets. Many users prefer to use pattern lock because they are easier to be recalled over alphanumeric characters [Standing et al. 1970; Weiss and De Luca 2008]. To unlock a device protected with pattern lock, the user is asked to draw a predefined sequence of connected dots on a pattern grid[2] (see Figure 2e).

There are several rules for creating an Android pattern: (1) a pattern must consist of at least four dots; (2) each dot can only be visited once; and (3) a previously unvisited dot will become visited if it is part of a horizontal, vertical or diagonal line segment of the pattern. Considering these constraints, the total number of patterns on a 3 × 3 grid is 389,112 [Uellenbeck et al. 2013]. Given the large number of possible patterns, performing brute-force attacks is ineffective, especially for the patterns with complex structures [Kelley et al. 2012; Mazurek et al. 2013], because the device will be automatically locked after five failed tries. Previous works also show that a brute-force attack is likely to fail on patterns with complex structures [Kelley et al. 2012; Mazurek et al. 2013].

[2] In this article we use the Android default pattern grid with 3 × 3 dots, unless otherwise stated.



(a) The video was recorded from a distance of 2.5 meters.

(b) The device screen seen from the video filmed in (a).

(c) An outdoor scenario.

(d) The device screen seen from the video filmed in (d).

Fig. 1. Examples of scenarios in which a mobile phone camera is used to film the unlocking process. In these scenarios, the camera does not need to have a clear sight of the screen.

3 THREAT MODEL

In our threat model, we assume an adversary wants to access some sensitive information from or to install malware on a target device that is protected by pattern lock. This type of attack is mostly likely to be performed by an attacker who can physically access to the target device for a short period of time (e.g. via attending a meeting or a party where the user is present). To quickly gain access to the device, the attacker would like to obtain the user's locking pattern in advance.

The attack starts from filming how the user unlocks the device. Video recording can be done on-site or ahead of time. Because filming can be carried out from a distance of as far as 2 meters using a mobile phone camera (or about 9 meters using a DSLR camera) and the camera does not need to directly face the target device, this activity often will not be noticed by the user. Moreover, given that many users use the same pattern across devices and applications, the pattern obtained from one device may also be used to break the user's other devices.

Examples of Attacking Scenarios. Figure 1 illustrates two day-to-day scenarios where filming can be performed without raising suspicion to many users. The filming camera had a left- or right-front view angle from the target device indoor or outdoor and did not directly face the screen of the target device. Due to the filming distance (2-3 meters), the recoded video typically does not have a clear vision of the content displayed on the screen as shown in Figure 1.

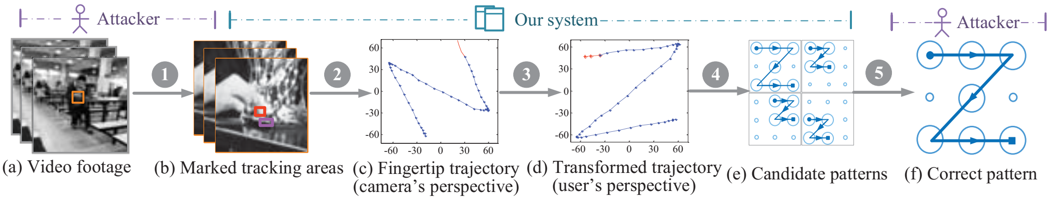
Assumptions. Our attack requires the video footage to have a vision of the user's fingertip that was involved in pattern drawing as well as part of the device. We believe this is a reasonable assumption because in practice many users often do not fully cover their fingers and the entire device when drawing a pattern. This is particularly true when holding a large-screen device by hands. To launch the attack, the attacker needs to know the layout of the grid, e.g. whether it is a 3×3 or a 6×6 grid. This can be simply decided by seeing target device.

4 OVERVIEW OF OUR ATTACKING SYSTEM

In this section, we give an overview of our attacking system which analyzes the user's fingertip movement to infer the locking pattern[3]. The system takes in a video segment that records the entire unlocking process. It produces a small number of candidate patterns to be tested on the target device. Figure 2 depicts the five key steps of our attack:

① **Video Filming and Preprocessing:** The attack begins from filming how the pattern is drawn. This can be done at a distance of $2-3$ meters away from the user using a mobile phone rear camera. After recording, our system can automatically cut out a video segment that contains the entire

[3] A simple variant method of the system can also break PIN-based passwords. This is demonstrated in Section 7.9.



(a) Video footage (b) Marked tracking areas (c) Fingertip trajectory (camera's perspective) (d) Transformed trajectory (user's perspective) (e) Candidate patterns (f) Correct pattern

Fig. 2. Overview of the attack. Our system takes in a video segment that records the unlocking process (a). The adversary first marks two areas of interest on the first video frame (b): one contains the fingertip involved in pattern drawing, and the other contains part of the device. Our system then tries to track the fingertip's location w.r.t. to the device. The tracking algorithm produces a fingertip movement trajectory from the camera's perspective (c) which is then transformed to the user's perspective (d). Finally, the resulted trajectory in (d) is mapped to several candidate patterns (e) to be tested on the target device (f).

unlocking process (Section 5.1). The attacker then need to mark two areas of interest from a video frame: one area consists of the fingertip used to draw the pattern, and the other consists of part of the device (see Figure 2 (b)).

② **Track Fingertip Locations:** Once the areas of interest are highlighted, a computer vision algorithm will be applied to locate the fingertip from each video frame (Section 5.2.2). The algorithm aggregates the successfully tracked fingertip locations to produce a fingertip movement trajectory. This is illustrated in Figure 2 (c). Keep in mind that at this stage the tracked trajectory is presented from the camera's perspective.

③ **Filming Angle Transformation:** This step transforms the tracked fingertip locations from the camera's perspective to the user's. We use an edge detection algorithm to automatically calculate the filming angle which is then used to perform the transformation (Section 5.3). For example, Figure 2 (c) will be transformed to Figure 2 (d) to obtain a fingertip movement trajectory from the user's perspective.

④ **Identify and Rank Candidate Patterns:** In this step, our software automatically maps the tracked fingertip movement trajectory to a number of candidate patterns (Section 5.4). We rank the candidate patterns based on a heuristic described in Section 5.4.2. For instance, the fingertip movement trajectory in Figure 2 (d) could be mapped to a number of candidate patterns shown in Figure 10. We show that our approach can reject most patterns to leave no more than five candidate patterns to be tried out on the target device.

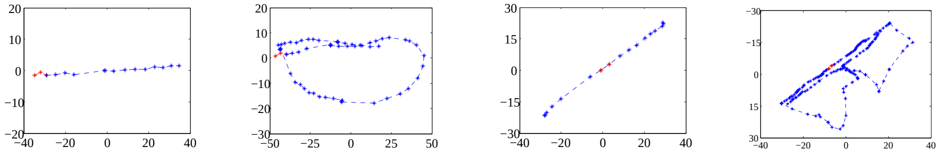
⑤ **Light Weight Trials:** In this final step, the attacker tries the candidate patterns one by one on the target device.

5 IMPLEMENTATION DETAILS

5.1 Video preprocessing

This step aims to identify the unlocking process from the video footage. While all our participants (see Section 6.1) consider this as a straightforward manual task, we developed a *simple yet effective* heuristic to automatically detect the video segment in some typical scenarios. Our heuristic is based on the following observations: (1) before or after unlocking, users often pause for a few seconds; (2) two consecutive on-screen operations (e.g. swiping, zooming etc.) typically expose some spatial-temporal motion characteristics.

In our initial test, we find that there exists at least 1.5 seconds pause before or after pattern drawing due to delay of the user or the device. We also found that identical on-screen activities often follow closely. These consecutive on-screen operations have some spatial-temporal motion characteristics that are different from pattern drawing. Figure 3 shows the spatial-temporal motion



(a) a horizontal-swiping gesture

(b) two consecutively horizontal-swiping gestures

(c) a zooming gesture

(d) two consecutive zooming gestures

Fig. 3. Spatial-temporal characteristics for performing an on-screen gesture once (a, c) and twice (b, d).

```
Algorithm 1: Unlocking process identification heuristic
Input:
    $IV$: Video footage
    $frameCount$: Pause threshold before or after unlocking
Output:
    <start,end>: Start and end of the unlocking video segment
 1: $frames[] \leftarrow getVideoFrames(IV)$
 2: $LEN \leftarrow getFramesLen(frames[])$
 3: for $i = 1 : LEN - frameCount$ do
 4:     $sL \leftarrow hasFingertipChanged(frames[i : i + frameCount])$
 5:     if $!sL$ then
 6:         $sNo = i + frameCount$
 7:         for $j = sNo : LEN$ do
 8:             if $checkLoop(frames[j : LEN])$ then
 9:                 $eNo = i$
10:                 break;
11:             else if $!hasFingertipChanged(frames[j : j + frameCount])$ then
12:                 $eNo = i$
13:                 break;
14:             end if
15:         end for
16:         break;
17:     end if
18: end for
19: $< start, end >\leftarrow getTargetVideo(frames[], sNo, eNo)$
```

structure for two gestures, swiping and zooming, when they are performed once (a, c) and twice (b, d). This diagram indicates that the spatial-temporal motion of two identical on-sreen activities contains one or more looping structures for which pattern drawing does not have.

Our heuristic for identifying the pattern drawing process is described in Algorithm 1. The input to the algorithm is a video capturing the unlocking process, and the output of the algorithm is a time-stamp tuple, <start, end>, which marks the start and the end of a video segment. To locate the video segment, we first filter out on-screen activities where the fingertip location does not change within a timeframe of 1.5 seconds (lines 4 and 11). This allows us to exclude some basic on-screen activities such as clicking. Figure 4 shows that all our participants paused at least 1.5 seconds before or after pattern drawing due to delay of the user or the device. We use the number of video frames,

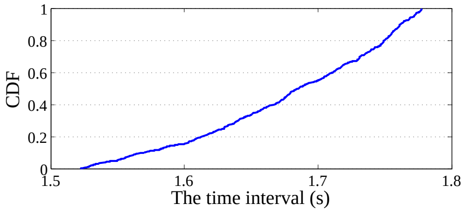


Fig. 4. The cumulative distribution function (CDF) of the time interval between pattern drawing and other on-screen activities.

frameCount, as a proxy to estimate the time interval between two on-screen operations. Here, a time interval of 1.5s translates to 45 frames or 90 frames when the video was shot at 30 or 60 frames per second (FPS) respectively. We also use the spatial-temporal characteristics described above to exclude two consecutive swiping or zooming gestures (line 8). Finally, we exploit the observation that users typically paused at least 1.5s before or after unlocking to locate the start and end points of pattern drawing (line 19).

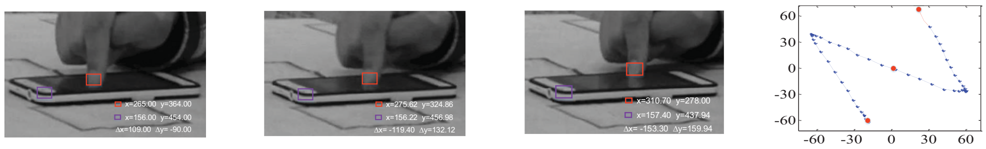
Limitations. Our heuristic is not perfect. It is likely to fail if the user was typing using a Swype-like method (i.e. entering words by sliding a finger from the first letter of a word to its last letter) during video recording. In this case, our method will identify multiple video segments of which one may contain the pattern unlock process. If multiple segments are detected, the algorithm will ask the user to confirm which video segment to use. In this scenario, the first identified segment is likely to be the correct one. In practice, an experienced attacker would wait patiently to avoid this complicated situation by finding the right time for filming (e.g. for a screen lock, the time is just after the device is retrieved). The attacker could also watch the video to manually cut it to ensure to obtain the correct video segment.

5.2 Track fingertip locations

After cutting out the video segment of pattern drawing, we need to track the finger motions from the video segment. We achieve this by employing a video tracking algorithm called *Tracking-Learning-Detection (TLD)* [Kalal et al. 2011]. This algorithm automatically detects objects defined by a boundary box. In our case, the objects to be tracked are the user's fingertip and an area of the device. These are supplied to the algorithm by simply highlighting two areas on the first frame of the video segment (see Figure 2 b). The algorithm tries to localize the fingertip from each video frame and aggregates the successfully tracked locations to produce a fingertip movement trajectory as an output (see Figure 2 c).

5.2.1 Generate The Fingertip Movement Trajectory. The TLD algorithm [Kalal et al. 2011] automatically detects objects based on the examples seen from the first frame. For each tracked object, the algorithm generates a confidence between 0 and 1. A tracking is considered to be successful if the confidence is greater than a threshold. We set this threshold to 0.5 which is found to give good performance in our initial design experiments using 20 patterns[4]. TLD has three modules: (1) a tracker that follows objects across consecutive frames under the assumption that the frame-to-frame motion is limited and objects are visible; (2) a detector to fully scan each individual frame to localize all appearances of the objects; and (3) a learner that estimates errors of the detector and updates the detector to avoid these errors in future frames.

[4] To provide a fair evaluation, the patterns used in our initial test runs in the design phase are different from the ones used later in evaluation.



(a) The first video frame

(b) A middle video frame

(c) The last video frame

(d) Fingertip trajectory

Fig. 5. Tracking the fingertip movement trajectory. For each video frame, the system tracks two areas: one surrounds the fingertip and the other covers the edge of the device. The fingertip position is determined by computing the relative coordinates of the central points of the two areas. The red points highlighted in the final results (d) are the touching points tracked from the three video frames.

In some specific cases, the algorithm may fail to detect the objects in many video frames due to poor selections of interesting areas. If this happens, our system will ask the user to re-select the areas to track. We have also extended TLD to report when a fingertip position is seen on the footage. This temporal information is recorded as the number of video frames seen with respect to the first frame of the video segment. This is used to separate two possibly overlapping line segments described in Section 5.4.

5.2.2 Camera Shake Calibration. By default, the TLD algorithm reports the position of a tracked object with respect to the top-left pixel of the video frame. However, videos recorded by a hand-held device are not always perfectly steady due to camera shake. As a result, the top-left pixel of a video frame may appear in a different location in later frames. This can drastically affect the precision of fingertip localization, leading to misidentification of patterns.

Our approach to cancel camera shake is to record the fingertip location with respect to a fixed point of the target device. To do so, we track two areas from each video frame. One area is an edge of the device and the other is the fingertip. Both areas are highlighted on the first frame by the user. The location of a successfully tracked fingertip is reported as the relative coordinates of the two center points of the marked areas. This approach can also be used to calibrate the minor motions of the target device during pattern drawing.

Example: To illustrate how our camera-shake calibration method works, consider Figure 5 where two areas are firstly marked by two bounding boxes in subfigure (a). Both areas will then be automatically detected by the TLD algorithm in following video frames as shown in subfigures (b) and (c). The coordinates of the two center points of each box are the values of x and y, and their relative positions are represented by $\triangle X$ and $\triangle Y$. For each frame where both areas are successfully tracked, we compute the relative coordinates, $(\triangle X, \triangle Y)$, which are reported as the location of the tracked fingertip.

5.3 Filming angle transformation

In practice, the filming camera will not directly face the target device to avoid raising suspicion by the target user. As a result, the fingertip movement trajectory generated by the tracking algorithm will look different than the actual pattern. For example, for the pattern presented in Figure 2 (a), if the video is filmed from the attacker's front-left to the target device (i.e. with a filming angle of approximate 45 degrees), we get the trajectory shown in Figure 2 (c). Using this trajectory without any postprocessing will lead to misidentification of candidate patterns. Therefore, we must transform the resulting trajectory to the user's view point. To do so, we need to estimate the angle between the filming camera and the target device. Our approach is described as follows.

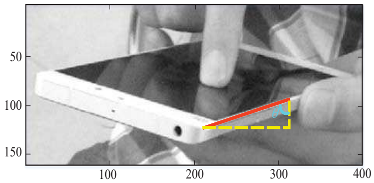


Fig. 6. Filming angle calculation. The filming angle, θ, is the angle between the edge line of the device and a vertical line.

We use an edge detection algorithm called Line Segment Detector (LSD) [Grompone et al. 2010] to detect the longer edge of the device. The filming angle is the angle between the detected edge line and a vertical line. This is illustrated in Figure 6. In Section 7.5, we show that a minor estimation error of the filming angle has little impact on the attacking success rate. By default, we assume that the pattern grid is presented in the portrait mode[5]. If this is not the case, i.e. the pattern grid is shown in the landscape mode, we need to use the shorter edge of the device to calculate the filming angle. We believe that an attacker interested in a particular target device would have some knowledge of how the pattern grid is presented under different orientation modes and be able to identify the device orientation by watching the video. There are also other methods to be used to identify the filming angle [Torralba and Oliva 2002].

Based on the estimated filming angle, θ, we use the following formula to transform the tracked fingertip movement trajectory from the camera's view point to the user's:

$$S = TS^{'} \qquad , \qquad T = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{1}$$

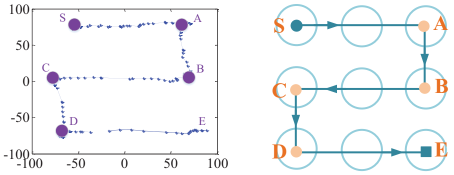
where T is a Transformation Matrix, $S^{'}$ is the coordinate of a point of the tracked trajectory, and S is the resulting coordinate after the transformation. For each video frame, our algorithm individually calculates the filming angle and perform the transformation, because the filming angle may change across video frames.

5.4 Identify and rank candidate patterns

In this step, the fingertip movement trajectory will be mapped to a number of candidate patterns to be tested on the target device. Our goal in this step is to exclude as many patterns as possible and only leave the most-likely patterns to be tried out on the target device. Our approach is to use the geometry information of the fingertip movement trajectory, i.e. the length and direction of line segments and the number of turning points, to reject patterns that do not satisfy certain criteria. In this section, we first describe how to identify overlapping line segments and extract length and direction information before presenting how to use the extracted information to identify and rank candidate patterns.

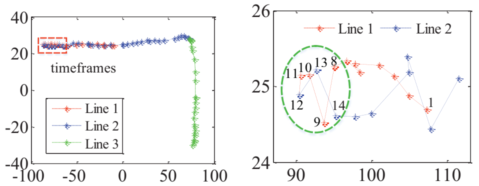
5.4.1 Extracting Structure Information. A pattern can be defined as a collection of line segments where each line segment has two properties: the length of the line, l, and the direction of the line, d. We define a pattern, P, as a collection of line segment prosperities, $P = \{L, D\}$. Here $L = \{l_1, l_2, \cdots, l_n\}$ is a collection of the lengths of all line segments (that are numbered from 1 to n) of the pattern, and $D = \{d_1, d_2, \cdots, d_n\}$ is the collection of directions for all line segments in L. We extract the length and the direction of each line segment from the tracked fingertip movement trajectory and store them into arrays $L[]$ and $D[]$ respectively.

[5] The pattern grid of the Android native pattern lock is always presented in the portrait mode regardless of the orientation of the device.



(a) tracked fingertips (b) pattern example

Fig. 7. The tracked fingertip movement trajectory (a) of a pattern (b).



(a) overlapping lines (b) zoom-in view

Fig. 8. Separating two overlapping line segments.

Table 1. Mappings from line slopes and fingertip-horizontal movements to direction numbers

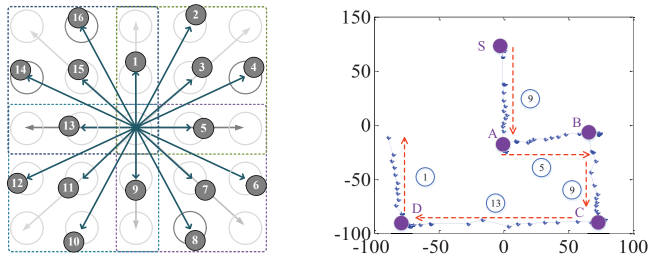
Direction No.	1	2	3	4	5	6	7	8
slope (L → R)	$+\infty$	2	1	$\frac{1}{2}$	0	$-\frac{1}{2}$	-1	-2
Direction No.	9	10	11	12	13	14	15	16
slope (R → L)	$-\infty$	2	1	$\frac{1}{2}$	0	$-\frac{1}{2}$	-1	-2

Identify Line Segments. The first step of geometry information extraction is to identify individual line segments from the trajectory. This can be achieved by finding turning points, the start and the end points of the pattern, because two points define a line segment. For example, turning points, A and B, in Figure 7 define a line segment, AB. We use a linear fitting method [Kutner et al. 2004] to discover turning points. A specific challenge here is how to separate two overlapping line segments. It is to note that up to two lines can be overlapped on a pattern grid. The naive linear fitting algorithm would consider two overlapping segments to be a single line as their points stay close to each other. We overcome this problem by using the temporal information (that is recorded by the tracking algorithm) to separate two overlapping points.

Example: As an example, consider a fingertip movement trajectory shown in Figure 8 (a). The red rectangle on the figure is a timeframe consisting of 20 tracked points. If we zoom in on the timeframe, we get Figure 8 (b) where a point is labelled with a frame number according to when the point was seen, starting from 1 for the earliest point. In this example, there are more than 6 overlapping points in the timeframe, which are marked by a green circle. We use the center point (No.10) of the overlapping points as the turning point to separate the two line segments.

Extract the Line Length. The physical length of a line segment depends on the sizes of the screen and the pattern grid, and the space between two touch dots. To ensure our approach is independent of the device, we normalize the physical length of a line segment to the shortest line found on the tracked trajectory. For the example as shown in Figure 7 (a), the line lengths for segments, SA, AB, BC, CD, and DE, are $2l_s, l_s, 2l_s, l, 2l_s$, respectively. Here segments AB and CD have the shortest length, l_s. The physical length of a line segment is calculated by computing the Euclidean distance between the start and the end points of a segment.

Extract Direction Information. In addition to the line length, we also want to know to which direction the finger moves. This information is useful for inferring which dots are selected to unlock the pattern. Figure 9 (a) shows all possible 16 directions on a 3×3 pattern grid. The directions are numbered from 1 to 16 in clockwise. For each line segment of the tracked trajectory, we calculate its line slope and the horizontal movement of the finger (i.e. left → right or vice versa). This information will then be checked against Table 1 to determine the direction number of the line segment. The



(a) line direction number (b) numbering line segment

Fig. 9. All possible line directions (a) and an example trajectory (b).

horizontal movement of the fingertip is determined by first using the temporal information to find out the start and the end points of the line and then comparing the horizontal coordinates of the two points. The line slope is also computed based on the coordinates of the start and the end points of the line segment. Figure 9 (b) gives the direction number of each tracked line segment of a fingertip movement trajectory.

5.4.2 Map the Tracked Trajectory to Candidate Patterns. In this step, we use the extracted geometry information to map the fingertip movement trajectory to a small number of candidate patterns which will then be ranked using a heuristic.

Identify Candidate Patterns. Our implementation simply enumerates all possible patterns for a given pattern grid to identify candidate patterns, starting from the top-left touch point. We reject patterns that do not meet the requirements that the correct pattern is expected to have. The requirements are the number of line segments (this is checked by counting the number of turning points), and the length and the direction for each line segment. This is an *automatic* process performed by our software system without any user involvement. We consider two line segments having the same length and slope if the difference between them is less than a threshold. Specifically, the relative length threshold, $lengthTh$, is set to 1.12 and the slope threshold, $directionTh$, is set to 0.25. To determine the thresholds, we have evaluated a range of possible settings using 30 patterns in our initial design experiments[6]. We found that our chosen thresholds lead to good performance – our attack only fails on 1 out of the 30 patterns due to blur motions of the video footage.

Example: We use the pattern depicted in Figure 2 as an example to describe our algorithm. Figure 10 gives several possible mappings for the fingertip movement trajectory shown in Figure 2 (d). For this particular trajectory, the collections of lengths and directions are $L = \{l, \sqrt{2}l, l\}$ and $D = \{5, 11, 5\}$ respectively. Any pattern that does not meet L or D should not be considered as a candidate pattern for this trajectory. For this reason, Figure 10 a(1)–a(9) will be rejected. Take Figure 10 a(1) as an example, the line lengths and directions for all four line segments of this pattern are $\{l, \frac{\sqrt{5}}{2}l, l\}$ and $\{5, 12, 5\}$ respectively.

Rank Patterns. Candidates patterns are then ranked using a simple heuristic. The heuristic assumes a pattern starting from a left dot of the grid is more likely to be the correct pattern over a pattern starting from a right dot. This assumption is supported by recent studies which show that people tend to select a left dot as the starting point to construct a pattern [Løge 2015; Uellenbeck et al. 2013]. If two candidate patterns start from the same dot, we consider the pattern with a longer

[6] In most cases, our participants drew pass the centers of all dots when forming a line. We also use a threshold to determine if two line segments are of the same length – this allows us to handle situations where the draw of a line does not pass the center of a dot.

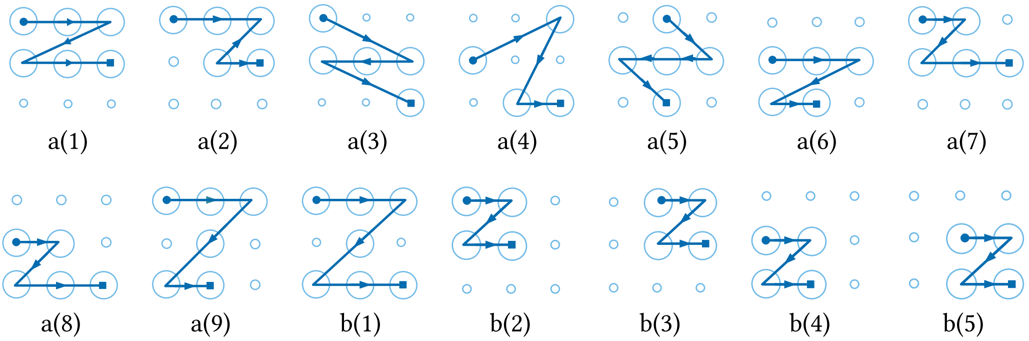


Fig. 10. Possible mappings for the trajectory presented in Figure 2 (d).

total line length is more likely to be the correct pattern. We also considered other additional ranking heuristics by giving a higher priority to patterns that starts from a left dot and has a longer line or starting from the median dot and has shorter line. In total, we considered five ranking heuristics that give a higher priority to patterns of different starting dots and line lengths. Different ranking heuristics would rank the candidate patterns in different order; and we found that our chosen heuristic can lead to a successful attack using the least number of attempts on our test data. We stress that since our attack generates no more than five candidates on a 3 × 3 grid, the order of which candidate patterns to be tested first has negligible impact on the success rate. This is because the Android system by default allows more than five fail attempts and an attacker can test all the five patterns in a short period of time.

6 EXPERIMENTAL SETUP

6.1 Data Collection

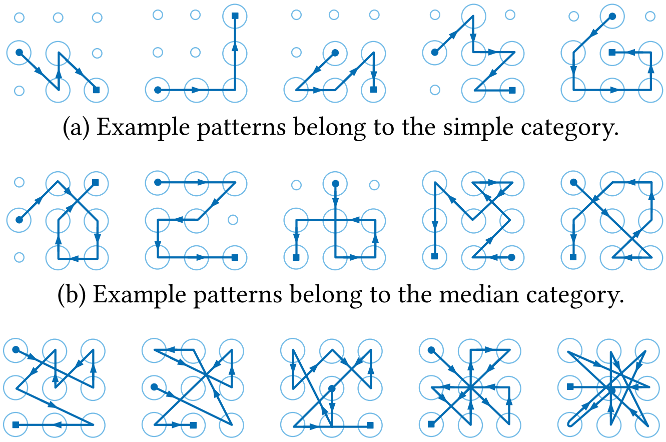
The patterns used in our evaluation were collected from users who use at least one Android device (a smartphone or a tablet) on a daily basis. Our participants include 95 females and 120 males who were undergraduate or postgraduate students in our institution. The majority of our participants are in an age group of under 30.

To collect the patterns, we have conducted a “pen-and-paper" survey by asking participants to fill in an anonymized questionnaire. The questionnaire and survey were approved by the research ethics board (REB) of the host institution. We have made sure that our survey complied with strict privacy regulations. For example, we did not collect any personally identifiable information other than the gender and age group of the participant. Our participants were well informed on the purpose of the study and how the data will be managed and used. The survey forms were distributed as voluntary homework so that the participants can take the survey form away to fill in. Users were invited to return the survey form anonymously within three weeks to a dedicated, locked mailbox, if they wish to participate in the study. To avoid a user submits multiple copies of the same form, each survey form is given a unique, randomly generated 32-digital number.

We have distributed over 1,000 survey forms to be taken to fill at home, for which 220 forms have been returned. The return rate of our questionnaires is in line with the standard survey return rate of 20% [Fox et al. 1988]. By excluding 5 incomplete forms, we have obtained 215 valid forms. These result in 120 unique patterns[7].

Overall, 37.6% of our participants confirmed that they use pattern lock as the screen lock to protect their Android devices on a daily basis; and 33% of those who do not use a pattern as their

[7] Available to be downloaded from: https://dx.doi.org/10.17635/lancaster/researchdata/113



(a) Example patterns belong to the simple category.

(b) Example patterns belong to the median category.

(c) Example patterns belong to the complex category.

Fig. 11. Examples of patterns collected from our participants. Patterns are grouped into *simple*, *median* and *complex* categories, according to their complexity scores.

screen lock said that they are often required to use a pattern for authentication by an application like `Alipay`. Furthermore, 60% of our participants also indicated that the pattern they provided is currently being used or have been used in the past by themselves. Other participants (often those did not use a locking pattern on a daily basis) indicated that they have provided a pattern which they would like to use if a locking pattern is required. While we cannot guarantee that the self-reported patterns are actually used, most of participants confirm that the supplied patterns are the ones that have been used in the past, are currently being used, or somethings they would like to use for sensitive applications. Based on this information, we are confident that the patterns we collected represent some of the real world patterns. Finally, all participants believe that a complex pattern provides stronger protection than a simple counterpart.

6.2 Pattern Complexity Classification

We quantify the complexity of a pattern using the complexity (strength) score proposed in [Sun et al. 2014]. The complexity score, CS_P, of a pattern, P, is defined as:

$$CS_P = S_P \times \log_2(L_P + I_P + O_P) \tag{2}$$

where S_P is the number of connected dots, L_P is the total length of all line segments that form the pattern, I_P is the number of intersections (which are also termed as "knight moves" in some prior work [Von Zezschwitz et al. 2015]) and O_P is the number of overlapping linear segments. To calculate the line length, we assume the length between two horizontally or vertically adjunct dots is one. Thus, our method is independent of the size of the screen and the grid.

Intuitively, the more connected dots (S_P), line segments (L_P), intersections (I_P) and overlapping line segments (O_P) that a pattern has, the more complex it is. There are other methods to quantify the complexity score of pattern locks, including the methods proposed by Song *et al.* [Song et al. 2015] and Andriotis *et al.* [Andriotis et al. 2014]. These methods in general suggest that patterns with more connected dots and intersections are considered to provide stronger security strengths [Aviv and Susanna 2016].

Pattern Grouping. Based the complexity score, we divide the collected patterns into three complexity categories: *simple*, *median* and *complex*. A simple pattern has a score of less than 19, a

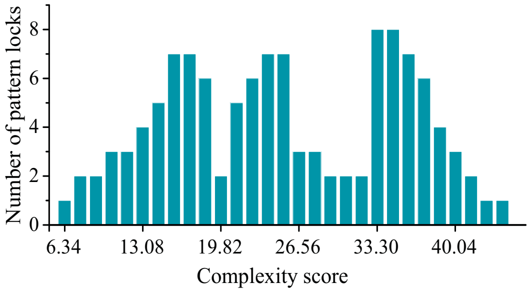


Fig. 12. The distribution of complexity scores for the collected patterns.

Table 2. Screen sizes for the test phones

Screen size	**MI4**	**Honor7**	**Note4**
Height(cm)×Width(cm)	13.9 × 6.9	14.3 × 7.2	15.4 × 7.9

median complex pattern has a score between 19 and 33, and a complex pattern must have a score greater than 33. This classification gives us roughly 40 patterns per category. Figure 11 gives some examples for each category while Figure 12 shows the distribution of these patterns according to their complexity scores. Based on this definition, the complexity scores of the patterns we collected range from 6.4 to 46.8.

6.3 Video Recording and Preprocessing

Recording Devices. We used three smartphones for video recording: an Apple iPhone4S, a Xiaomi MI4 and a Meizu2. Each mobile phone was used to record 40 patterns with a 1080p HD resolution of 30 FPS under different settings described as follows.

Video Recording Setup. By default, we used the Android 3×3 native pattern grid, but we evaluated our approach using other pattern grids with different sizes in Section 7.6. We recorded each pattern under three filming angles, 45, 90 and 135 degrees, by placing the camera on the left-front, front, and right-front of the target device respectively. By default, videos were recorded at a distance of 2 meters from the target device and we evaluated the impact of the filming distance in Section 7.2.

Recording Participators. We recruited ten postgraduate students (five male and five female students) from Northwest University to reproduce the 120 patterns and the 60 most complex patterns (see Section 7.1) on three target mobile phones: a Xiaomi MI4, a Huawei Honor7 and a Samsung Note4. Table 2 lists the screen size for each target mobile phone.

Video Filming and Pattern Drawing. Before recording, our participants were given the opportunity to practice a pattern several times (on average, 10 trials), so that they can draw the pattern at their natural speed. In the experiments, our participants could use any of their fingers for drawing, and they could use more than one finger for drawing. Most of our participants used their index fingers for drawing, one user used his middle finger and one used index and middle fingers for drawing. When drawing the pattern, some participants sat, while others stood, some hold the device by hands, while others placed it on a table. Each pattern was drawn on three target devices and recorded under three filming angles. Thus, for the 120 patterns collected from users, we recorded 1,080 videos in total.

We compare the drawing speed of our participants when they practiced a pattern for 10 times against the speed when they practiced the same pattern for 50 times. We found that there is little difference in the drawing speed. We have also asked our participants to draw five locking patterns used by five different users, and compared the drawing speedup after 10 trails against the drawing speed of the pattern owners. We found that 10 trails are sufficient to achieve a similar drawing speed.

Video Preprocessing. For each video stream, we used the algorithm described in Section 5.1 to cut out the video segment of the unlocking process. We left around 200 to 300 milliseconds of the video segment before and after the pattern unlocking process. To track the fingertip locations, we used Windows Movie Make to highlight two areas of interest on the first frame of the video segment: one area surrounds the fingertip, and the other contains an edge of the phone (see Section 5.2.2).

Experimental Tools and Platform. Our prototyped attacking system built upon a TLD library [Kalal [n. d.]]. The developed software ran on an Intel Core i5 PC with 8GB RAM. The operating system is Windows 10. Our implementation can be ported onto Android or Apple iOS systems, which is our future work. On our evaluation platform, our software takes less than 30 seconds to process a video to produce candidate patterns.

7 EXPERIMENTAL RESULTS

In this section, we first present the overall success rate for cracking the 120 patterns collected from our participants plus the top 60 most complex patterns on a 3×3 pattern grid. We then analyze how the success rate is affected by various filming conditions: the filming distance and angle, the camera shake effect, lighting, the screen size of the mobile device, and the filming cameras. Next, we conduct a limited study to understand how our attack performs when the video only captures the user's fingertip. Finally, we demonstrate that our video-based attack can also be used to crack PIN-based passwords.

7.1 Overall Success Rate

Result 1: *We can successfully crack over 95% of the patterns in five attempts and complex patterns are less secure compared to simple patterns under our attack.*

In this experiment, videos were recorded from a distance of 2 meters away from the target device. This mimics a scenario where the adversary sits at the next table to the user in a public space (e.g. a restaurant). The smartphones used for filming in this experiment were hand-held.

7.1.1 Evaluation using collected user patterns. Figure 13 shows the success rate for cracking different types of patterns within 1, 2, 3, 4 and 5 attempts. We used the 120 patterns that have been collected through our user studies in this experiment. For all the patterns used in this evaluation, our approach does not generate more than five candidate patterns. For complex patterns, we are able to crack all except one (with a 97.5% success rate) *in the first attempt.* For simple and median patterns, the success rate increases with more tries. Using five attempts, we are able to crack all simple patterns and all but one median patterns. The reason that we failed on one median and one complex patterns is because of some blur motions of the video footage (probably caused by the video compressing algorithm), which leads to many tracking failures. But we are able to crack the same pattern using a video filmed by a different device. It is important to note that the native Android system allows up to five failed tries before locking the device [Egelman et al. 2014]. This means, in practice, our approach is able to successfully crack most locking patterns.

Another interesting observation is that in contrast to many people's intuition, complex patterns do not provide stronger protection under our attack – as can be seen by the fact that most of the complex patterns can be cracked in one attempt. This is because although complex patterns can

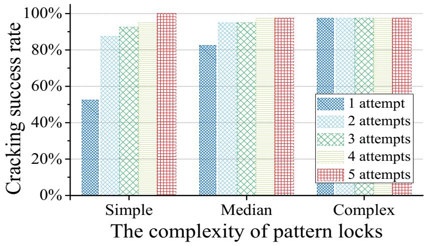


Fig. 13. For each pattern category, the figure shows the success rate using no more than 1, 2, 3, 4 and 5 attempts.

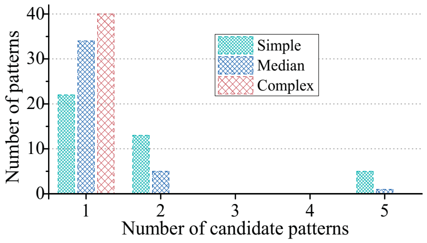
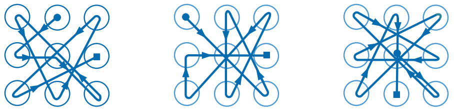


Fig. 14. The distribution of candidate patterns for each category.



complexity score: 43.8

complexity score: 44.7

complexity score: 46.8

Fig. 15. Three most complex patterns on a 3 × 3 grid based on Equation 2.

better protect the user against direct observation techniques like shoulder surfing [Rogers 2007], their unique graphical structures help our algorithms to narrow the possible options down. This is confirmed by Figure 14. It shows that for most median and all complex patterns, our system produces one candidate pattern – the correct one for most of our test cases.

7.1.2 Evaluation on the most complex patterns. We also evaluated our approach using the top 60 most complex patterns (according to Equation 2) on a 3 × 3 grid. To evaluate our approach on a wide range of patterns, we exclude patterns that are simply a rotation to an already chosen pattern. Figure 15 illustrates three highly complex patterns which have a complexity score between 43.8 and 46.8. The three patterns use all the nine dots of the grid and have a larger number of line segments, intersections and overlapping lines when compared to simpler patterns. Because of their complex graphical structures, remembering these patterns using direct observation techniques would be difficult. In this experiment, we can crack all the complex patterns in one attempt. This result reinforces our claim that complex patterns are less secure under video-based attacks.

7.1.3 Evaluation using alternative security metric. In addition to using the complexity metric defined by Equation 2, we also evaluate our attack based on how likely a pattern will be used by users. For this purpose, we use the guessing probability proposed in [Aviv and Susanna 2016]. This metric measures the pattern's strength by considering how likely a pattern is to be guessed. The guessing probability is the likelihood estimation from the hidden Markov model trained on collected, real-world data [Uellenbeck et al. 2013]. The larger the likelihood, the more likely the pattern would have been selected by a user. We choose this metric because a similar security metric based on statistical analysis of real-world passwords haven been widely used in prior studies of text-based passwords [Bonneau 2012; Kelley et al. 2012].

To translate the guessing probability to a frequency score, we first sort all the 120 testing patterns used in this experiment in ascending order, based on their guessing probabilities. By doing so,

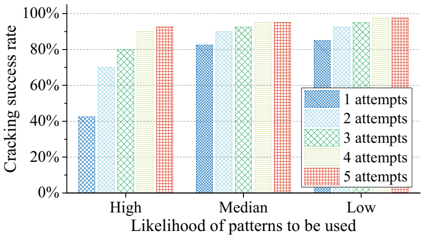


Fig. 16. The success rate when grouping patterns based on Equation 3.

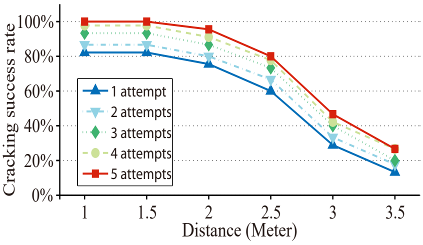


Fig. 17. Impact of the filming distance.

Table 3. % of identical patterns in each category using Equations 2 vs 3

Group	simple v.s. high	median v.s. median	complex v.s. low
% of overlap	72.5%	67.5%	80%

patterns with a higher probability (i.e. more commonly used patterns) will appear after those with a lower probability (less commonly used patterns) on the sorted list. Next, we give each pattern a numeric number (termed *guessing number*), starting from 1 for the first pattern of the sorted list, and we increase the number by 1 as we move down to next (more commonly used) pattern on the sorted list. We then use the following formula to calculate the frequency score, f_P, of a pattern, P:

$$f_P = \log_{10} G_P \tag{3}$$

where G_P is the guessing number of pattern P.

Less commonly used patterns have a smaller guessing number and thus will have a lower frequency score using Equation 3. With this metric in place, we divide our 120 patterns collected from our participants into three groups: *low*, *median* and *high*. Patterns in the *high* group are more likely to be used by users than the patterns in the *low* group. The *high* group has a value of less than 4.2, the *median* group has a score between 4.2 and 5.1, and the *low* group must have a score greater than 5.1. Using this partition strategy, each group has around 40 patterns. Table 3 counts the percentage of identical patterns for each group classified using Equations 2 and 3 respectively. As can be seen from the diagram, while there is a significant degree of overlap, the resulted categorizations are not identical using the two different metrics.

Figure 16 illustrates the cracking success rate for different categories under numbers of attempts. As can be seen from the diagram, the success rate with one attempt for the patterns in the *high* group is 42.5%. This success rate is lower than patterns in other groups. This is because the patterns in the *high* frequently used group are typically simple and symmetry patterns, for which our tracking algorithm produces more than one candidate pattern. This is in line with our observation using the complexity metric defined in Equation 2. Nonetheless, our attack can successfully crack over 90% of the pattern of each group. This confirms that the video-side channel is a real threat for the Android locking pattern.

7.2 Impact of Filming Distances

Result 2: *We can crack over 80% of the patterns in five attempts, if the video was filmed using a smartphone within a distance of 2.5 meters away from the target.*

To illustrate how the filming distance affects the attacking success rate, we used all the 120 collected patterns and we varied the filming distance from 1 meter to 3.5 meters. Figure 17 shows how the cracking success rate changes as the filming distance increases. There are minor discrepancies

Table 4. Tracking precision vs filming distance

Distance	1 m	2 m	3 m	3.5 m
fingertip	100%	98.7%	80.9%	68%
device edge	100%	99.4%	90.6%	69%

in the success rate between this diagram and Figure 13 because we used less patterns in this experiment. When the filming distance is less than 2 meters, our approach can crack all patterns in five attempts. The success rate drops significantly when the filming distance is greater than 2.5 meters. Beyond this point, the quality of the video filmed by a mobile phone tends to drop significantly with many object deformations. The degradation of the video quality makes it difficult for the TLD algorithm to successfully track objects across video frames. This is confirmed by Table 4 which shows that the tracking precision for the fingertip and the device edge drops from around 99% to 68% when the filming distance increases from 2 meters to 3.5 meters. The increased tracking failures result in an increased number of missing points on the tracked trajectory, leading to a deteriorative performance in identifying candidate patterns. Nonetheless, our approach can achieve a high success rate when the filming distance is within 2.5 meters. Such a distance allows an attacker to record the video without raising suspicions in many day-to-day scenarios.

We also evaluated our approach on videos filmed using a entry-level single-lens reflex (SLR) camera, Nikon D90, with a low-end 105mm lens. The SLR camera was placed from a distance of 9 meters away from the target device. For this set of videos, we are able to achieve the same performance when compared to using videos filmed by a mobile phone camera with a 2-meter filming distance. Therefore, in practice, an attacker can also use a professional video recording device to launch the attack from a further distance.

7.3 Impact of Camera Shake

Result 3: *Our method can tolerate a certain degree of camera shake in the hand-held mode.*

In this experiment, we used an IPhone4S smartphone to record how a pattern is drawn on a Huawei Honor7 phone. This experiment was carried out under three settings: *fixed*, *hand-held* and *shaky*, where the filming device was respectively fixed using a tripod, hand-held, and hand-held but with constant movements of approximate 2cm in the horizontal or the vertical directions. The recording device was placed on the left-front, front, and right-front of the target device. In the experiment, we affixed the target device on a table using double-sided tapes.

We use a reference point to quantify camera shake. The point is the center position of an area of the target device. The area is marked by a boundary box on the first frame (see Figure 5). We calculate the difference (in terms of pixels) for where the reference point was seen in two consecutive video frames. We then use the difference to measure the degree of camera shake. Figure 18 shows the cumulative distribution function (CDF) of camera shake under the three different filming settings. Here, the wider the distribution is, the less steady the filming is. The shaky mode is least stable where the difference of the reference point between two video frames can be up to 250 pixels.

Figure 19 shows that our approach has the same performance under the hand-held and the fixed modes. The modest camera shake under the hand-held mode has little impact on performance thanks to our camera-shake calibration method. We observe deteriorative performance under the shaky mode, but the performance degradation is modest (80% vs 97% in 5 attempts). In reality, an attacker would avoid drastic camera shake by firmly holding the video recording device.

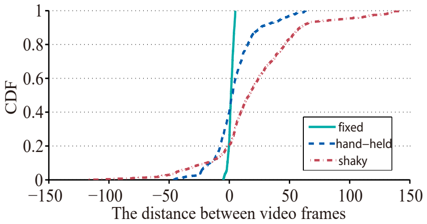


Fig. 18. CDF for different video recording modes.

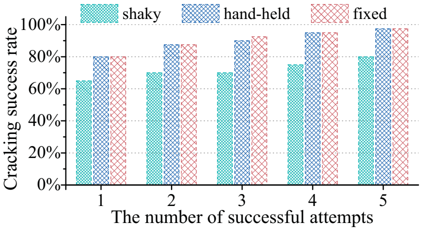


Fig. 19. Impact of camera shake.

Table 5. Lighting Conditions

Scenarios	**Indoor**	**Indoor**	**Indoor**	**Outdoor**
Time	nighttime	nighttime	daytime	daytime
Light Source	warm LED	white fluorescent	sunlight	sunlight
Light Intensity (Lux)	55 – 70	70 – 100	150–240	500–9500

7.4 Impact of Lighting Conditions

Result 4: *Low-light has a negative impact on the success rate of the attack*

In this experiment, videos were recorded under different lighting conditions both indoor and outdoor. The experimental settings are given in Table 5. For each setting, we tested all the 120 patterns on a Xiaomi MI4 phone and used an iPhone4S phone to record the video. The filming camera was placed on the left-front, front, and the right-front of the target device from a distance of 2 meters. Figure 20 shows that the success rate increases when video filming was performed in a brighter lighting condition as the light intensity changes from 55 lux to 9500 lux. This is expected as low-light leads to increased video noise, blurred motions and poor focus, which all have a negative impact on the TLD algorithm. Nonetheless, our attack can still crack over 70% of the patterns in a filming environment of low light. We stress that the impact of the lighting conditions are also camera-dependent where some cameras can better tolerate low-lights than others.

7.5 Impact of Filming Angle Estimation

Result 5: *Our attack performs well when the error of filming angle estimation is less than 5 degrees.*

Recall that our attack needs to transform the fingertip movement trajectory to the user's perspective based on an estimation of the filming angle (Section 5.3). Because our filming angle estimation algorithm gives highly accurate results, we did not find the estimation error to be an issue in our experiments. Nonetheless, it is worth studying how the estimation error affects the success rate of our attack. To do so, we deliberately added an error of 5-10 degrees to the estimation in this experiment.

Figure 21 shows the results of this experiment. When the error is less than ±5 degrees, there is little impact on *complex* patterns and no impact at all on *simple* and *median* patterns. However, an estimation error of more than 10 degrees can significantly affect the success rate. Given such errors, the resulting trajectory after transformations will be significantly different from the correct pattern. For example, when the estimation error is 10 degrees from the true value, on average, 0.8, 2.6 and 4.2 line segments per pattern respectively will be incorrectly labelled for *simple*, *median* and *complex* patterns. This explains why the success rate for complex patterns drops significantly when the filming angle estimation error is greater or equal to 10 degrees.

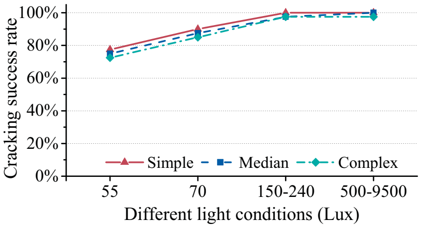


Fig. 20. The cracking success rate within five attempts under different lighting conditions.

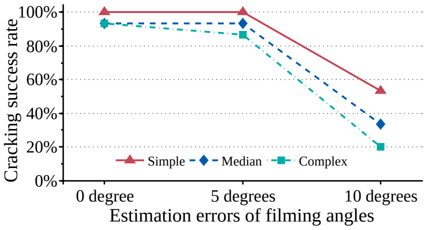
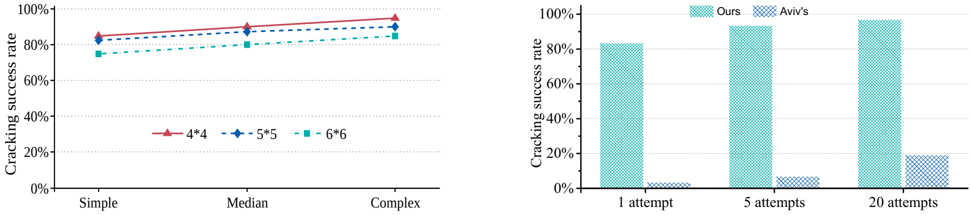


Fig. 21. Impact of estimation errors of filming angles.



(a) success rate for different locking grids

(b) success rate for a 4 × 4 locking grid

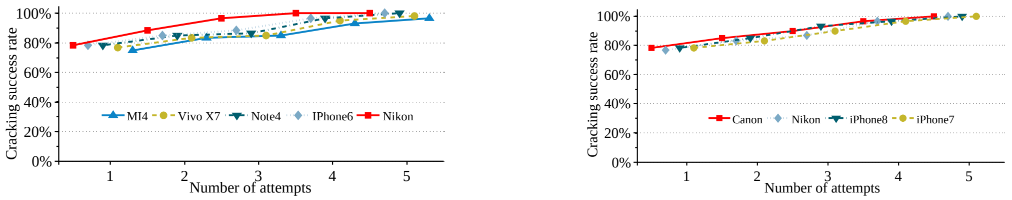
Fig. 22. Success rates for different locking grids.

7.6 Evaluation on Other Pattern Grids

Result 6: *A pattern grid with more dots provides stronger protection but our attack can still crack most of the patterns.*

There are a few applications (such as CyanLock) and customized ROMs available to increase the size of the pattern grid from 3 × 3 to 4 × 4, 5 × 5, and 6 × 6. Although a 3 × 3 grid remains a popular choice (as it is supported by the native Android OS), it is worth studying whether having more touch dots on a pattern grid leads to stronger security. In this experiment, we first ranked all possible patterns for each grid setting in ascending order according to their complexity scores. We then equally divided the patterns into three groups, simple, medium and complex, and asked our participants to randomly select 20 patterns from each group for evaluation. We report the success rate of our attack within five attempts. In the experiments, we have adapted our algorithms for each grid setting by adjusting the algorithm parameters (such as the line direction numbers).

Figure 22 (a) shows the success rate of our attack for different grids. Similar to a 3 × 3 grid, our approach achieves a higher success rate for complex patterns over simple ones. On average, we can crack 90% of the complex patterns. We observed that a grid with more dots does provide stronger protection. For complex patterns, the success rate of our attack drops from 95% on a 4 × 4 grid to 87% on a 6 × 6 grid. For simple patterns, the success rate of our attack drops from 85% on a 4 × 4 grid to 75% on a 6 × 6 grid. This is because a fingertip trajectory in general could be mapped to a larger number of candidates on a grid with more dots. For instance, the pattern shown in Figure 2 (f) can be mapped to 55 candidate patterns on a 6 × 6 grid as opposite to 5 on a 3 × 3 grid. Overall, our attack can crack over 75% (up to 95%) of the patterns within five attempts. One of the purposes of introducing pattern grids with more dots is to allow users to use more complex patterns. However, this experiment suggests that complex patterns remain less security on these grids under our attack.



(a) camera brands

(b) iPhones and SLR cameras

Fig. 23. The cracking success rate for different target screen sizes and filming cameras.

Table 6. Filming camera specs

Parameters	Nikon D90	Canon XC10	iPhone8	iPhone7	IPhone6	Vivo X7	MI4	Note4
Frame Rate (fps)	24	30	30	30	30	30	30	30
Pixels	12.3mp	12mp	12mp	12mp	8mp	13mp	13mp	16mp
Focus (mm)	8	10	5	5	4.15	4	4	4.2
Sensitivity (ISO)	400	500	2500	2500	3200	3200	3000	5000

We also compared our attack against the guessing based attack presented by Aviv *et al.* [Aviv et al. 2015]. This experiment is conducted on a 4 × 4 locking grid using 30 patterns from the dataset used in [Aviv et al. 2015]. The result is shown in Figure 22 (b). Our attack achieves a success rate of over 80% (up to 97%), while the attacking mechanism proposed by Aviv *et al.* gives a success rate of under 20%. It is to note that our attack fails to reconstruct one pattern using 20 attempts because our approach produces more than 20 candidate patterns to be tested. Nonetheless, our attack significantly outperforms the attack described in [Aviv et al. 2015] by effectively exploiting the visual information of pattern structures.

7.7 Impact of Different Camera Brands

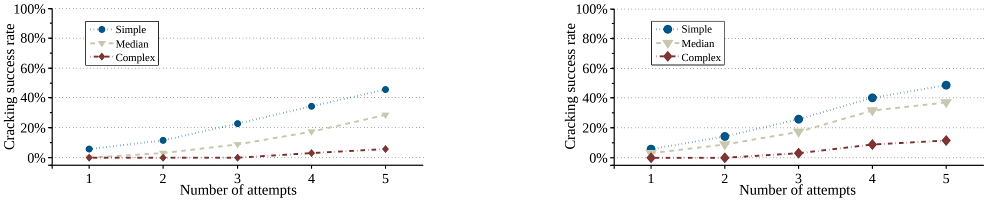
Result 7: *The difference in cameras has little impact on the success rate.*

Intuitively, the effectiveness of our attack can be affected by the video-quality of the filming camera. To understand the impact of the filming camera, we asked 10 participants to draw 90 randomly chosen patterns (30 patterns per pattern category). We record the drawing using eight devices, including two SLR cameras and six mobile phones. Table 6 gives these recording devices and their main performance specs. In this experiment, our target device is a Huawei Honor7 mobile phone. The filming distance is 2 and 9 meters from target device when using a mobile phone and a SLR respectively.

Figure 23 (a) shows that our method can reconstruct all patterns within five trials when using a Nikon SLR, an iPhone6 and a Note4 phone as the recording device. Furthermore, our attack remains effective when using other mobile cameras by giving a success rate of over 97% under five attempts. Figure 23 (b) compares the success rate of our attack when using two other generations of iPhones and different SLR cameras. Again, our method can crack all the testing patterns within five attempts regardless of which filming device is used, albeit a more recent mobile camera does help to achieve a slightly higher success rate using less trials. The experimental results show that our attacking method can work effectively on mainstream mobile phones and SLR cameras.

7.8 Guessing Patterns with Eyes

Result 8: *Our attacking methodology is more likely to succeed compared to direct observation techniques.*



(a) video watching

(b) direct observations

Fig. 24. Success rates of guessing patterns through watching the video (a) or direct observations (b).

In this experiment, we investigate whether an attacker can infer the pattern by simply watching the video or through direct observations. To answer this question, we asked each of our 10 participants to watch 60 videos (where a pattern was drawn by other participants) to guess the pattern. We only played the video segment during which a pattern is drawn to the participant (around 3 seconds per video). To familiarize participants with the process, we played five sample videos and showed the correct patterns at the end of each video to our participants before the experiment. Each participant then had 10 minutes to watch a video and five chances to guess a pattern. They could adjust the playing speed and replay the video multiple times as they wished.

Figure 24 (a) shows the success rate of pattern guessing with bare eyes. Our participants correctly guessed for nearly half of the simple patterns in five attempts. However, they found that it is difficult to infer complex patterns with many line segments, overlapping lines and intersections [Von Zezschwitz et al. 2015]. The success rate of guessing complex patterns is less than 10% in five attempts. This is not a surprising result because although it is possible to correctly guess patterns with simple structures by watching the video, doing so for patterns with more complex structures is much harder.

In accordance with attacking setup, we also asked participants to directly observe how a pattern was drawn from a distance of two meters away from the target device. The intuition behind this evaluation is that human eyes can catch richer information over a digital video camera. The results of this experiment are shown in Figure 24 (b). As can be seen from the diagram, although the success rate is improved compared to directly watching the video, the chances for guessing the correct pattern in 5 attempts are quite low. In fact, the success rates are 48.3%, 38.3% and 11.7% respectively for simple, median and complex patterns.

Note that the success rate of shoulder surfing attacks is relatively low because we asked participants to stand at the same direction as our filming camera faces (the left-front corner from the target device). The observation angle and distance affect the success rate. In practice, an attacker will need to stay much closer to launch the attack, but doing so is more likely to raise suspicion.

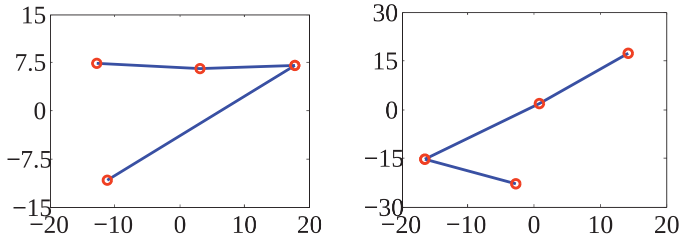
7.9 Attacking PIN-based Passwords

Result 9: *PIN-based passwords are also vulnerable under our video-side channel attack. We can break over 85% of the pin-based passwords within five attempts using a variant method based on our approach.*

An interesting question to ask is, could this method be used to attack PIN-based passwords? To answer this question, we apply our attacking method on 30 4-digital passwords. Among these passwords, 12 of them are most common used passwords (given by a PIN analysis survey conducted by Berry [Berry 2012]). The remaining passwords are randomly selected. For each password, we ask our participants to type in the password on a Xiaomi MI4 phone. Table 7 lists all PIN-based passwords considered in this experiment. In this experiment, we used an Sony 6X phone to record

Table 7. PIN-based passwords used in our experiments

Category	PIN-based Passwords				
Commonly used PINs	1234	1111	1212	1004	2000
	6969	4321	1122	2001	2580
	1357	2468			
Randomly generated PINs	1205	3570	0729	3719	9867
	5946	3451	7403	2209	3560
	1043	4628	5372	2830	7102
	6193	2941	3471		



(a) password: 1234 (b) password: 3570

Fig. 25. Examples of tracked fingertip trajectory comprised of touching points. The red circle represents the touching points.

the video from three angles of the target device: the left-front, front and right-front. The filming distance is 2 meters.

Variant Methodology. Because the differences between the PIN-based password and the pattern lock, we need to adapt our method. The two main differences are summarized as follow: (1) the number of the touch dots is different; and (2) each dot on the PIN pad can be visited multiple times, while each dot can only be visited once on pattern lock. The later difference requires us to identify dots that have been visited multiple times. Our preliminary experiments suggest that we can reconstruct the trajectory of PIN-based password by connecting the touching points[8]. We can obtain the location of touching point by tracking the up-and-down motion direction of the fingertip. This is inspired by the prior work conducted by Shukla *et al.* [Shukla et al. 2014]. To reconstruct PIN-based passwords, our attacking method records some additional geometric information, including the direction and length information. Figure 25 shows the tracked fingertip trajectory using the new attacking method.

Figure 26 compares our method against the attack proposed by Shukla *et al* [Shukla et al. 2014]. With the first attempt, the success rate of our approach is around 52% because most of evaluated passwords have two or more candidates, which is slightly lower compared to the 62% accuracy given by Shukla's. However, both approaches achieve a comparable accuracy when using more than three attempts. This shows that our findings are in line with prior studies on video-based side-channel for PIN-based passwords.

[8] Touching points are the points that are tracked when the user's fingertip touches the screen.

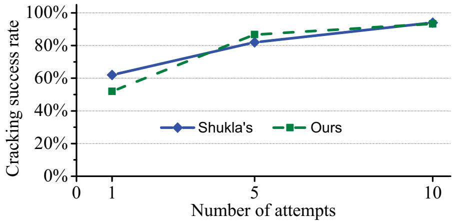


Fig. 26. The success rate of cracking PIN-based passwords with different number of attempts.

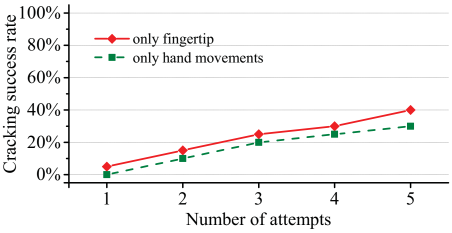


Fig. 27. The cracking success rate drops significantly when the video footage only captures the user's fingertip or hand movements.

7.10 Limited Study

Our attacking method requires the user's fingertip and part of the device to be seen in the video footage. As described in Section 5.2.2, this is essential for calculating the coordinates of the fingertip and for camera shake calibration. An interesting question to ask is that: "can we relax this requirement?" That is, will the attack still be effective if the video footage only captures the user's fingertip or hand movements. If our findings suggest that the success of the attack requires seeing part of the device, a potential countermeasure is to educate users to cover their devices when entering their patterns. To conduct this limited study, we ensure that the video only captures the user's fingertip or hand movements during recording. This experiment is performed on 20 patterns randomly selected by our participants.

Figure 27 shows the success rate of our attacking method is low when the device is not seen in the video footage. The success rate is even lower when the camera only captures the hand movements but not the fingertips. With only the fingertip or the hand location, our attack fails to cancel the camera shake effect, leading to a low-quality fingertip movement trajectory. This leads to a worse success rate (less than 50%, 30% and 20% for simple, medium and complex patterns respectively).

8 COUNTERMEASURES

In this section, we first analyze the dominating factors of our attack. According to these factors, we illustrate the reasons that some possible countermeasures approved by the public are also vulnerable. At last, we propose a possible remedy, a variant of pattern lock mechanism, which can effectively protect the mobile phone from the video-based attacks.

8.1 Possible Countermeasures

The success of our attack depends on the following three factors: (1) knowledge of the pattern grid; (2) a decent quality video footage allowing the algorithm to track the fingertip movement; (3) successfully identifying a video segment that captures the entire process of pattern drawing.

For the first factor, the attacker can obtain the relevant information by simply looking at the pattern grid of the target operating system or application. Randomization techniques such as randomized pictures [Biddle et al. 2012; Schneegass et al. 2014; Siadati et al. 2015; Zezschwitz et al. 2013], which randomly shuffle the location of touch points each time, could be a solution. However, randomization-based solutions often come at the cost of poorer usability. This can prevent them to be used at a large scale. Regarding the second factor, there are ways, such as KALEIDO [Zhang et al. 2015], to prevent unauthorized videotaping by dynamically changing the colour and brightness of the screen to confuse the filming camera. A non-technical solution for this aspect would be to educate users to fully cover their fingers when drawing a pattern. But doing this on a large-screen device could be awkward especially when the device is held by one hand. However, changing users' behavior is non-trivial [Chimalakonda and Nori 2013].

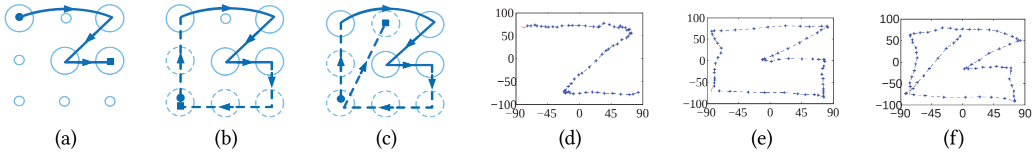


Fig. 28. Examples of our new pattern lock mechanism. A true pattern of solid lines is shown in (a). We propose to make two changes to form a pattern. Our first change allows the user to skip some dots when creating a pattern. For example, in (a) the central dot in the first line is skipped. Our second change requires the user to draw a given random pattern structure (e.g. the dash lines in b and c) before or after drawing the true pattern. The tracked fingertip trajectories of a, b, and c are shown in d, e, f respectively. The first change makes it difficult for the tracking algorithm to identify which dots are skipped, and the second change forces the attacker to use multiple video recordings to identify the true pattern. As a result, this new pattern lock mechanism decreases the effectiveness of the video-based attack.

For the third factor, the attacker's solution depends on the type of the pattern. For a screen lock, pattern drawing is the first activity (except for receiving a phone call or making an emergency call) when the device is retrieved. Therefore, identifying the video segment is straightforward. When the pattern is used by applications, we have observed that users typically pause for a few seconds before or after entering the pattern. Therefore, an experienced attacker should also be able to identify the video segment in case our automatic algorithm (presented in Section 5.1) fails to do so. A potential countermeasure is to mix pattern unlocking with other on-screen activities. For examples, before and after pattern drawing, the system can ask the user to type in a sentence using a Swype-like method or to draw some graphical shapes. The problem of this approach is it may annoy users by asking them to do more, especially for screen unlocking – an activity that is performed many times a day.

8.2 A Feasible Remedy

We propose a countermeasure to make it difficult to obtain a meaningful video footage. When designing the approach, we try to find a balance between the usability and the security. Our approach requires making two changes to the pattern lock: (1) when forming a pattern the user can skip some of the dots in a vertical, horizontal, or diagonal line (e.g. the central dot of the top line is skipped in Figure 28 (a)); (2) before or after drawing the correct pattern, the user is asked to draw a given random pattern to confuse the attacker (e.g. sub-figures (b) and (c) in Figure 28). For the second change, we relax the rules for creating a pattern to allow a touching dot to be visited multiple times (e.g. the bottom-left dot at Figure 28 (c) is visited twice). Further, for purpose of increasing the number of candidate patterns, we also change the rule that a previously unvisited dot can be bypassed if it is part of a horizontal, vertical or diagonal line segment of the pattern.

The two changes mentioned above can significantly decrease the effectiveness of the attack. Our first change allows the user to skip some dots, which makes it difficult for the tracking algorithm to identify which dots are skipped (because the video-camera is not able to precisely capture depth information). The second change makes it harder for the attacker to identify which part of the track pattern is the true pattern. Figures 28 (d) - (f) respectively show the tracked fingertip trajectory of the patterns shown in Figures 28 (a) - (c). As can be seen from the tracked trajectories, using a pattern that is directly mapped from the trajectory will lead to a failed attempt. Because each time the system will ask the user to draw a random pattern structure, directly use the tracked pattern (ignore the skipping dots) will not be accepted by the system.

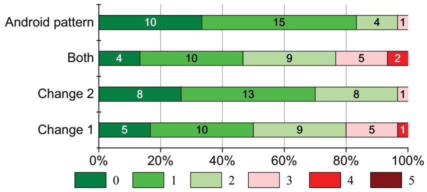


Fig. 29. The usability of our countermeasure has little negative effect comparing to Android pattern lock.

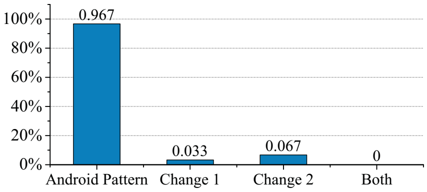


Fig. 30. The cracking success rate drops significantly when using our countermeasure.

8.3 Usability Study of the Proposed Countermeasure

To understand the usability of our countermeasures, we conduct a user study. We ask each of our 30 volunteers to apply the countermeasure to three randomly assigned Android patterns (90 unique patterns in total). Among the three revised patterns came up by a participant, two must contain one of the changes we proposed, and the other one must contain both changes. We ask our participants to draw their revised patterns 10 times. We then ask our participants to rate their experience on using the revised patterns. The rating is based on a score ranging between 0 and 5 – where 5 means there is a significant impact on their experience and 0 means there is no impact on their experience.

Figure 29 shows that comparing to the original pattern lock mechanism, our proposed changes have little impact on the user experience. On average, it takes 2.5 seconds for a user to draw a new pattern. This is slightly longer than drawing an standard Android pattern, which takes 2 seconds on average. Nevertheless, 21 out of our 30 participants consider the increased time to be negligible and think there is no impact on their experience; and other 7 users think the changes are acceptable. There are two users who specifically see two patterns that combine both changes to be a poor design, and one of them also rates a pattern that contains the first change to result in poor user experience. However, these two users also consider one of the original patterns to be difficult to draw, because the pattern has a complex structure; therefore, we think the poor user experience is largely due to the complexity of the original pattern. We also ask our participants whether they want to use our proposed changes. Among 30 participants, 25 and 27 participants like change 1 and change 2, respectively, and 23 participants are happy with both changes. Based on the user study, we conclude that our countermeasure does not significantly affect the user experience for most of our participants.

To evaluate the strength of our countermeasure, we applied our attacking method to reconstruct the revised and the original patterns. Figure 30 shows that our countermeasure significantly reduces the success rate of the attack. The attack can successfully crack 96.7% of the original patterns. However, it can only successfully reconstruct two of the revised patterns, leading to a success rate of less than 7%. After having a close look at the two successful cracked revised patterns, we found that little changes have been introduced to the original patterns. In other words, better changes could be introduced by the users to increase the strength of the revised patterns. This experiment confirms that randomness can improve the security of pattern lock under our attack.

We stress that our countermeasure is not a panacea. In fact, finding a balance between the usability and security for graphical passwords remains an open problem [Abdullah et al. 2008]. Even with our countermeasure, an attacker can still use multiple videos that record the same user when doing pattern drawing to find the common pattern structure (which will likely to be the true pattern). Therefore, while our countermeasure is simple to implement and can increase the overhead for performing the attack, an experienced adversary could still successfully launch the attack.

9 IMPLICATIONS

While pattern lock is preferable by many users [Bruggen 2014], this work shows that it is vulnerable under video-based attacks. Our attack is able to break most patterns in five attempts. Considering Android allows five failed attempts before automatically locking the device, our work shows that this default threshold is unsafe. We demonstrated that, in contrast to many users' perception, complex patterns actually do not provide stronger protection over simple patterns under our attack.

It is worth mentioning that our approach is only one of the many attacking methods that researchers have demonstrated. Examples of these attacks include video-based attacks on keystroke-based authentication [Shukla et al. 2014; Yue et al. 2014], sensor-based attacks for pattern lock [Zhang et al. 2016]. Authentication methods that combine different authentication methods [Ling et al. 2016; Luca et al. 2012; Mannan and van Oorschot 2007; Stefan et al. 2012] to constantly check the user's identity could be a solution.

10 RELATED WORK

Our work lies at the intersection between computer vision based attacks and cracking graphical- and touch-based authentication methods. This work brings together techniques developed in the domain of computer vision and motion tracking to develop a new attack. Our work is the first attempt of reconstructing a locking pattern from a video footage without capturing the content displayed on the screen.

Computer Vision-based Attacks. No work has targeted using video footage to crack Android pattern lock and this is the first to do so. Our work is inspired by the work presented by Shukla *et al.* [Shukla et al. 2014] on video-based attacks of PIN-based passwords. In addition to addressing the new challenges highlighted in Section 1, our work differs to their approach in two ways. Firstly, we target a different authentication method, i.e. graphical-based passwords are fundamentally different from PIN-based passwords. Secondly, our approach does not require knowledge of the size of the screen or the grid. Other work in the area including [Yue et al. 2014] which attacks PIN-based passwords by analyzing how the screen brightness changes when entering a password. But the subtle changes of the screen brightness can be dramatically affected by the lighting condition. This restricts the application of their approach. There is a body of work using reflections to recover information typed by the user [Backes et al. 2009; Kuhn 2002; Raguram et al. 2011; Xu et al. 2013]. These schemes require having a clear vision of the content displayed on the screen while our approach does not have such a requirement.

Attacks on Touch-based Authentication. Ballard *et al.* implemented a forgery attack on handwriting authentication [Ballard et al. 2007]. Using a small number of training examples, they achieve a high success rate for this attack. More recently, Serwadda *et al.* show that a simple robot can achieve high penetration rates against touch-based authentication systems by analyzing on-screen gestures including swiping and zooming [Serwadda and Phoha 2013]. Maggi *et al.* present a fast, automatic shoulder surfing attack against touchscreen keyboards [Maggi et al. 2011]. In this article, we present a new, video-based attack for graphical-based passwords. Research in this area all demonstrates the need for a closer look at the security risks of touch-based authentication.

Cracking Graphical-based Passwords. Aviv *et al.* demonstrated that it is possible to reconstruct a locking pattern by analyzing the oily residues left on the screen [Aviv et al. 2010]. This method is highly restricted as oily residues can be messed up by any on-screen activities after pattern drawing. Abedlrahman *et al.* explored the fact that PINs or patterns are likely to be recognized by tracking the heat left on the screen [Abdelrahman et al. 2017]. Likewise, their approach is significantly disrupted by other on-screen operations after drawing the PIN or pattern. Zhang *et al.* exploit the WiFi signal interferences caused by finger motions to recover patterns [Zhang et al. 2016]. Their method requires a complex setup and is highly sensitive to moving objects of the environment because the WiFi signal can be disrupted by a moving object.

Study of Android Pattern Lock. Uellebenk *et al.* study how people use Android pattern lock on a daily basis [Uellenbeck et al. 2013]. They found that although there is a large number of Android patterns, in practice many people only use a small set of them due to users' bias in generating patterns. Løge explored the correlation between human's characteristics (e.g. ages and genders) and the choice of patterns [Løge 2015]. Her study shows that users have a bias in selecting the starting dot to form a pattern and people tend to use complex patterns for sensitive applications. Aviv *et al.* conducted a large user study to understand the security of the Android graphical based passwords [Aviv and Fichter 2014]. They analyzed the security and usability preference of users, using six visual features of the pattern lock including pattern length, number of crosses, etc. After conducting a larger user study, they developed a brutal-force algorithm to crack the pattern lock [Aviv et al. 2015]. Their results show that 15% and 20% of the patterns generated on a grid of 3×3 and 4×4 dots respectively can be cracked within 20 guesses.

Motion Tracking. In addition to TLD, there are other methods proposed in the past for tracking object motions. Some of them apply image analysis to track the hand and gesture motions from video footage [Beh et al. 2014; Stenger et al. 2006; Yang et al. 2002]. In this article we do not seek to advance the field of motion tracking. Instead we demonstrate that a new attack can be built using classical motion tracking algorithms. We show that the attack presented in this work can be a serious threat for Android pattern lock. This has never been attempted in prior work on motion tracking.

11 CONCLUSIONS

This article has presented a novel video-based side-channel attacking method for Android pattern lock. The proposed method is able to successfully break most locking patterns in five attempts, based on the video footage of the entire unlocking process, filmed a distance of $2 - 3$ meters away from the target device using a mobile phone rear camera. The attack is achieved by employing a computer vision algorithm to track the fingertip movement from the video, and then using the geometry information of the fingertip movement trajectory to identify the most likely patterns to be tested on the target device. Our approach was evaluated using 120 unique patterns collected from 215 independent users and some of the most complex patterns. The experimental results show that our approach is able to successfully crack over 95% of the patterns in five attempts. We show that, in contrast to many people's belief, complex pattern actually provides weaker protection over simple patterns under our attack. Our study demonstrate that Android pattern lock is vulnerable to video-based side-channel attacks and redesigning of pattern-based authentication is needed to offer stronger security guarantee. In order to remit this attack, we analyze the successful factors of our attack and propose to a variant of the Android pattern lock mechanism. We show that our countermeasure can effectively protect mobile devices from video-based attacks.

12 ACKNOWLEDGEMENT

This work was partly supported by the National Natural Science Foundation of China (NSFC) through grant agreements 61672427, 61672428 and 61572402; the China Computer Federation-NSFOCUS Kunpeng Grant (CCF-NSFOCUS2017009); the UK Engineering and Physical Sciences Research Council (EPSRC) through grant agreements EP/M01567X/1 (SANDeRs) and EP/M015793/1 (DIVIDEND); and the Royal Society International Collaboration Grant (IE161012). The user patterns used to evaluate this work are openly available from the Lancaster University data archive at https://dx.doi.org/10.17635/lancaster/researchdata/113.

REFERENCES

Yomna Abdelrahman, Mohamed Khamis, Stefan Schneegass, and Florian Alt. 2017. Stay cool! understanding thermal attacks on mobile-based user authentication. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 3751–3763.

Muhammad Daniel Hafiz Abdullah, Abdul Hanan Abdullah, Norafida Ithnin, and Hazinah Kutty Mammi. 2008. Towards Identifying Usability and Security Features of Graphical Password in Knowledge Based Authentication Technique. In *Second Asia International Conference on Modelling & Simulation*. 396–403.

Panagiotis Andriotis, Theo Tryfonas, and George Oikonomou. 2014. *Complexity Metrics and User Strength Perceptions of the Pattern-Lock Graphical Authentication Method.* 115–126 pages.

Adam J Aviv, Devon Budzitowski, and Ravi Kuber. 2015. Is Bigger Better? Comparing User-Generated Passwords on 3x3 vs. 4x4 Grid Sizes for Android's Pattern Unlock. In *Computer Security Applications Conference*. 301–310.

Adam J Aviv and Dane Fichter. 2014. Understanding visual perceptions of usability and security of Android's graphical password pattern. In *Computer Security Applications Conference*. 286–295.

Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith. 2010. Smudge attacks on smartphone touch screens. In *Proceedings of the 4th USENIX conference on Offensive technologies*. 1–7.

Adam J. Aviv and Heidt Susanna. 2016. Refining Graphical Password Strength Meters for Android Phones. In *USENIX Twelfth Symposium on Usable Privacy and Security (SOUPS)*.

Michael Backes, Tongbo Chen, Markus Duermuth, Hendrik P. A Lensch, and Martin Welk. 2009. Tempest in a Teapot: Compromising Reflections Revisited. In *Security and Privacy, 2009 IEEE Symposium on*. 315–327.

Lucas Ballard, Daniel Lopresti, and Fabian Monrose. 2007. Forgery Quality and Its Implications for Behavioral Biometric Security. *IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics A Publication of the IEEE Systems Man & Cybernetics Society* 37, 5 (2007), 1107–1118.

Davide Balzarotti, Marco Cova, and Giovanni Vigna. 2008. ClearShot: Eavesdropping on Keyboard Input from Video. In *IEEE Symposium on Security and Privacy*. 170–183.

Jounghoon Beh, David Han, and Hanseok Ko. 2014. *Rule-based trajectory segmentation for modeling hand motion trajectory.* Elsevier Science Inc. 1586âĂŞ1601 pages.

Nick Berry. 2012. PIN analysis. Available: http://www.datagenetics.com/blog/september32012/index.html. (2012).

Robert Biddle, Sonia Chiasson, and P C Van Oorschot. 2012. Graphical passwords: Learning from the first twelve years. *ACM Computing Surveys (CSUR)* (2012).

Joseph Bonneau. 2012. The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy (SP '12)*. 538–552.

Dirk Van Bruggen. 2014. *Studying the Impact of Security Awareness Efforts on User Behavior.* Ph.D. Dissertation. University of Notre Dame.

Sridhar Chimalakonda and Kesav V. Nori. 2013. What makes it hard to teach software engineering to end users? some directions from adaptive and personalized learning. 128, 6 (2013), 324–328.

Antonella De Angeli, Lynne Coventry, Graham Johnson, and Karen Renaud. 2005. Is a picture really worth a thousand words? Exploring the feasibility of graphical authentication systems. *International Journal of Human-Computer Studies* 63, 1âĂŞ2 (2005), 128–152.

Serge Egelman, Sakshi Jain, Rebecca S Portnoff, Kerwell Liao, Sunny Consolvo, and David Wagner. 2014. Are You Ready to Lock?. In *ACM Sigsac Conference on Computer and Communications Security*. 750–761.

Malin Eiband, Mohamed Khamis, Emanuel Von Zezschwitz, Heinrich Hussmann, and Florian Alt. 2017. Understanding Shoulder Surfing in the Wild: Stories from Users and Observers. In *CHI Conference on Human Factors in Computing Systems*.

Richard J. Fox, Melvin R. Crask, and Jonghoon Kim. 1988. MAIL SURVEY RESPONSE RATE A META-ANALYSIS OF SELECTED TECHNIQUES FOR INDUCING RESPONSE. *Public Opinion Quarterly* 52, 4 (1988), 467–491.

von Gioi R Grompone, J Jakubowicz, J. M. Morel, and G Randall. 2010. LSD: a fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 32, 4 (2010), 722–32.

Zdenek Kalal. [n. d.]. TLD: Tracking-Learning-Detection. Available: http://kahlan.eps.surrey.ac.uk/featurespace/tld/. ([n. d.]).

Zdenek Kalal, Mikolajczyk K, and Matas J. 2011. Tracking-Learning-Detection. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 34, 7 (2011), 1409–22.

Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio Lopez. 2012. Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy (SP '12).* 523–537.

Markus Guenther Kuhn. 2002. *Compromising emanations: eavesdropping risks of computer displays.* Ph.D. Dissertation. University of Cambridge.

Michael H. Kutner, Christopher J. Nachtsheim, and John Neter. 2004. Applied Linear Regression Models (5th Ed.). *Technometrics* 26, 4 (2004).

Zhen Ling, Junzhou Luo, Qi Chen, Qinggang Yue, Ming Yang, Wei Yu, and Xinwen Fu. 2016. Secure fingertip mouse for mobile devices. In *IEEE INFOCOM 2016 - the IEEE International Conference on Computer Communications.* 1–9.

Marte Dybevik Løge. 2015. *Tell Me Who You Are and I Will Tell You Your Unlock Pattern.* Master's thesis. Norwegian University of Science and Technology.

Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. 2012. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Sigchi Conference on Human Factors in Computing Systems.* 987–996.

Federico Maggi, Alberto Volpatto, Simone Gasparini, Giacomo Boracchi, and Stefano Zanero. 2011. Poster: fast, automatic iPhone shoulder surfing. In *ACM Conference on Computer and Communications Security.* 805–808.

Mohammad Mannan and Paul C van Oorschot. 2007. Using a personal device to strengthen password authentication from an untrusted computer. In *Financial Cryptography and Data Security.* Springer, 88–103.

Michelle L. Mazurek, Saranga Komanduri, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Patrick Gage Kelley, Richard Shay, and Blase Ur. 2013. Measuring Password Guessability for an Entire University. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security.* ACM, 173–186.

Rahul Raguram, Andrew M. White, Dibyendusekhar Goswami, Fabian Monrose, and Jan Michael Frahm. 2011. iSpy:automatic reconstruction of typed input from compromising reflections. In *ACM Conference on Computer and Communications Security.* 527–536.

J Rogers. 2007. Please enter your four-digit pin. *Financial Services Technology* (2007).

Stefan Schneegass, Frank Steimle, Andreas Bulling, Florian Alt, and Albrecht Schmidt. 2014. SmudgeSafe: geometric image transformations for smudge-resistant user authentication. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing.* 775–786.

Abdul Serwadda and Vir V. Phoha. 2013. When kids' toys breach mobile phone security. In *ACM Sigsac Conference on Computer & Communications Security.* 599–610.

Diksha Shukla, Rajesh Kumar, Abdul Serwadda, and Vir V Phoha. 2014. Beware, Your Hands Reveal Your Secrets!. In *ACM CCS.* 904–917.

Hossein Siadati, Payas Gupta, Sarah Smith, Nasir Memon, and Mustaque Ahamad. 2015. Fortifying Android Patterns using Persuasive Security Framework. In *UBICOMM 2015, The Ninth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies.*

Youngbae Song, Geumhwan Cho, Seongyeol Oh, Hyoungshick Kim, and Jun Ho Huh. 2015. On the Effectiveness of Pattern Lock Strength Meters:Measuring the Strength of Real World Pattern Locks. In *ACM Conference on Human Factors in Computing Systems.* 2343–2352.

Lionel Standing, Jerry Conezio, and Ralph Norman Haber. 1970. Perception and memory for pictures: Single-trial learning of 2500 visual stimuli. *Psychonomic Science* 19, 2 (1970), 73–74.

Deian Stefan, Xiaokui Shu, and Danfeng Yao. 2012. Robustness of keystroke-dynamics based biometrics against synthetic forgeries. *Computers & Security* 31, 1 (2012), 109–121.

B Stenger, A Thayananthan, P. H. Torr, and R Cipolla. 2006. Model-based hand tracking using a hierarchical Bayesian filter. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 28, 9 (2006), 1372–84.

Chen Sun, Yang Wang, and Jun Zheng. 2014. Dissecting pattern unlock: The effect of pattern strength meter on pattern selection. *Journal of Information Security & Applications* 19, 4âĂŞ5 (2014), 308–320.

Antonio Torralba and Aude Oliva. 2002. Depth Estimation from Image Structure. *Pattern Analysis & Machine Intelligence IEEE Transactions on* 24, 9 (2002), 1226–1238.

Sebastian Uellenbeck, Christopher Wolf, and Thorsten Holz. 2013. Quantifying the security of graphical passwords:the case of android unlock patterns. In *ACM Sigsac Conference on Computer & Communications Security.* 161–172.

Emanuel Von Zezschwitz, Alexander De Luca, Philipp Janssen, and Heinrich Hussmann. 2015. Easy to Draw, but Hard to Trace?: On the Observability of Grid-based (Un)lock Patterns. In *ACM Conference on Human Factors in Computing Systems*. 2339–2342.

Roman Weiss and Alexander De Luca. 2008. PassShapes: utilizing stroke based authentication to increase password memorability. In *Nordic Conference on Human-Computer Interaction: Building Bridges*. 383–392.

Yi Xu, Jared Heinly, Andrew M White, Fabian Monrose, and Jan Michael Frahm. 2013. Seeing Double: Reconstructing Obscured Typed Input from Multiple Compromising Reflections, Around the Corner. In *ACM Conference on Computer and Communications Security*. 1063–1074.

Ming Hsuan Yang, Narendra Ahuja, and Mark Tabb. 2002. Extraction of 2D Motion Trajectories and Its Application to Hand Gesture Recognition. (2002), 1061–1074.

Guixin Ye, Zhanyong Tang, Dingyi Fang, Xiaojiang Chen, Kwang In Kim, Ben Taylor, and Zheng Wang. 2017. Cracking Android pattern lock in five attempts. In *The Network and Distributed System Security Symposium (NDSS)*.

Qinggang Yue, Zhen Ling, Benyuan Liu, Xinwen Fu, and Wei Zhao. 2014. Blind Recognition of Touched Keys: Attack and Countermeasures. *Computer Science* (2014).

Emanuel Von Zezschwitz, Anton Koslow, Alexander De Luca, and Heinrich Hussmann. 2013. Making graphic-based authentication secure against smudge attacks. (2013), 277–286.

Jie Zhang, Xiaolong Zheng, Zhanyong Tang, Tianzhang Xing, Xiaojiang Chen, Dingyi Fang, Rong Li, Xiaoqing Gong, and Feng Chen. 2016. Privacy Leakage in Mobile Sensing: Your Unlock Passwords Can Be Leaked through Wireless Hotspot Functionality. *Mobile Information Systems* 2016, 2 (2016), 1–14.

Lan Zhang, Cheng Bo, Jiahui Hou, Xiang Yang Li, Yu Wang, Kebin Liu, and Yunhao Liu. 2015. Kaleido: You Can Watch It But Cannot Record It. In *International Conference on Mobile Computing and NETWORKING*. 372–385.

Received October 2017; revised January 2018; accepted June 2018