A Survey of Novel Countermeasures Against Mobile Lock Screen Keystroke Inference Attacks

Bill Wu billwu@mit.edu
Deepankar Gupta deepg14@mit.edu
Eric Jiang jiange@mit.edu
Rishi Sundaresan rishisun@mit.edu

Abstract

Mobile device security is a major research focus due to the widespread use of smartphones, tablets, and similar devices. While many applications exist to protect user data, their value is eliminated when an attacker gains information about user passwords. In this study, we survey four external attacks where attackers gain information about user passwords inputted on soft keyboards through keystroke inference. The attacks analyze external cues such as tablet motion and gaze to learn about the victims' passwords. We then propose novel countermeasures to the attacks and analyze their impacts on user experience and security.

1. Introduction

When one thinks of mobile security, usually any of the following come to mind (depending on the type of mobile system): file and disk encryption, password managers, Google's Play Protect, network firewalls, sandboxed apps, malware protection, authentication, and much more [SSC+08] [MK16] [GVS20] [AND20]. However, a phone's contents are protected by a user-inputted password that can be as simple as a 4-digit PIN or an Android lock pattern. If one's phone password is exposed, most of the security efforts listed above become utterly useless. If an attacker were to have access to a victim's unlocked phone, they would not only have access to texts, calls, and personal media, but also access (through cookies) to email, social media, internet accounts, and much more. What is especially worrying is the ability to reset account details for sites that require two-factor authentication, since the attacker would possess full access to the victim's physical "trusted device". Along with publicly available information like the last 4 digits of the victim's SSN, their address, and other personal information, it would be entirely possible for an adversary to obtain the victim's bank credentials, to blackmail the victim, or to cause any other undesirable outcome. These threats set the motivation for the rest of the ideas in this paper.

With the advent of biometric identification and Apple Face ID, mobile devices are increasingly more resistant to lock screen password leaks. However, many devices still use a phone password inputted via a "soft keyboard", which usually consists of a 4-digit or 6-digit PIN, a geometric pattern lock, or an alphabetical password entered via a traditional QWERTY keyboard. These input methods all inherently require a user to tap or swipe in a consistent pattern, which opens an entire class of attacks called keystroke inference attacks. These attacks aim to infer keystrokes through information learned from various side channels or by directly observation. The attacks pose a significant threat to mobile users since in an insecure public environment, if a victim were to unlock their mobile device, an attacker could steal the user's password and launch any number of further attacks mentioned before, compromising mobile security and user privacy.

In this paper, we present countermeasures that aim to mitigate keystroke inference attacks on mobile lock screen passwords, from a variety of different angles that have already been explored in the existing literature on touchscreen device security. The structure of the following sections can be summarized as follows:

1. We introduce the background for lock screen passwords in order to better understand potential vulnerabilities.
2. We summarize four different keystroke inference attacks and for each, we propose countermeasures and analyze their effectiveness and practicality.
3. We suggest potential applications for the proposed countermeasures, discuss other ideas that we considered throughout the research process, and explore future areas of work.

2. Background

2.1. System Functionality

Although mobile devices differ in many ways, including but not limited to the company that produced them and the

operating system they run, the lock screen password system that they use follows the same template. The user of the mobile device can set their lockscreen to be protected by either an alphanumeric password, a PIN number, or a pattern-based password [HLE16]. Not all mobile devices feature all of these password types. Once the user chooses a password type and an appropriate password, the lock screen password system functions as follows from the user's perspective:

1. The user swipes to access password input screen. This screen shows a prompt (a request for a string of characters, a request for a PIN number, or a request for a pattern on a grid of points or circles).
2. The user responds to the prompt and enters his or her password attempt. In the case of alphanumeric passwords or PIN numbers, an appropriate soft keyboard appears. In a gridlock setting, the grid itself can be thought of as the soft keyboard.
3. The screen responds to the attempt. If password is correct, the phone unlocks and reveals the device's home screen. If the password is incorrect, the user is given an indication of it.
4. In many modern mobile devices, if the user makes multiple incorrect attempts, he or she will be prevented from entering another password for a variable amount of time.

2.2. Keystrokes

When the user inputs a password in any of the three formats through a soft keyboard, a single depression on the keyboard is referred to as a *keystroke*. To be more specific, for an alphanumeric password, the action of a user pressing on the specific region on the soft keyboard to input a single character of the password would be considered a keystroke. A similar definition holds for PIN number passwords. In the case of a pattern-based password, the grid can be viewed as a soft keyboard. The keystrokes in the grid are entered via swiping on the soft keyboard grid. With this context, each swipe on the grid can be viewed as a sequence of keystrokes.

3. Plausible Attacks and Countermeasures

As discussed in the previous section, the input of a password to unlock a mobile device relies on a series of keystrokes on a soft keyboard. If an adversary were to be able to determine the keystrokes that a mobile device user made to input his or her password, then the adversary would be able to guess the user's password with decent probability, perhaps even being able to reconstruct the correct password. Many recent attacks take advantage of this notion and are aptly dubbed *keystroke-inference attacks*. In this section, we cover four keystroke-inference attacks, spanning three types: motion-based, wifi-based, and video-based. For each attack, we will give an overview of the attack's pipeline. We will then follow up with our proposed countermeasures for dealing with such attacks, as well as an analysis regarding the strengths and drawbacks of our proposed countermeasures.

3.1. Tablet Motion Based Attack: VISIBLE

3.1.1 Attack Description

Visible is an external attack where the attacker attempts to gain information about the victim's keystrokes in tablets [SJC+16b]. The setup is as follows:

The user is typing on an iPad/Tablet, and the attacker is positioned across from the victim such that the attacker is seeing the back of the tablet. Using a high-resolution camera or camcorder, the attacker videotapes the back of the tablet as the victim is pressing keys on a soft keyboard. The attacker also notes elements of the scene (distances between the camera and tablet, angle offset, etc.). This setup is detailed in 1. To get this information, the attacker can also have another camera taking pictures of the scene at a different angle.

After videotaping the process, the attacker later reconstructs the scene (needing the same tablet model to do so). The attacker then experiments with how pressing different keys look and the respective optical flow that they tablet motion causes in video, eventually building a model to predict the keystroke from the tablet motion. This model is then applied to infer keystrokes from the video of the victim.

VISIBLE's average accuracy of predicting a single key is 36.2%, with the correct key being in the one-hop neighbor of the prediction 83.6% of the time. This indicates that the attack does an exceptional job of gaining regional information about keystrokes.

3.1.2 Countermeasure Proposal

A countermeasure for this attack would ideally introduce enough noise into tablet motion readings such that the resulting posterior distribution over the key pressed is uniform. Although the countermeasure may not be able to accomplish this precisely, the countermeasure needs to introduce enough noise such that it is computationally infeasible for the attacker to determine the keystrokes on the soft keyboard.

There are two ways we can accomplish this:

1. Randomize the positioning of letters on the soft keyboard. The attack relies on knowing the locations of

Figure 1. VISIBLE attack setup. Two video cameras are taking a video of the backside of the table as a user inputs keystrokes. Picture taken from [SJC+16a]

each letter beforehand, so this process would prevent the attack completely.

2. Introduce tablet vibrations/motions at every pressed key. These vibrations will add adversarial noise to models that attackers use to predict keystroke from motion.

3.1.3 Countermeasure Analysis

Countermeasure 1 is extremely effective in preventing this attack (it is also effective against most keystroke attacks), but it does degrade the user experience. Users will spend a lot of time typing their passwords since they will have to look at the keyboard and find each letter they press. Many users may not enjoy the product, and it is unclear whether this tradeoff of usability for security is worth it for many users.

Countermeasure 2 does not have as much of a usability problem. However, the amount of motion necessary to introduce significant noise in models will most likely hinder the user experience and make the tablet harder to hold. This amount of motion can be introduced into the tablet physically, as many methods of tablet feedback motion exist that can be harnessed to create motion.

3.2. Wifi-Based External Attack: WindTalker

3.2.1 Attack Description

There are several different wifi-based attacks that target passwords and other sensitive inputs on mobile devices, but many of them follow the same basic pipeline. As such, we will use WindTalker as a representative example of a wifi-based attack (workflow detailed in 2).

WindTalker is a wifi-based attack explored in [LML+16]. The premise of the attack begins with the attacker setting up a rogue WiFi hotspot, which the victim connects to obtain free WiFi. Once the victim connects, the adversary can log information known as channel state information (CSI) from the communication channel between the victim's device and the rogue hotspot. After connecting to the rogue hotspot, the victim's keystrokes can cause fluctuations in the CSI [ALWS15]. These fluctuations do occur as the victim unlock the screens of their mobile devices, and they become the basis for the adversary's keystroke inference attack.

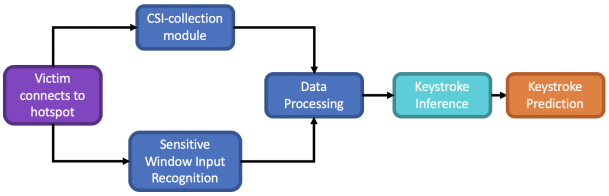

Figure 2. Attack pipeline for WindTaker, adapted from Figure 4 in [LML+16]. Note that other wifi-based attacks will follow a similar sequence of steps.

.

Once the adversary collects CSI data, he or she can pinpoint what points in the time series correspond to the user inputting sensitive input. Once these sensitive points in time have been identified, the adversary cleans the data through noise removal via a directional antenna on the hotspot source and then dimensionality reduction through low-pass filtering. Finally, the filtered data is passed through a machine-learning algorithm that determines possible keystrokes and ranks them by likelihood of being the victim's password.

Li and his team tested this attack on XiaoMi devices and Samsung devices [LML+16]. On XiaoMi devices, they achieved a 1-digit recovery rate of 79% while in Samsung devices, they observed a 64% recovery rate, both of which demonstrate how much of a threat the WindTalker attack can be.

3.2.2 Countermeasure Proposal

Wifi-based attacks seem to be based on running machine learning models on CSI data. It should follow that these

methods will perform poorly if we can remove the vulnerabilities being exploited by such attacks in the data.

According to [ALWS15], keystrokes lead to fluctuations that are carried in CSI values for all subcarriers between the transmitter and the receiver. In particular, the hand movements that produce the keystrokes lead to changes in existing multipath signals and to the creation of new multipath signals that constructively and destructively interfere with each other, thereby causing changes in the CSI values that are built off of them. These moments are detected by the sensitive input window detection portion of wifi-based attacks such as WindTalker and are crucial to the success of the attack. One way to cripple this module would be to induce the same kinds of CSI fluctuations during times of sensitive input during other times as well.

We can accomplish this by creating a protocol that periodically prompts users to enter password-like codes whenever they are connecting public wifi servers, or servers different from the ones they regularly use. In a given mobile device, every set amount of time, the user would be brought to a lock screen that displays a password of the same type that the user has chosen to lock his or her phone, which the user is prompted to copy. In doing so, the same fluctuations should be produced in the CSI that make the user vulnerable to a wifi-based attack in the first place. The user would be able to choose how often they wish to see such a prompt to deter potential attackers. This method should give several dummy windows for the sensitive input window detection modules to detect in a wifi-based attack, camouflaging the actual password input window. While the attack can infer keystrokes for all such windows, it should be difficult for the attacker to determine which set of inferred keystrokes is the victim's password.

3.2.3 Countermeasure Analysis

Our proposed countermeasure's biggest strengths are ease of implementation and lack of modification to CSI. Other works, such as [YCWY20] and [THCS14] offer protocols for securing CSI. However, in doing so, they modify the rules that mobile devices globally follow when connected to wifi. In particular, any mobile device that follows these "secure CSI" protocols end up securing CSI not just from attackers but also from parties who legally use it and need it. The countermeasure suggested above does not tamper CSI-data but simply fabricates the data and does not interfere with any services or processes that are supposed to use CSI.

An obvious drawback is inconvenience for users. They would have a scheduled interruption in whatever tasks they would be completing on their mobile devices, which makes our approach less appealing. However, the user would be given the choice of how often they have the dummy password prompts appear. Placing this choice in the hands of mobile device users will allow for them to choose a value that matches their perceived level of danger. We would, however, only offer a handful of choices that reflect our perceived level of risk that a user experiences when seeking out public wifi for his or her mobile device to connect to. This compromise should offset some frustration by giving users some choice over how they are being secured, should they choose to use our protocol.

It should be noted that our dummy password prompts appear to share some of the same paradigms as Captcha, a layer of security that deters bots from using online services [Raj17]. Since our dummy password prompts do require the password being displayed to be replicated by the user, they have the added bonus of hindering any unintended bot processes that may be running on the user's device as well.

3.3. PIN Skimmer and Spy Camera

3.3.1 Attack Description

There are two video based attacks that involve sensors that will be included in the video based attacks. The first of which is called PIN Skimmer. The idea behind PIN Skimmer is that it is a side-channel attack that utilizes a mobile phone's video camera and microphone to infer number-only soft keyboard PINs on a smartphone. Thus, this kind of attack is mainly used for phone lock screen passwords that use the traditional 10 digit keypad. The microphone is used to let the attacker know when the touch occurs and the camera is used to estimate the orientation of the smartphone. By using this orientation, the attacker is able to determine the position of the digit that is tapped [SA13]. The results of this attack showed that 30% of PINs were guessed after 2 attempts and 50% of PINs were guessed after 5 attempts.

The other video based attack that we will consider is Spy Camera; its workflow is detailed in 3. Spy Camera is a technique in which the attacker utilizes a mobile phone's camera to secretly take pictures and or record videos, which are then sent to the attack when WiFi is available. The way that the attack is able to hijack the mobile phone's camera is if the user of the phone downloads a seemingly harmless app that contains malicious code. This malicious code will be able to run background processes, enabling the attacker to hijack the phone's camera and take control of it. By using this "Spy Camera", the attacker is then able to use eye tracking technology to figure out phone pass codes and application passwords with ease [WD14].

3.3.2 Countermeasure Proposal

A similarity between these two attacks is that they both use some variant of technology to deduce what digit the user is inputting. For PIN Skimmer, the attacker utilizes the camera to guess the orientation of the phone, linking it to the digit in which the user presses onto the phone. For Spy

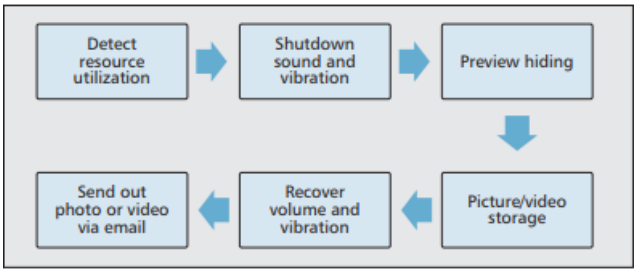

Figure 3. Spy Camera workflow. Based on Figure 1 from [WD14]

.

Camera, the attacker utilizes the camera to keep track of the user's eye movements. Since the numbers on the phone are static and are always in the same position, there can be machine learning techniques that deduce what position of the eye correspond to the digit pressed. Thus, the solution that we propose against these two attacks are a randomization of the numbers of the keypad.

3.3.3 Countermeasure Analysis

Randomizing the 10 digits on the keypad will be able to combat against these two attacks. Adding this randomization will make it hard for the attacker to link the eye position to the digit being pressed. This introduces a sizeable amount of noise which will help defend against this attack. However, a negative aspect to this solution is that this solution will inconvenience all users because every time the user has to input the code, the layout of the 10 digits will be different. Thus, this solution may frustrate the user and the user will have to spend a bit more time to input the code than usual.

3.4. EyeTell

3.4.1 Attack Description

EyeTell is a novel video-based keystroke inference attack developed by Chen et al [CLZ+18]. The attack infers a victim's keystrokes entered on a mobile soft keyboard by capturing the victim's eye movement and converting it to a continuous "gaze trace". The user's keystrokes can then be inferred, using different methods for different soft keyboard input types (PIN, pattern, and alphabetical). The full workflow can be seen in 3.4.1.

Experimental results from the group show that EyeTell can identify top-10 4-digit PINs with probability 74%, top-10 Android lock patterns with probability 75.3%, and top-10 likely words with probabilities with probability 63.19%.

Compared to similar keystroke inference attacks, EyeTell presents the greatest adversarial potential since it only requires a video of the victim's gaze trace. Other attacks such as [BCV08], [SJC+16a], [SA13], [CC11], etc require situational assumptions or various assisting tools: sensors, malware on the victim's device, analysis of tablet backside motion, and/or a direct video of the victim's fingers interacting with the soft keyboard. The relatively few requirements needed to execute an EyeTell attack and its high experimental efficacy for keystroke inference make the attack a serious threat to user privacy, and thus provides a particularly significant point of study for our proposed security policy.

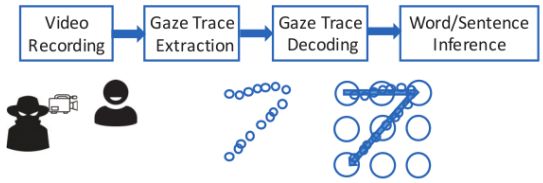

Figure 4. EyeTell workflow. Based on Figure 4 from [CLZ+18]

.

3.4.2 Countermeasure Proposal

EyeTell's greatest strength is its ability to infer keystrokes based on just a victim's gaze trace, without the need for any external tools or malware on the victim's device. As a result, our proposed countermeasure aims to interfere as minimally as possible with the actual lock screen password inputting process, while also diverting the victim's gaze trace enough to nullify the attack. To achieve this, we propose to have system-level software that, when a user begins to input their lock screen password, will randomly light up certain keys on the soft keyboard. The light source must be distracting enough to potentially divert a user's gaze trace, while also not being too much of a discomfort to the user experience. The distraction must also not prevent the user from reliably entering their password, and cannot occur so frequently that either users become annoyed or filter it out.

3.4.3 Countermeasure Analysis

Of course, further research and experimental trials would have to be conducted to iterate on an appropriate light distraction; however, if the user's gaze trace can be diverted even once during the process and if the light distraction occurs at random intervals, then this complicates and degrades the attack in the following ways (we assume that an adversary is aware of such countermeasures):

1. EyeTell's gaze extraction step itself will inherently have more noise, since the victim's gaze trace is no longer guaranteed to perfectly reflect that of their exact password. This increases the difficulty to obtain an accurate gaze trace to begin with.

2. We define a successful light distraction as an instance of the victim's gaze trace being diverted to a random part of the soft keyboard that is not the next keystroke in the lock password. If we let the number of successful light distractions be k, and the total length of the lock screen password be n, then even if we assume that the gaze extraction step is perfect, the total number of possible password candidates increases by a factor of $\binom{n}{k}$. This is because there are $\binom{n}{k}$ possible locations for the k "false" keystrokes. Furthermore, unless there is reason to believe that a specific inferred "keystroke" was merely a result of the light distraction, if the distraction happens to land on the keyboard input button that changes the keyboard to show symbols and numbers instead of letters, this can significantly increase the number of possible password entries. This would effectively create an entirely new branch of possible passwords, creating an exponential effect if $k > 1$. A similar argument can be made for a distraction lighting up the backspace button, which prevents the attacker from knowing when an error in the password inputting was truly an accident or if it was merely a "fake" deletion.

3. EyeTell also uses a dictionary for assistance with word inference. With random letters "added" to the gaze trace, strings of characters would no longer require dictionary lookups, but instead would require some form of fuzzy searching, which is much more complex and may not even account for words themselves being purposefully misspelled.

Despite the numerous potential benefits towards gaze trace disruption and increasing inference complexity, our proposed countermeasure will most likely encounter practical drawbacks. We outline these potential concerns here:

1. If users are distracted enough, they could forget where they were in entering their password, or how to finish their password altogether. This would make for an undesirable user experience, since most users would like to minimize time spent unlocking their phones.

2. The implementation for the light distractions may be complex, since not only should an appropriate light source be chosen, but also the soft keyboard inputs would have to support the light feature. This would require major updates to the phone's system-level software.

3. If the distractions are disruptive enough to the user experience, they may turn them off in their settings, rendering any implementation useless. This would leave users vulnerable again to EyeTell, with an additional dead-weight security feature on their phone.

4. Conclusion

4.1. Summary and Potential Impact

Mobile devices are becoming increasingly integrated into people's daily lives. Only a decade ago, the main features mobile devices offered were simply voice calling and text messaging, but modern-day phones can browse the web, host video call, run applications relying on sensitive personal information, and more. As mobile devices become more and more central to the lives of average consumers, attackers are also adapting and continuing to develop new techniques to attack mobile devices. In this paper, we study phone lock screen password systems, which are the most common lines of defense between attackers and the contents of a mobile device. We cover recent three kinds of recent keystroke-inference attacks − motion-based, wifi-based, and video-based − and offer potential countermeasures to deter such attacks.

Many of the measures we propose should be relatively simple to implement and to render specific portions of the various attack pipelines ineffective. They may inconvenience users by being more tedious or by interrupting their workflow on their mobile devices, but we believe that it is a small price to pay for protection against the types of attacks covered in this paper. Many of these features could be made so that the user can toggle them on and off. In doing so, users have the ability to avoid inconveniences when they are in environments where they can be more confident that they will be not at risk from the types of attacks covered in this paper, such as the comfort of their homes.

4.2. Further Discussion and Alternatives

We ultimately settled on the four keystroke inference attacks, detailed in Section 3, due to their different requirements and vectors of attack. We also came across similar attacks, such as using malware on a Smartwatch to obtain hand motion side channel information about the password inputting process, as described in [LZD+15]. We realized that this attack along with many others, such as [CC11] [BCV08], all fall under a similar category as PIN Skimmer or Spy Camera, detailed in 3, and they all benefit from the same proposed countermeasure. Other different attacks may also have their potency mitigated by any or a combination of the suggested countermeasures.

We also considered a few other countermeasure options for our four attacks:

1. A drop-down menu that contains randomized digits (for PIN attacks). This would have the benefit of forcing attackers to solve a much more difficult problem of assessing finger scrolling speed to judge which digit was selected.

2. A randomized grouping of characters in a new soft

keyboard format (for word inference attacks). This would introduce more complexity in character/symbol selection, which is easy for the user, but extremely difficult for an adversary since group placement would be randomized, so the number of possibilities increases exponentially.

3. A new soft keyboard format, where the screen would be populated with randomly selected characters and numbers, along with a predesignated target location on the screen. The user would then be asked to drag the entire image on the screen until the target character appeared in the target zone. This would render any gaze trace or "smudge attacks" [AGM+10] useless, since paths dragged on the screen are unique between attempts, and character positions are completely randomized.

However, we realized that the randomization from these ideas inherently degrades from the user experience and would perhaps never be practical. We eventually settled on multiple different ways to add noise to the inputting process in such a way that user experience is not sacrificed too heavily.

Another important note is that most modern mobile systems have an exponentially increasing lock timeout period after a certain number of incorrect password attempts. If we assume this to be true on the victim devices, then in combination with adding noise to lower the accuracy rate of the attacks' inference attempts, our countermeasures effectively exponentially increase the time required for attackers to steal a device's password. For all intents and purposes, no practical adversary has infinite computing power or time, and so this would further strengthen device lock screen security as a result of the proposed countermeasures.

4.3. Future Work

As for future work that can be done, a few ideas include developing a thorough security policy for phone lock screen systems. Because of its importance in the security of a phone and all of its contents, the lock screen should be secure to the utmost extent. However, with any system, there are always going to be loopholes. As a result, the countermeasures we explored should be considered, especially if attacks become potent or practical enough to become "mainstream" attacks. Additionally, because this paper proposes a few countermeasures, it is worthwhile to hold a survey of a subset of people that have our countermeasures installed on their phones and ask about their user experiences. In general, product manufacturers need to find a balance between security and user experience, rather than solely prioritizing security.

References

[AGM+10] Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith. Smudge attacks on smartphone touch screens. In *Proceedings of the 4th USENIX Conference on Offensive Technologies*, WOOT'10, page 1–7, USA, 2010. USENIX Association.

[ALWS15] Kamran Ali, Alex Xiao Liu, Wei Wang, and Muhammad Shahzad. Keystroke recognition using wifi signals. *ACM MobiCom*, 2015.

[AND20] Application sandbox. *Android Open Source Project*, Jan 2020.

[BCV08] D. Balzarotti, M. Cova, and G. Vigna. Clearshot: Eavesdropping on keyboard input from video. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 170–183, 2008.

[CC11] Liang Cai and Hao Chen. Touchlogger: inferring keystrokes on touch screen from smartphone motion. pages 9–9, 08 2011.

[CLZ+18] Yimin Chen, Tao Li, Rui Zhang, Yanchao Zhang, and Terri Hedgpeth. Eyetell: Video-assisted touchscreen keystroke inference from eye movements. *2018 IEEE Symposium on Security and Privacy (SP)*, 2018.

[GVS20] Chaitanya GVS. Mobile security: What to expect in the year 2020, Feb 2020.

[HLE16] Marian Harbach, Alexander De Luca, and Serge Egelman. The anatomy of smartphone unlocking: A field study of android lock screens. *2016 CHI Conference*, 2016.

[LML+16] Mengyuan Li, Yan Meng, Junyi Liu, Haojin Zhu, Xiaohui Liang, Yao Liu, and Na Ruan. When csi meets public wifi: Inferring your mobile phone password via wifi signals. *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[LZD+15] Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, and Kehuan Zhang. When good becomes evil: Keystroke inference with smartwatch. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1273–1285, New York, NY, USA, 2015. Association for Computing Machinery.

[MK16] Jyoti Malik and Rishabh Kaushal. Credroid: Android malware detection by network traffic analysis. In *Proceedings of the 1st ACM Workshop on Privacy-Aware Mobile Computing*, PAMCO '16, page 28–36, New York, NY, USA, 2016. Association for Computing Machinery.

[Raj17] Omid Rajaee. In-depth study of captcha, 04 2017.

[SA13] Laurent Simon and Ross Anderson. Pin skimmer: Inferring pins through the camera and microphone. pages 67–78, 11 2013.

[SJC+16a] Jingchao Sun, Xiaocong Jin, Yimin Chen, Jinxue Zhang, Rui Zhang, and Yanchao Zhang. Visible: Video-assisted keystroke inference from tablet backside motion. 01 2016.

[SJC+16b] Jingchao Sun, Xiaocong Jin, Yimin Chen, Jinxue Zhang, Yanchao Zhang, and Rui Zhang. Visible: Video-assisted keystroke inference from tablet backside motion. In *NDSS*, 2016.

[SSC+08] A. Schmidt, H. Schmidt, J. Clausen, Kamer Ali Yüksel, O. Kiraz, Seyit Camtepe, and Sahin Albayrak. Enhancing security of linux-based android devices. 01 2008.

[THCS14] Yu-Chih Tung, Sihui Han, Dongyao Chen, and Kang G. Shin. Vulnerability and protection of channel state information in multiuser mimo networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 775–786, New York, NY, USA, 2014. Association for Computing Machinery.

[WD14] Longfei Wu and Xiaojiang Du. Security threats to mobile multimedia applications: Camera-based attacks on mobile phones. pages 80–87, 03 2014.

[YCWY20] Y. Yang, Y. Chen, W. Wang, and G. Yang. Securing channel state information in multiuser mimo with limited feedback. *IEEE Transactions on Wireless Communications*, 19(5):3091–3103, 2020.