TinyLock: Affordable defense against smudge attacks on smartphone pattern lock systems

Taekyoung Kwon*, Sarang Na

Graduate School of Information, Yonsei University, Seoul 120-749, Republic of Korea

A R T I C L E   I N F O

Article history:
Received 10 August 2013
Received in revised form
13 November 2013
Accepted 9 December 2013

Keywords:
Human–computer interaction
Pattern lock
Smartphone
Smudge attack
Authentication

A B S T R A C T

A pattern lock system is a widely used graphical password mechanism in today’s mobile computing environment. To unlock a smartphone, a user draws a memorized graphical pattern with a finger on a flat touchscreen whereas the finger actually leaves its oily residues, also called smudges, on the surface of the touchscreen. The smudges can be exploited by adversaries to reproduce the secret pattern. Unfortunately, however, security is still dependent on a user’s behavior that is to carefully remove them after use. In this paper, we study an affordable defense to resist the smudge attacks without losing the ease-of-use property of the pattern lock system and without demanding user’s attentional behavior after use. We present TinyLock as our main result. TinyLock is a simple tweak of the user interface under the existing pattern lock paradigm but it can effectively resist the smudge attacks. Furthermore, TinyLock can be more resilient to shoulder-surfing attacks than the contemporary pattern lock systems. Our user study shows that TinyLock can significantly improve security of the pattern lock system while incurring minimal cost increase in terms of unlocking time.

1. Introduction

The recent proliferation of smartphones and tablet computers are changing the way we live our daily life, in particular, the way we interact with computing devices. Indeed, we frequently access our smartphones in public places with a “finger” on a flat touchscreen, to access various kinds of Internet services and applications (Rahmati and Zhong, 2013), such as e-mail, social networks, e-banking and e-shopping through public wireless networks; to store private information, such as passwords, schedules, photos, contact list and chat messages in its internal storage; and to pay for goods and to access other services as security tokens, for instance, NFC-based e-wallet and two-factor authentication. Therefore, we are greatly concerned about other people sharing our own smartphones (Karlson et al., 2009). To prevent any unwanted access, many users lock their smartphones when not in use. To unlock the smartphone, the users are required for authentication on its small touchscreen. The most widespread user authentication method offered by the latest smartphone is obviously password-based authentication including textual passwords, PINs, and graphical patterns. Such small secrets are easy to memorize and have historically been accepted by users due to their familiarity while convenience heavily depends on the type of keyboard and the makeup of the password (Schaub et al., 2012).

A pattern lock system, which is popularly known as Android pattern lock, is a graphical password authentication mechanism. The Android pattern lock incorporates a $3 \times 3$ grid of small dots as its basic user interface on the touchscreen. On the small grid, a user is required to draw a memorized graphical pattern with a finger. The streaks drawn by the user must be a one-way path connecting four or more dots on the
grid. Basically, that is a specific form of directed graph having nine vertices at maximum (four at minimum) and distinct starting and ending points. It is estimated that 389,112 patterns can possibly be drawn on the Android pattern lock system (Aviv et al., 2010). The patterns produce the larger password space than that of four-digit PINs but users are apt to choose easy-to-remember and/or easy-to-draw patterns (Andriotis et al., 2013). There are variants that arrange more dots on the grid but the $3 \times 3$ grid pattern lock is mostly used for its ease-of-use property.

Recently, Aviv et al. (2010) presented a smudge attack and subsequently Andriotis et al. (2013) enriched it on the smartphone pattern lock system: Oily residues remaining on the surface of the touchscreen can be exploited by adversaries to reproduce the user’s secret pattern, more easily with photographs taken by cameras and editing tools on commodities. Security of the current system is solely dependent on the user’s behavior to remove them intentionally or even unintentionally from the whole touchscreen. Unfortunately, however, it was reported that smudge attacks are effective not only in ideal cases where the touchscreen is clean but also in distortion cases where the touchscreen is wiped in part and stained more (Aviv et al., 2010). The pattern smudges are distorted only in part by the further use of apps or calls and the incidental contact with clothes or hands. The smudge attacks on the pattern lock systems have become a matter of great concern regarding privacy. Another concern about the pattern lock system is a shoulder-surfing attack (Tari et al., 2006; Long and Wiles, 2008). When a user draws his/her secret pattern, nearby adversaries can observe what the user is drawing. Although hiding the feedback streaks (as done in the Android pattern lock system) could mitigate this problem, the adversaries could still obtain the secret if the pattern was too simple. Unfortunately, many users choose simple patterns (Andriotis et al., 2013). We need more visual occlusion.

To deal with such a challenging problem in this paper, we present TinyLock, a novel defense against the above attacks. In TinyLock, we basically tweak the user interface design of the pattern lock system to be affordable and usable for the users who are not the enemy (Adams and Sasse, 1999). To be affordable, our defense does not require any particular hardware on smartphones but only runs in software. To be usable, our defense does not require a user to draw more complicated patterns nor to distort them with awareness. TinyLock entails three new design concepts: (1) a tiny grid which is an actual drawing pad smaller in its size, (2) confirmation dots that feedback a starting point selected by the user, and (3) a virtual wheel that appears after a correct pattern was drawn, actually to squash the smudges in a small spot without needing user’s attention. Users can draw their patterns on the tiny grid with a vibration feedback, by confirming the staring point on the confirmation dots if necessary, and then rotate the virtual wheel to finalize the unlock process. The users are actually enforced to confine and squash smudges in a small spot without their awareness while drawing patterns. They are also enforced to more occlude the streaks and to less move a finger on the grid than in the Android pattern lock system. We performed a user study and were able to confirm that TinyLock is more secure than the Android pattern lock system while incurring minimal cost increase in terms of unlock time.

The remainder of this paper is organized as follows: In Section 2, we briefly review the threat model and the related work. In Section 3, we introduce TinyLock and perform a brief analysis for usability and security. In Section 4, we describe our user study. In Section 5, we conclude this paper.

2. Preliminary

2.1. Threat model

In the pattern lock system, a user chooses his/her own grid pattern, which may be easy-to-use (Andriotis et al., 2013), and registers it on his/her smartphone system (briefly the system from now on). The system is locked when it is not in use whereas an adversary is assumed not capable of directly accessing the secret pattern stored in the user’s system. To unlock the system, the user must correctly draw the secret pattern with a finger on the touchscreen. Although the Android pattern lock system provides an option to hide the feedback of stretches in order to mitigate the shoulder-surfing attacks, the user’s finger is oily and so still leaves recognizable smudges on the touchscreen. Thus, the adversary is able to exploit the smudges remaining on the touchscreen so as to mount the smudge attacks when the system is after use. Readers are referred to our own experimental screenshots in Fig. 1 and to (Aviv et al., 2010) for more details of the experimental setups and the attacks.

As for the adversary, we make a strong assumption that the smudge attacker has full access to the touchscreen of the target system at which pattern lock authentication has occurred (Aviv et al., 2010). The smudge attacker can have a close visual look or take a photograph with a proper glancing angle at the surface of touchscreen, possibly in possession of the smartphone. The attacker is capable of controlling the lighting conditions and of taking a picture. She is even allowed to alter the touchscreen to increase the retrieval rate. As explained in Aviv et al. (2010) such a strong attacker can possibly exist in the real world applications, for instance, considering search and seizure procedures in many countries and states. A weaker assumption is also realistic that the smudge attacker can only take a picture by controlling the camera angle at a distance. For example, the smartphone which is locked after a brief use, can be put on a desk or just held by the user in one hand by revealing its touchscreen. It is very likely that the smudges remaining on the touchscreen can be exploited by adversaries to reproduce the secret pattern chosen (Andriotis et al., 2013) and drawn by the user.

2.2. Related work

The Android pattern lock is the most popularly used graphical password mechanism whereas there have been a great number of such methods that incorporate graphical password concepts. Let us review a few representative schemes first. Jermyn et al. introduced a graphical password scheme called “Draw-A-Secret (DAS)” (Jermyn et al., 1999). In DAS, the user draws a registered secret picture on a $5 \times 5$ grid for authentication. The system verifies the picture drawn on the grid according to the coordinates of the grid. Although the full
password space of DAS was greater than that of the textual password, many researchers studied that the actual password space of DAS was significantly narrowed by the predictable and poor password choice of the users (Nali and Thorpe, 2004; Thorpe and van Oorschot, 2004). The Android pattern lock can be seen as a tweaked version of DAS on the touchscreen. Dhamija and Perrig introduced a graphical password scheme called Deja Vu, based on the hash visualization technique to generate random images (Dhamija and Perrig, 2000). In Deja Vu, a user was authenticated by identifying a certain number of selected images from the given set of images. Weinshall and Kirkpatrick also studied a recognition-based graphical password scheme and claimed easier memorability through the longer-term user study (Weinshall and Kirkpatrick, 2004). Wiedenbeck et al. proposed a graphical password scheme called PassPoints (Wiedenbeck et al., 2005b; Wiedenbeck et al., 2005c; Wiedenbeck et al., 2005a). On an arbitrary image in PassPoints, the user was authenticated by clicking his/her chosen points (pixels with tolerance) in the right order.

Researchers have also studied the problems of graphical password schemes in terms of security, usability, and storage. User-chosen graphical passwords were predictable (van Oorschot and Thorpe, 2008) and/or tended to concentrate on several hotspots on the images (Dirik et al., 2007; Thorpe and van Oorschot, 2007; Bicakci et al., 2009). Furthermore, even random graphical passwords were susceptible to shoulder-surfing attacks (Man et al., 2003; Tari et al., 2006). Such a security problem unleashed a number of security improvement measures (Chiasson et al., 2012; Gao et al., 2009; Hoanca and Mock, 2006; Lin et al., 2007; Weinshall, 2006; Wiedenbeck et al., 2006). However, the excessively longer entry time and the heavier registration process were still complained by users regarding the usability (Suo et al., 2006, 2005). Even worse, some of the security measures were broken down (Golle and Wagner, 2007; Asghar et al., 2011). Graphical passwords mostly required much more storage space than textual and numerical passwords (Suo et al., 2005).

The Android pattern lock was remarkable in mitigating the usability and storage problems of graphical passwords by employing a $3 \times 3$ grid of dots but not really in security. Aviv et al. (2010) showed that the smudge attacks can reproduce secret patterns from the smudges remaining on the touchscreen. Andriotis et al. (2013) recently enriched the smudge attacks with exploiting the user’s behavior of selecting poor patterns. Meanwhile, there have been several attempts to incorporate analog sensing capabilities to distinguish the real user more effectively (Angulo and Wästlund, 2012; Conti et al., 2011; De Luca et al., 2012). In the most recent related work, Zezschwitz et al. studied a method to prevent the smudge attacks (von Zezschwitz et al., 2013). They proposed three graphical password schemes, one of which was the pattern lock system that rotated and placed the grid at random on the touchscreen. However, due to the randomness, users experienced a difficulty in locating the grid. Furthermore, the pattern smudges were still remaining only with different orientation and location on the touchscreen.

3. New pattern lock system

3.1. TinyLock concept

To resist the smudge attacks affordably, we tweak the user interface design of the pattern lock system in software. As illustrated in Fig. 2, TinyLock is composed of three following components on the touchscreen.

- Tiny grid: We reduce the size of the $3 \times 3$ grid being touched by the user and locate the tiny grid on the casual thumb position on the touchscreen. The main reason for replacing the regular-sized grid with the tiny grid is that it is easier to occlude the smudges by following the virtual wheel as described below. For authentication, the user can draw a secret pattern on the tiny grid rather than on the larger...
grid. On the tiny grid, the dots and the stretches drawn by the user will be highlighted or hidden in the same way as the Android pattern lock system. The tiny grid feedbacks a short (10 ms) vibration when passing through a new dot so that the user can recognize the new dot more easily. From now on, we mean 10 ms for the short vibration.

- **Confirmation dots**: We provide another upper grid for initial confirmation purpose only. To start drawing, the user must select a single dot on the tiny grid as a starting point. If it is the case, the relative dot on the upper grid will feedback the selection by flickering and with a short vibration. The confirmation dots can help the user confirm his/her selection of the initial dot without experiencing visual finger occlusion.

- **Virtual wheel**: When the user released his/her finger to finish drawing and the drawn pattern was correct, we replace the tiny grid with the virtual wheel on the touchscreen. To complete the unlock process, the user must simply rotate the virtual wheel clockwise or counterclockwise. That is, the user is enforced even without clear awareness to distort smudges left on the touchscreen during unlocking the smartphone.

**Fig. 2** - TinyLock. (a) Initial layout. (b) Selecting an initial dot to begin an unlock process – the confirmation dot feedbacks this selection. (c) Drawing a pattern on the tiny grid – the tiny grid feedbacks the stretches being drawn. (d) Rotating the virtual wheel to complete the unlock process – the virtual wheel enforces distortion of the smudges.

**Fig. 3** - Comparison of the smudges. (a) Simple pattern in Android pattern lock. (b) Simple pattern in TinyLock. (c) Complex pattern in Android pattern lock. (d) Complex pattern in TinyLock.
3.2. Prototype implementation

We believe that “usable security” functions should be considered at both the design and the implementation phases because the user interface of new mobile computing devices should have features different from the conventional ones. We implemented our prototype system on the Google Android 4.1 platform to run on Samsung-Google Galaxy Nexus smartphones (4.65”, 720 × 1280 pixels, 316 ppi) and Samsung Galaxy Tab (10.1”, 1280 × 800 pixels, 149 ppi) for further evaluation of our method.

As illustrated in Fig. 4-(a), we basically set the size of the confirmation dots and that of the tiny grid, respectively, 0.709 × 0.709 inch² (18 × 18 mm²). Note that the size of the Android pattern lock is 1.614 × 1.614 inch² (41 × 41 mm²). In Fig. 4-(a), the size of the square enclosing the tiny grid to limit the area of showing streaks is 0.984 × 0.984 inch² (25 × 25 mm²). The basic size of the tiny grid was decided by our pre-study which we call the prototype user study in the following subsection. We located the tiny grid under the middle of the 4.65” touchscreen, i.e., the place where a thumb can be naturally positioned in a single-hand grip. As illustrated in Fig. 5, we located TinyLock with slight position adjustment on a larger screen of tablets. Fig. 5-(a) and (b) illustrate TinyLock, respectively, positioned for right-handed and left-handed users. Both positions can be switched easily by tapping a wished position on the larger touchscreen.

Following the Android pattern lock system, we implemented two modes of drawing operations: The normal mode (Fig. 4-(b)) visually feedbacks the streaks drawn by the user on the tiny grid while the stealth mode (Fig. 4-(c)) hides them all to further resist shoulder-surfing attacks. In both modes, however, the system still feedbacks a starting point by flickering on the confirmation dots and also feedbacks with vibration when passing through new dots on drawing, in order to ease user’s recognition. Although the initial dot can be shoulder-surfed...
from the confirmation dots, the whole streaks drawn by the user can be more secured in the stealth mode. Note that it is another considerable choice in the stealth mode to even hide the visualization feedback from the confirmation dots on initial dot selection and instead to provide a vibrotactile feedback only when the user selected a correct initial dot (i.e., not any dot). However, in this case, there is a trade-off that the system can disclose the initial dot information whenever an attacker directly accesses the smartphone which was locked by the user.

Although the horizontally-centered location of the tiny grid (which is not much different from the normal grid) would be fair to both right-handed and left-handed users, i.e., symmetrically accessible with both hands, we also implemented a customization mode (as illustrated in Fig. 4-(d)) to set up the size and the location of the tiny grid at user’s convenience. The user can use one of the three predefined grids or freely change the size and the location of the grid, e.g., by long pressing the tiny grid or by manipulating the arrow keys on the customization mode. The virtual wheel will automatically change its size and location according to the customized tiny grid at setup. On the larger screen of tablets, the handedness may require asymmetric positions of the tiny grid. For instance, Fig. 5-(c) and (d) show that a left-handed user is unlocking the tablet in the normal mode of TinyLock.

We used our prototype implementation with several variations in size in our prototype user study (Section 3.3) and then with the fixed size and location of the tiny grid in our main user study (Section 4). We also but briefly examined the handedness and the screen size effect on usability (Section 3.4.3).

3.3. Prototype user study: speed and accuracy

According to the Fitts’s law (Fitts, 1992; MacKenzie, 1992), there is a significant trade-off between speed and accuracy, associated with pointing actions whereby targets that are smaller and/or further away require more time to acquire. Thus, it was required to determine very carefully the basic size of the tiny grid. To evaluate the feasibility of drawing on the tiny grid and to decide the basic size of the grid in our prototype system, we conducted a prototype user study before performing the main user study. We briefly summarize the results and discuss several implications before describing our main user study.

3.3.1. Design and procedure

We aimed at featuring a simple drawing task, which incurred pointing and dragging actions on the small area of the touchscreen. To measure the speed and accuracy of drawing, we set the tiny grid in three size, 0.433 × 0.433 inch² (11 × 11 mm², enclosing square 15 × 15 mm², small), 0.571 × 0.571 inch² (14.5 × 14.5 mm², enclosing square 20 × 20 mm², middle), and 0.709 × 0.709 inch² (18 × 18 mm², enclosing square 25 × 25 mm², large), respectively, followed by the virtual wheel fit to the size of the enclosing square. We conducted the user experiment in two steps. In the first step, the user task was to select nine dots in order (from top-left to bottom right) by following the audio cues whereas the independent variable was grid size (small, middle, large). In the second step, the user task was to draw two (simple, complex) patterns on the three tiny grids until successful authentication. Two patterns chosen for the prototype study were “Z” (simple) and the complex one (simple, complex) and grid size (small, middle, large). We informally analyzed the distortion results of the smudges in every combination. In both steps, we counterbalanced the order of the grid size conditions with the Latin square to reduce learning effects of the same task performed on the tiny grid of different size and we also instructed participants to have exercise trials before evaluation in every combination. The transaction time (from the first touch to the last release in the second step) and the results were logged for later analysis.

3.3.2. Participants and results

After the approval of the Ethical Review Board organized by the university’s Student Affairs Section and Research Support Division, we were able to recruit 18 participants (10 males, 8 females) with college education in the local university. To represent the general population of users, we try to balance their majors and ages. The participants had (corrected-to-) normal eyesight and were right-handed.1 Their average age was 28.3, ranging from 18 to 44, and the average experience of using smartphone (cell phone) was 2.9 (11.6) years. All the participants had pre-knowledge about the Android pattern lock system. We gave them a small gratuity for the user test.

In the first step, the evaluation factor was the accuracy of the first selection on every dot. Among the evaluation results (18 × 9 per grid size), error rates were (0.049, sd: 0.057) in the small size of the tiny grid (0.012, sd: 0.036) in the middle size of the tiny grid, and (0.0, sd: 0.0) in the large size of the tiny grid. A Repeated Measures-ANOVA test (1 × 3) suggested that there was a significant difference in the accuracy of selection among the different size of the tiny grid in our experiment (F(2,34) = 9.609, p < 0.001). The participants performed significantly more accurate on the larger grid.

In the second step, the evaluation factor was both accuracy and speed of drawing. As for speed, we only measured the final successful authentication time per combination. Among the evaluation results (18 × 2 per grid size), the fastest combination for average unlocking time was the large tiny grid with simple patterns (median: 2.104 s; mean: 2.186 s; sd: 0.392 s). It was followed by the middle tiny grid with simple patterns (median: 2.096 s; mean: 2.251 s; sd: 0.463 s), the small tiny grid with simple patterns (median: 2.198 s; mean: 2.439 s; sd: 0.677 s), the large tiny grid with complex patterns (median: 3.412 s; mean: 3.408 s; sd: 0.538 s), the middle tiny grid with complex patterns (median: 3.685 s; mean: 3.710 s; sd: 0.463 s), and the small tiny grid with complex patterns (median: 4.156 s; mean: 4.188 s; sd: 0.677 s). A 2 × 3 (pattern type × grid size) RM-ANOVA test showed that there was a significant main effect for grid size (F(1,44, 24.030) = 13.081, p < 0.001). However, there was also a significant main effect for pattern

---

1 When we recruited participants, we only did right-handed users because we thought that the left-right centered tiny grid should be fair to both hands. However, it would be necessary to incorporate left-handed users in the future studies, particularly on larger tablets as illustrated in Fig. 5. Readers are referred to Section 3.4.3 for more studies.
type \( F(1, 17) = 460.538, p < 0.001 \). The interaction effect between pattern type and grid size was also significant \( F(2, 34) = 5.410, p < 0.01 \) in our user experiment.

As for accuracy, there was no error in the larger tiny grid without regard to the pattern type. There was also no error in the middle tiny grid with simple pattern. However, there were errors in the middle tiny grid with complex pattern (0.056, sd: 0.236), the smaller tiny grid with simple pattern (0.056, sd: 0.236), and the smaller tiny grid with complex pattern (0.222, sd: 0.428). A \( 2 \times 3 \) (pattern type \times \) grid size) RM-ANOVA test showed that there was a significant main effect for grid size \( F(1.272, 21.629) = 4.103, p < 0.05 \). However, there was no significant main effect for pattern type \( F(1, 17) = 2.957, n.s. (p = 0.104) \). The interaction effect between pattern type and grid size was not significant \( F(1.252, 21.281) = 1.178, n.s. (p = 0.304) \) in our user experiment. Fig. 6 graphically illustrates the result.

Informally, our research team member (who did not know the pattern conditions) analyzed the smudges remaining on the touchscreen in the second step. Consequently, we were not able to identify patterns from the smudges remaining on the touchscreen in both conditions. This must be due to the virtual wheel concept illustrated in Fig. 7. According to the results above, we decided to fix the basic size of the tiny grid as \( 0.709 \times 0.709 \) inch\(^2\) (18 \times 18 mm\(^2\), enclosing square 25 \times 25 mm\(^2\)).

3.4. Analysis and discussion

We briefly discuss security and usability issues of TinyLock based on the lessons learned from our prototype user study, and defer the details of our main user study to Section 4. We also conducted more experiments for completing our discussion.

3.4.1. Tiny grid size: security and usability

The small size of the tiny grid can render several useful characteristics from security perspectives. First of all, a finger (usually a thumb) can occlude much more portion of the grid while drawing a pattern in TinyLock than in the Android pattern lock system. Furthermore, the finger can move less over the tiny grid. It is also easier to cover and protect the drawing area with the other hand. These characteristics, saying, more finger occlusion, less finger move, and easier hand protection on the tiny grid, could make the shoulder-surfing attacks more difficult in TinyLock. Although TinyLock was constructed to share the similar property of the existing pattern lock against the shoulder-surfing attacks, we were able to observe such behavioral phenomena in our prototype user study. Finally, the size of smudges could also become smaller, so that the small virtual wheel over the tiny grid can effectively distort the smudges. The user can simply rotate the virtual wheel to finalize the unlock process whereas the rotation with finger can totally distort the smudges as illustrated in Fig. 7.

The small size of the tiny grid also renders an issue from usability perspectives. Compared to the larger grid of the Android pattern lock system, the tiny grid of our method provides the narrower space to draw patterns. For instance, on the 4.65" Galaxy Nexus smartphone, TinyLock basically provides a \( 3 \times 3 \) grid of tiny dots in \( 0.709 \times 0.709 \) inch\(^2\) whereas the Android pattern lock does a larger grid in \( 1.614 \times 1.614 \) inch\(^2\). The larger grid must be more convenient to draw patterns but the far distance among dots can increase the time to traverse the grid. Furthermore, it will take again more time to erase or distort the remaining smudges on the larger grid. On the other hand, as we already observed in our prototype user study, the tiny grid is sufficiently convenient and fast to draw even more complex patterns and to distort them on the touchscreen. The feedback of the confirmation dots can help users confirm the selection of a starting point even if they experience visual occlusion over the tiny grid.

![Fig. 6 – Results of the prototype study – speed and accuracy of drawing.](image-url)
After the right selection of the starting point, users can draw patterns on the tiny grid with ease help of the short vibration and streak feedbacks on the tiny grid. We will discuss more details in the following Section.

3.4.2. Pattern space and smudge attack: security
In TinyLock, although we reduced the size of the grid being touched, we kept the same number of dots as that of the Android pattern lock system on the grid. We also preserved the pattern construction policy: A pattern should be constructed with 4 dots at minimum and 9 dots at maximum with distinct starting and ending points. Thus, the theoretical password space of TinyLock is the same as that of the Android pattern lock system (Aviv et al., 2010). The total number of possible patterns is “389,112” in TinyLock.

Let $\mathcal{P}$ denote the set of such possible patterns. When launching the smudge attack, the attacker may try to reconstruct the secret pattern, $p_\mathcal{S}$, from the smudge, $s$, left on the touchscreen. Let us define that $s = p$ if $s$ is the right smudge of the pattern $p$. Although the attack can be done on the existing pattern lock system (Aviv et al., 2010) because $s = p$, it may not be the case on TinyLock because $s = w(p)$ for a one-way function $w(\cdot)$. We can regard $w(\cdot)$ as a one-way function because the wheel smudges actually squash the pattern smudges in a small spot on the touchscreen. However, if the attacker has a list of the formal templates, $t_s = w(p_s)$, for all or a part of patterns, $p_s \in \mathcal{P}$, she can compare $s$ to the list and try to find $p_s$ such that $t_s = s$ where $s$ means that $t_s$ is the nearest to $s$ in the list of templates. With regard to this, it should be a concern that $p_s$ is equal to $p$ or not with high probability.

Apart from the security evaluation based on our user study and so user’s behavior in Section 4, we analyze the security of TinyLock against the smudge attack as we discussed above. For the purpose, we selected ten patterns and reconstructed corresponding TinyLock smudges on the clean touchscreen. We took photos of ten smudges and constructed a list of ten smudge templates. We then simulated users and again constructed TinyLock smudges of the ten patterns. To simulate the worst case of security, we took photos of the user smudges at the same light condition and video resolution on the clean touchscreen. Finally, we used a nearest duplicate image detector called DupDetector and compared the user smudges to the list of formal templates. Note that every user smudge has its own template in the list. Fig. 8 graphically illustrates the results. Interestingly, the smudge templates were not nearest to the corresponding smudges for the patterns. They were rather randomly distributed as shown in Fig. 8. Although the set of patterns was small, we believe the result is convincing when we consider the simulated high video quality of the templates and the smudges in our experiment. It is convincing that the smudge attack can be discouraged in TinyLock.

3.4.3. Handedness and screen size: usability
Apart from the main and the prototype user studies, we additionally conducted an informal study to briefly examine whether the handedness and the screen size would affect the unlocking time performance of TinyLock. For the purpose, we designed a between-group study with two independent variables, handedness (left-handed, right-handed) and screen size (smartphone, tablet). With this factorial between-group design, we recruited 24 participants (15 males, 9 females; 12 left-handed; average age 26.7; average smartphone experience 2.4 years; pattern lock knowledge – yes) in the local university to organize 2 x 2 groups of 6 participants (i.e., 2 left-handed and 2 right-handed groups) and had each group complete the unlocking task under one of the four conditions (based on the handedness) with TinyLock. We gave them a small gratitude for the user test.

Before starting the evaluation phase, we demonstrated TinyLock with simple pattern “Z” and had every participant try it shortly. To measure the unlocking time in the evaluation phase, we arranged the complex pattern used in our prototype study and asked each participant to complete the unlocking task after five practice trials of it. The participants were required to use both hands only in tablet combinations. Every participant succeeded in the evaluation phase. We analyzed the results using the Univariate analysis (factorial ANOVA analysis) in SPSS. The fastest combination was right-handed smartphone (median: 2.885 s; mean: 2.815 s; sd: 0.299 s) and followed by right-handed tablet (median: 2.771 s; mean: 2.825 s; sd: 0.275 s), left-handed smartphone (median: 2.925 s; mean: 2.912 s; sd: 0.235 s), left-handed tablet (median: 2.944 s; mean: 2.931 s; sd: 0.378 s). The factorial ANOVA test suggested that there was no significant difference between participants who completed the smartphone unlocking tasks and those who completed tablet unlocking tasks ($F(1,20) = 0.014, n.s.$ ($p = 0.906$)). There was no significant difference between left-handed participants and those who
right-handed in completing unlocking tasks ($F(1,20) = 0.689$, n.s. ($p = 0.416$)).

4. Main user study: usability and security

We conducted the evaluation of TinyLock in comparison with Android pattern lock system. For evaluation, we implemented both systems on the Google Android 4.1 platform. The TinyLock implementation was the same prototype used in our prototype user study with fixing the basic size of the tiny grid as 0.709 \times 0.709 \text{ inch}^2 (18 \times 18 \text{ mm}^2), enclosing square 25 \times 25 \text{ mm}^2 on the 4.65\textdegree touchscreen of 720 \times 1280 pixels (316 ppi) in Galaxy Nexus smartphones. The whole interaction was logged for later analysis with regard to the drawing time (from the first touch to the last release) and authentication result. Sony HDR-PJ580 video camcorder was used to film the smudges remaining on the surface of the touchscreen and was prepared in clear and high resolution. The materials of the attack photos were actually the smudges remaining on the surface of the touchscreen in our usability study. Namely, the user task was to guess four (2 \times 2) patterns of the other users. For the purpose, we numbered $n$ participants in the user study. Participant $i$ was given the attack photos of the participant ($i \mod n) + 1$’s Android pattern lock combinations and the participant ($i + 1 \mod n) + 1$’s TinyLock combinations for $1 \leq i \leq n$. As a result, to each participant, different values were assigned for pattern type conditions to avoid learning effects. Instead, we fixed the order of conditions as given in the 2 \times 2 combination. The maximum number of guesses was five in every combination, as restricted in the real-world Android pattern lock system. This strategy was to simulate the real-world attacks: The adversaries who mounted the smudge attacks can further use their guesses to perpetrate active attacks on the locked target smartphone within five attempts.

After finishing our user study, we performed an informal analysis on videos and photos taken in the user study. Our research team members who were very familiar with both pattern lock systems but did not know the used patterns actually simulated attackers in the further analysis study.

4.2. Participants

We recruited participants with college education in the local university. Before recruitment, we explained our user study step by step and ethical treatment of participants to the Ethical Review Board organized by the university’s Student Affairs Section and Research Support Division. After the official approval of the Ethical Review Board, we were able to recruit 24 participants (15 males, 9 females) in the local university. In order to represent the general population of users, we try to balance their majors and ages. The participants had (corrected-to-)normal eyesight and were right-handed. Their average age was 29.5, ranging from 22 to 49, and the average experience of using Android smartphones (cell phones) was 2.8 (12.1) years. 22 participants had pre-knowledge of the pattern lock system and experience of using it whereas 15 had been actually using it. We gave them a small gratuity for the user test.

4.3. Procedure

At an instructional meeting, which we call orientation, we explained our user study step by step and demonstrated both Android pattern lock and TinyLock systems. We then started the user study with a questionnaire collecting demographic information about participants. The schedule was pipelined for the usability evaluation experiments of consecutive three days and for the security evaluation experiments of discrete one day, per participant.

4.3.1. Evaluation procedure for usability

In Day 1, we instructed each participant to perform a tutorial session for respective pattern lock systems. The tutorial
session was that the participant drew the shape of “Z” for three times in respective pattern lock systems. We then asked each participant to choose an easy-to-use pattern after mentioning that 3 or less streaks and 4 or 5 dots were desirable. We ran our system to produce more complex patterns with 7 or more dots and 5 or more streaks. We announced that we would take photos for the smudges left in the evaluation trial. Given the random combination of independent variables, the participants were asked to have a short training time until feeling acquaintance (mostly three authentication attempts) and to show us their practice entry with successful authentication. Finally, we asked the participants to enter their patterns for evaluation after cleaning the surface of the touchscreen. The evaluation task was to unlock the smartphone using the prototype system. For each combination, the participants were allowed five evaluation trials at maximum (after cleaning) if they failed in authentication (as like the contemporary pattern lock systems) but such a maximum case never happened. After finishing one combination, we took photos of the smudges of successful authentication and asked the participants to answer simple questions about the system. Finally, we asked them to fill out a post-test questionnaire for comparisons. The Likert-type scale was used for ratings from 1 (strongly disagree) to 5 (strongly agree). The unlock time and authentication results were logged for later analysis.

In Day 2, we asked each participant to enter their patterns that were used in Day 1 experiments for the combinations of TinyLock only. Before each evaluation trial, we showed their patterns and gave sufficient time to recall them (to simulate the most of everyday usage) but disallowed any practice trials to see if they actually remembered how to use our system, not the patterns themselves. This strategy was to simulate the user’s remembrance of the patterns and the frequent access to the smartphones. The surface of the touchscreen was cleaned before every evaluation trial. The smudges remaining after the successful authentication was taken by our camera. The unlock time and the authentication results were also logged for later analysis.

In Day 3, we asked each participant for the same user task as Day 2 except that we also asked to repeat the unlock process in the stealth mode for the combinations of TinyLock.

4.3.2. Evaluation procedure for security
After finishing the usability study, we asked each participant to guess the other users’ secret patterns by looking at the clearly taken photos. For the purpose, we used the attack photos taken in the usability study. Day 1 photos were mainly used and partly back-patched by Day 2 and Day 3 photos having better quality. Although we had mentioned at orientation, we explained the participants briefly again what the smudge attack is and how it can be done effectively. Given the combinations, we asked the participants to guess the target pattern at most five times. To help their analysis, we gave them the attack photos in various tune as illustrated in Fig. 9. We used Adobe Photoshop 13 (CS 6) for the tune. We also instructed that the participants could zoom in and zoom out repeatedly at their convenience. After finishing the experiment, they were asked to fill out a post-test questionnaire for comparisons. The Likert-type scale was used for ratings from 1 (strongly disagree) to 5 (strongly agree).

Finally, after finishing our user study for evaluating both security and usability, we also performed an informal analysis on the photos taken in the user study. We asked our research team member who was familiar with both pattern lock systems but did not participate in the user study, to simulate the attacker. We asked him to use Adobe Photoshop 13 (CS 6) or any other software tools for conducting more sophisticated analysis at his convenience. He reported the result after one week.

4.4. Hypotheses
We state the following main hypotheses:

- (H1) TinyLock is slower than Android pattern lock.
- (H2) TinyLock is more error-prone than Android pattern lock.
- (H3) TinyLock is more secure than Android pattern lock against smudge attacks.

Furthermore, we state the following additional hypotheses:

- (H4) TinyLock becomes slower in the longer-term use.
- (H5) TinyLock becomes more error-prone in the longer-term use.
- (H6) TinyLock is slower in the stealth mode.
- (H7) TinyLock is more error-prone in the stealth mode.
4.5. Results

4.5.1. Entry time

Entry time was measured for each successful unlocking task from the first press (to select a single dot as a starting point) to the last release (to finish drawing a pattern) on the touchscreen. Thus, a re-selection of the starting point was also incorporated. This strategy was to measure the actual interaction time including a possible re-selection as authentication speed. Fig. 10 illustrates the graphical results of the entry time (of successful authentication) for the Android pattern lock (briefly named as Android lock) and the TinyLock methods in combination with system-chosen (complex) and user-chosen (simple) patterns. Table 1 summarizes the numerical results.

The fastest combination was Android pattern lock with user-chosen patterns (median: 1.336 s; mean: 1.431 s; sd: 0.286 s). TinyLock with user-chosen patterns (median: 2.055 s; mean: 2.106 s; sd: 0.379 s), Android pattern lock with system-chosen pattern (median: 2.313 s; mean: 2.392 s; sd: 0.420 s), and TinyLock with system-chosen pattern (median: 3.155 s; mean: 3.191 s; sd: 0.513 s) followed it.

A 2 × 2 (pattern type × pattern lock system) RM-ANOVA test showed a highly significant main effect for pattern lock system ($F(1,23) = 148.151$, $p < 0.001$) and also for pattern type ($F(1,23) = 217.447$, $p < 0.001$). However, there was no significant interaction effect ($F(1,23) = 1.142$, n.s. ($p = 0.296$)) in the user experiment.

Regarding the questionnaire, the participants interestingly rated that the Android pattern lock system (mean: 3.88; sd: 1.076) was very slightly faster than TinyLock (mean: 3.83; sd: 0.963). We informally think that this result was due to the smaller size of the tiny grid and the subsequent use of the virtual wheel. Fig. 11 illustrates the main results of the questionnaires.

With respect to the above results, hypothesis H1 can be accepted.

4.5.2. Error rate

We measured whether the participant could unlock the smartphone with respective pattern lock systems within five attempts for evaluation. In both pattern lock systems, only one participant (different in the respective systems) succeeded in the second unlock attempt whereas the other remaining participants succeeded in the first attempt in the evaluation phase. Informally, we observed that visual occlusion incurred by a thumb on the grid was the possible main reason for the respective failures. Consequently, there was no one who could not unlock the smartphone within five attempts (actually two attempts) for both pattern lock systems in the evaluation phase. There was no significant difference between the different pattern types and between the different pattern lock systems.

Regarding the questionnaire, the participants interestingly rated that TinyLock (mean: 4.17; sd: 0.868) was slightly easier to use than the Android pattern lock system (mean: 4.00; sd: 0.834). We informally think that this result was due to the smaller size of the tiny grid which was favorable to the use of one hand as also illustrated in Fig. 11.

With respect to the above results, there was a failure to support hypothesis H2.

4.5.3. Security against smudge attacks

Security was evaluated in two ways. In the user study, after finishing the usability evaluation, the participants were asked to simulate the attacker in the security evaluation. Aside from the user study, one research team member was informally asked to simulate the attacker during the longer period.

In the user study, each participant analyzed the attack photos of four combinations and submitted five guesses per combination. The result was remarkable. Among 48 combinations with Android pattern lock, 48 patterns (100%) were guessed correctly in five attempts. However, among 48 combinations with TinyLock, no pattern (0%) was guessed at all in five attempts. There was a significant effect for pattern lock system.

In the informal study, the attacker analyzed the attack photos of 240 combinations that also involved 96

![Fig. 10](image_url) – Entry time of successful authentication for pattern lock and TinyLock with user-chosen (left) and system-chosen (right) patterns.

<table>
<thead>
<tr>
<th>Pattern lock methods</th>
<th>Median</th>
<th>Mean</th>
<th>Min</th>
<th>Max</th>
<th>Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android lock with simple pattern</td>
<td>1.336 s</td>
<td>1.431 s</td>
<td>0.941 s</td>
<td>1.964 s</td>
<td>0.286 s</td>
</tr>
<tr>
<td>TinyLock with simple pattern</td>
<td>2.055 s</td>
<td>2.106 s</td>
<td>1.525 s</td>
<td>2.839 s</td>
<td>0.379 s</td>
</tr>
<tr>
<td>Android lock with complex pattern</td>
<td>2.313 s</td>
<td>2.392 s</td>
<td>1.781 s</td>
<td>3.112 s</td>
<td>0.420 s</td>
</tr>
<tr>
<td>TinyLock with complex pattern</td>
<td>3.155 s</td>
<td>3.191 s</td>
<td>2.104 s</td>
<td>4.124 s</td>
<td>0.513 s</td>
</tr>
<tr>
<td>TinyLock with complex (Day 2)</td>
<td>3.276 s</td>
<td>3.344 s</td>
<td>2.661 s</td>
<td>4.268 s</td>
<td>0.409 s</td>
</tr>
<tr>
<td>TinyLock with complex (Day 3)</td>
<td>3.248 s</td>
<td>3.215 s</td>
<td>2.456 s</td>
<td>3.950 s</td>
<td>0.429 s</td>
</tr>
<tr>
<td>TinyLock with complex (stealth)</td>
<td>3.241 s</td>
<td>3.319 s</td>
<td>2.392 s</td>
<td>4.290 s</td>
<td>0.482 s</td>
</tr>
</tbody>
</table>

Note: s denotes time in second. Std stands for standard deviation.
combinations of the user study. The result was also remarkable. Among 48 combinations with Android pattern lock, 48 patterns (100%) were guessed correctly, 24 patterns with system-chosen patterns and 24 patterns with user-chosen patterns. However, among 192 combinations with TinyLock, no pattern (0%) was guessed at all.

Regarding the questionnaire, the participants of the user study also considered the Android pattern lock system (mean: 1.46; sd: 0.658) less secure than TinyLock (mean: 4.83; sd: 0.381). Fig. 11 illustrates the main results of the questionnaires.

With respect to these results, hypothesis H3 can be accepted.

4.5.4. Longer-term result and stealth mode

In the longer-term study of Day 2 and Day 3, TinyLock was evaluated for the same pattern combinations. We refer to Fig. 10 and Table 1 for the results: Day 2 (median: 3.276 s; mean: 3.344 s; sd: 0.429 s) for complex patterns. We also obtained Day 2 (median: 2.120 s; mean: 2.148 s; sd: 0.236 s) and Day 3 (median: 2.190 s; mean: 2.182 s; sd: 0.258 s) results for simple patterns. A (2 × 3) RM-ANOVA test suggested that there was no significant main effect in the entry time for Day 1, Day 2, and Day 3 uses of the same patterns in TinyLock (F(2,46) = 0.948, n.s. (p = 0.395)) and also no significant interaction effect (F(2,46) = 0.907, n.s. (p = 0.411)) in the user experiment. However, there was a significant main effect for pattern type (F(1,23) = 419.328, p < 0.001). With respect to this result, there was a failure to support hypothesis H4.

In Day 2, one participants with a system-chosen pattern unlocked the smartphone at the second attempts while the remaining participants did it at the first attempts in the evaluation phase. In Day 3, all the participants successfully unlocked the smartphone at the first attempt. In both experiments, there was no one who could not unlock the smartphone within five attempts (actually two attempts). There was no significant difference between the different pattern types and between the different modes. With respect to this result, there was a failure to support hypothesis H5.

In Day 3, we also asked the participants to enter their patterns in the stealth mode of TinyLock. We refer to Fig. 10 and Table 1 for the result (median: 3.241 s; mean: 3.319 s; sd: 0.482 s). A paired-samples t-test suggested that there was no significant difference in the entry time of complex patterns between normal mode and stealth mode in Day 3 in TinyLock (t(23) = –1.066, n.s. (p = 0.297)). With respect to this result, there was a failure to support hypothesis H6.

Also, one participant with a system-chosen pattern unlocked the smartphone at the second attempts while the remaining participants did it at the first attempts in the stealth mode. There was no one who could not unlock the smartphone within five attempts (actually two attempts). There was no significant difference between the different pattern types and between the different modes. With respect to this result, there was a failure to support hypothesis H7.

5. Conclusion

5.1. Summary

In this paper, we proposed the novel pattern lock mechanism called TinyLock which was evaluated both secure and usable in a practical sense for smartphone users. TinyLock was resilient to the smudge attacks with incurring minimal cost of entry time in our user study. The main component of TinyLock was (1) the tiny grid which reacts with streaks and vibration as a drawing pad, (2) the confirmation dots which reacts for an initial dot selection, and (3) the virtual wheel which removes the identifiable smudges very effectively. The size of the tiny grid followed by the virtual wheel was customizable. We believe that TinyLock can be applied to the contemporary smartphones for the general population of the smartphone users.

5.2. Limitations and future study

One can think that the small size of the tiny grid can be a drawback to a user who has a big thumb. However, as we
discussed in Section 3.4 and in Section 3.2, the small size can provide a benefit for both security and usability whereas the size and the location of the tiny grid can be easily customized. Although TinyLock was effective against the smudge attacks, it still shares the weakness of the pattern lock system against the shoulder-surfing attacks even in the stealth mode. For example, if an adversary uses a camera at right angle for her shoulder-surfing attacks as she did in the smudge attacks, the pattern can be disclosed with high probability. Nevertheless, the intrinsic characteristics of TinyLock, such as more finger occlusion, less finger movement, and easier hand protection on the tiny grid, could make the shoulder-surfing attacks more difficult on TinyLock than on the existing pattern lock systems.

In the future study, it would be promising to investigate alternative or enhanced authentication methods for other smart devices incorporating the touchscreen, based on the lessons learned from TinyLock.

Acknowledgment

This work was partly supported by the IT R&D program of MSIP/KEIT [10039180, Intuitive, convenient and secure HCI-based usable security technologies for mobile authentication and security enhancement in mobile computing environments]. This work was also supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2012R1A1B3000965).

REFERENCES

Taekyoung Kwon received the BS, MS, and PhD degrees in computer science from Yonsei University, Seoul, Korea, in 1992, 1995, and 1999, respectively. He is now an associate professor of information at Yonsei University, Seoul, Korea. From 1999 to 2000, he was a postdoctoral research fellow at the University of California, Berkeley. From 2001 to 2013, he was a professor of computer engineering at Sejong University, Seoul, Korea. From 2007 to 2008, he visited the University of Maryland, College Park, for sabbatical. His research interests include information security and privacy, applied cryptography, cryptographic protocol, usable security, and human-computer interactions.

Sarang Na received the BS and MS degrees in Computer Science and Engineering from Sejong University, Seoul, Korea, in 2011 and 2013, respectively. Currently, she is a PhD student at Graduate School of Information, Yonsei University, Seoul, Korea. Her research interests include cryptographic protocols, computer network security, mobile security and HCI.