Enhanced Knock Code Authentication with High Security and Improved Convenience

Yun-Hwan Jang[1] and Yongsu Park[2]
[1] Department of Information Security, Hanyang University
Wangshimriro 222, Seongdongku, Seoul 04763 – South Korea
[e-mail: dbsghksdlwkd@naver.com]
[2] Department of Computer Science, Hanyang University
Wangshimriro 222, Seongdongku, Seoul 04763 – South Korea
[e-mail: yongsu@hanyang.ac.kr]
*Corresponding author: Yongsu Park

Received February 12, 2018; revised March 18, 2018; accepted March 26, 2018; published September 30, 2018

Abstract

Since smartphone contains various personal data, security is one of the important aspects in smartphone technologies. Up to now, various authentication techniques have been proposed to protect smartphones. The pattern lock on the Android system is one of the most widely used authentication methods for low-cost devices but it is known to be vulnerable to smudge attack or shoulder surfing attack. LG's smartphone uses its own technique, which is called "Knock Code." The knock code completes the authentication by touching the user defined area in turn on the screen. In this paper, we propose the new, enhanced version of knock code by adding the sliding operation and by using flexible area recognition. We conducted security analysis, which shows that under the same password size, the search space is overwhelmingly larger than the original algorithm. Also, by using the sliding operation, the proposed scheme shows resilience against smudge attacks. We implemented the prototype of our scheme. Experimental results show that compared with the original Knock Code and Android pattern lock, our scheme is more convenient while providing better security.

Keywords: Authentication, graphical password, knock code, password security, usable security.

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (2017R1D1A1B03029550).

http://doi.org/10.3837/tiis.2018.09.024 **ISSN : 1976-7277**

1. Introduction

As smartphones are being widely used, the role of smartphones is growing even further in our life. Now, smartphone has variety of personal information such as contact address, location history, and credit card information, etc. Because it is closely related to daily life, security is considered as one of the most important aspects for smartphone technologies.

Smartphones have various authentication methods to verify legitimate users while minimizing inconvenience during authentication, e.g., pattern-lock, face recognition, fingerprint/retina recognition, etc. Among them, pattern-lock is still considered as one of the most convenient authentication in the Android system, especially for low-cost devices [1][2]. However, it is known that this technique is vulnerable to shoulder surfing [3][4], which is the attack by observing other person's authentication procedure over his/her shoulder to get the password. Also, it is vulnerable to smudge attack [5], which is to guess the password pattern by analyzing oily smears left on the device.

Knock Code™ [6] is the touchscreen-based authentication method, which was devised by LG Electronics. Recently, LG's share has reached 20% in the US smartphone market in the first quarter of 2017 [7]. Currently, LG Electronics is advertising Knock Code as one of most innovative features in LG's smartphone/smart pad devices. It has a strong point in that we can unlock the device while the screen is off. The off screen is divided into 4 areas: left top, left bottom, right top and bottom right. For authentication, the user should touch each area 6 to 8 times in the right order. Since this scheme can be run while the screen is off, the shoulder surfing attack is more difficult than the Android pattern lock. Also, LG points out its convenience because the user need not look at the screen for tapping. Nonetheless, Knock Code provides a level of security far greater than other devices, even those with fingerprint recognition systems [8].

In this paper, we present the enhanced version of Knock Code, where security is significantly higher than the original scheme while providing more convenience. Our scheme provides flexible area recognition, which enables that the user puts the password on the screen at any position/angle. Moreover, our scheme supports the sliding operation, which provides security and convenience together. We provide mathematical analysis on security of our scheme. Also, we provides the user study on security/convenience of our scheme.

This paper is organized as follows. In Section 2, we explain major previous graphical authentication techniques. Section 3 describes the proposed scheme. In Section 4 we analyze security and convenience of the proposed method and we conclude the paper in Section 5.

2. Related Work

Graphical password schemes can be broadly classified into 3 types: recall, recognition and cued-recall [1]. In Recall-based schemes, users first draw the password on blank or grid canvas. Later, they should recall and reproduce the secret drawing for authentication. In recognition-based systems, users first memorize a portfolio of images and then recognize their images for authentication. In cued-recall systems, users should remember specific locations within an image. Later, they point out the locations for authentication.

Roughly speaking, Knock Code corresponds to the recall-based schemes among them. In this section, we will explain various major recall-based schemes that have been studied.

DAS (Draw A Secret): DAS is one of the oldest graphic password techniques, which was designed in 1999. Also, it is considered as one of the most influential techniques for the subsequent graphic password research [9]. DAS authentication is done in a two-dimensional grid. First, the user can draw a line or make a point on the grid when setting the password. In login, if he/she draws in the same paths/points as you set it, authentication is successful. Security of DAS depends on the grid size and the password length. For 5x5 grid and the maximum password length of 12, theoretical password space is 2^{58} [9]. Numerous variants have been proposed, e.g. BDAS [10], Pass-Go [11], YAGP [12], Passdoodle [13], PassShapes [14], Android pattern lock, GrIDsure [15] etc.

TMD [16]: In 2013, Hsin-Yi Chiang et al. proposed Touchscreen Multi-layered Drawing (TMD) [8], a new pattern locking technique, which is more robust against shoulder surfing attacks than the Android pattern lock. TMD uses a 5x7 grid and supports multi-layer authentication, compared with the existing Android pattern lock. The 5x7 grid consists of the selected cell, the unselected cell and the warp cell. Initially, the upper left, upper right, lower left, and lower right are always warp cells while the remaining 31 cells (except the warp cells) are set as unselected cells. The user selects one unselected cell as a starting point. Then, he/she selects one adjacent unselected cell and it becomes the selected cell. When he/she arrives at the warp cell, registration finishes. If the multi-layer is supported, all selected cells become unselected cells and he/she repeat the selection procedure from the warp cell for the next layer. i.e., he/she draws a new pattern for the next layer. In login, if he/she draws the correct pattern for all layers, the authentication is successful.

When the number of layers is D, the lower bounds of the number of passwords that TMD can create are $2^{62} * D$ [16], which is much larger than the existing Android pattern lock.

TinyLock [17]: In 2014, Kwon et al. proposed TinyLock, which makes the grid size of the Android pattern lock small enough to be covered by only the thumb [17]. TinyLock puts two grids on top and bottom of the screen. The lower grid is for drawing pattern just as in the conventional method while the upper grid displays pressed cells for convenience in registration. Also, the phone vibrates whenever a new cell is pressed to help the user draw pattern. In order to defense against smudge attacks, after drawing the pattern, it is necessary to draw a circular virtual wheel in the clockwise or counterclockwise direction. User experiments show that speed and accuracy are slightly below the existing pattern locks, but smudge attacks can be effectively prevented.

Duplicated and Temporal Codes [18]: In 2016, Ashley Colley et al. proposed a new pattern lock technique, which has the same interface as the Android pattern lock but added two factors. Unlike the Android pattern lock, the selected cell can be selected again. Moreover, it supports long-press where if the pressed duration is more than the threshold, it is regarded as selection of the same cell twice. On the same smudge, this scheme allows different patterns, which implies this scheme has strong resistance to smudge attacks compared with the Android pattern lock. Moreover, the experiment shows that the speed of authentication is similar to that of the original Android pattern lock.

Pass-O [19]: In 2017, Harshal Tupsamudre et al. proposed a new pattern-locking technique, which is called Pass-O. In Pass-O, a 3x3 grid of Android pattern locks are re-arranged in a circular pattern. Pass-O can create up to 985,824 passwords, which is roughly 2.5 times more than existing Android pattern locks. [19] showed that considering the pattern length, the stroke length, and intersection, it is more resistant to the shoulder surfing attack than the existing pattern lock.

Knock Code™ [6]: Unlike other graphical passwords, LG's Knock Code operates even when the screen is off. Suppose that the screen that is divided into 4 areas (virtual 2x2 grid): upper-left, upper-right, lower-left, and lower-right, which is shown in **Fig. 1.** In registration, the user taps one of 4 areas 6~8 times. For log-in, the user should touch the correct area in order. In **Fig. 1**, the number in the box means the area identifier and the number in the circle indicates the tapping sequence, i.e., in this case the password is 2 ⇒ 1 ⇒ 3.

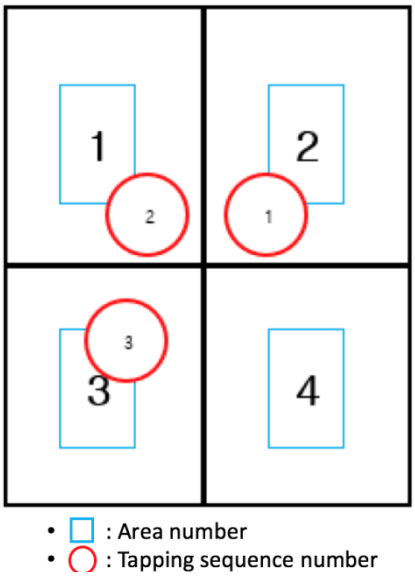

Fig. 1. An example of Knock Code

In **Fig. 2**, we can see that the tapping position is slightly different from **Fig. 1.** The Knock Code automatically determines horizontal/vertical line to split the area according to the first some taps, which is shown in **Fig. 2**. The center point is also moved to the bottom right, since the first touch of the second area is on the lower right side as compared to **Fig. 1.** In this way, regardless of where the first touch is located, the knock code can correctly authenticate the user, which provides flexibility and convinience.

In the case of the Knock Code, the number of possible passwords is 4^6 when password length is 6, 4^7 for 7, and 4^8 for 8, respectively. As a result, the password space size is $4^6+ 4^7+ 4^8= 4{,}096 + 16{,}384 + 65{,}536= 86{,}016$.

Since Knock Code can be used on the off-screen, the shoulder surfing attack is more difficult. Also, it has some resistance against the smudge attack is because every time vertical/horizontal line to split area is changed. When using Knock Code, the user need not look at the screen in tapping, which can be considered as another strong point of Knock Code.

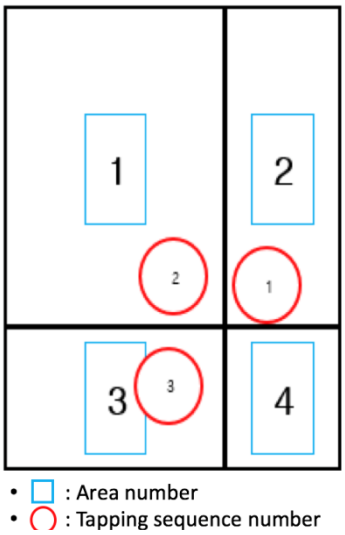

Fig. 2. Another example of Knock Code

3. Proposed scheme

In this section, we present enhanced knock code schemes which support 'slide' and 'touch' operations. Furthermore, they support flexible area recognition, which provides convenience and strong resilience against smudge attacks. First method supports the slide operation while keeping the number of regions as 4 as in the original Knock Code. The second method focuses on convenience for users. It reduces the number of areas to two and supports slides as input. We call them 2x2 knock code and 1x2 knock code, respectively. In Section 3.2 and Section 3.3, we explain 2x2 knock code and 1x2 knock code, respectively. Then, in Section 3.3 we describe flexible area recognition algorithm. Finally, in Section 3.4 we describe the generalized *k*-knock code, which includes 2x2 knock code and 1x2 knock code.

3.1 2x2 knock code

The 2x2 knock code supports slides as input. This scheme focuses on minimizing the number of tapping/sliding operations while providing high security. In this scheme, the user can do the following actions: tapping one of 4 areas or sliding (3 directions from each area). **Fig. 3** shows an example for 3 slides: from area 2 to area 3, to area 1, and to area 4, respectively.

Each tapping or each slide is denoted by a single "*move*." For each move, there are 4+12 = 16 cases exist. If the number of moves is 6 to 8, the size of the password space in 2x2 knock code is $16^6 + 16^7 + 16^8$ = 16,777,216 + 268,435,456 + 4,294,967,296 = 4,580,179,968 (about 4.5 billion). This is 53,248 times bigger than that of the original Knock Code.

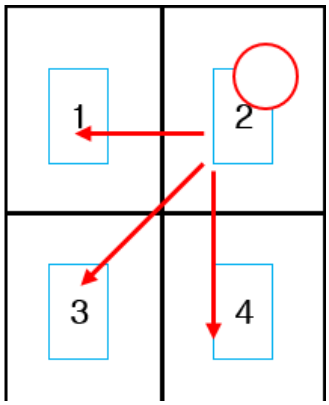

Fig. 3. 3 slides from area 2 in 2x2 knock code

3.2 1x2 knock code

Just as in 2x2 knock code, 1x2 knock code supports the slide operation. This scheme focuses on user's convenience such that it reduces the number of areas to two: left hand side (area 1) and right-hand side (area 2), which is shown in **Fig. 4**. With flexible area recognition, which will be explained in Section 3.3, this scheme has user-friendly interface, i.e., the user can use only his/her thumb for authentication while holding the smartphone.

For each area, the user can do one of the following operations: tapping or sliding. Hence, there are 4 cases for each move. If the number of moves is from 6 to 8, the size of the password space is $4^6 + 4^7 + 4^8 = 86{,}016$, which is exactly the same as that of the original Knock Code.

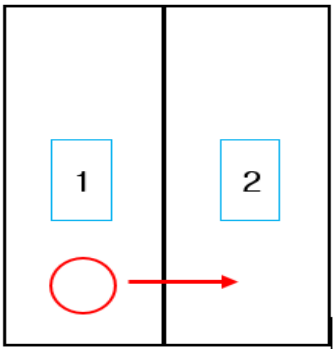

Fig. 4. sliding in 1x2 knock code

3.3 flexible area recognition

In this section, we describe our flexible area recognition algorithm. Our algorithm is based on *k*-means clustering [20]. We modified the original Knock code such that in our knock code, sliding operation is not possible within the same area, i.e., it is allowed in only between different areas.

Suppose that (x_i, y_i) $(1 \le i \le n_1)$ is the position of knock operation and that $(p_{1i}=(sx_i, sy_i), p_{2i}=(ex_i, ey_i))$ $(1 \le i \le n_2)$ is a pair of positions for the sliding operation where (sx_i, sy_i) corresponds to the starting position and (ex_i, ey_i) corresponds to the ending position. Our algorithm has multiple steps to classify all the positions into k groups. E.g., For 1x2 and 2x2

knock code, we divide them into 2 and 4 groups, respectively. Let k denote the number of groups and $\|p, q\|$ denote the Euclidian distance between points p and q. The flexible area recognition algorithm is as follows.

- **STEP 0:** The threshold values t_1, t_2 are initialized to be very small constant. If there is no sliding operation, we choose first k knock positions (x_i, y_i) $(1 \le i \le k)$ as c_i and goto STEP 4.
- **STEP 1:** For the pair of positions (p_1, p_2) for each sliding operation, if there is a point p_j such that $\|p_1, p_j\| \le t_1$ and p_j was already allocated with area A, then p_1 is allocated with area A. Otherwise, p_1 is allocated with a new area B. Repeat this for p_2.
- **STEP 2:** If the number of allocated area is more than k, we increase t_1 (e.g., $t_1=t_1*2$). and repeat STEP 1. If there is a sliding operation (p_1, p_2) such that p_1 and p_2 belong to the same area, we choose first k knock positions (x_i, y_i) $(1 \le i \le k)$ as c_i and goto STEP 4.
- **STEP 3:** For each area S_i $(1 \le i \le j)$, c_i = average of all points in S_i. If $j<k$, we set the remaining c_i $(j+1 \le i \le k)$ as the first k-j unallocated knock positions.
- **STEP 4:** For each point p, find the nearest center c_i: c_i. such that $\|p - c_i\|$ is the smallest for $(1 \le i \le k)$. Let p belong to Area S_i.
- **STEP 5:** For each group S_i, c_i' is set to be the average of all points in S_i: $c_i' = (p_1 + p_2 + \ldots + p_j) / j$, where $p_1, p_2, \ldots, p_j$ are in the same area, S_i.
- **STEP 6:** If $(c_i' - c_i) \le t_2$ for all i $(1 \le i \le k)$, algorithm ends. Otherwise, $c_i = c_i'$ and go to STEP 4.

Example: Assume that k=4, p_1=(1,2), p_2=(4,1), p_3=(4.5,1.2), p_4=(5,7), p_5=(2,7), and p_6=(6,2). **Fig. 5** illustrates these points. Among them, (p_1, p_2), (p_3, p_4) are sliding operations, which are shown in arrows. p_5 and p_6 represent knock operations. Suppose that $t_1=t_2=1$. Since there are 2 sliding operations, we go to STEP 1 from STEP 0. In STEP 1: p_1 is assigned as Area S_1 and p_2 is assigned as Area S_2. Because $\|p_2, p_3\| < 1$, p_3 is assigned as Area S_2. p_4 is assigned as Area S_3. In STEP 2, we go to STEP 3 without changing any values. In STEP 3, $c_1=p_1$, c_2=(4.25, 1.1), $c_3=p_4$, c_4 is set to the first unallocated knock position, p_5. In STEP 4, p_1 is in S_1, p_2 and p_3 are in S_2, p_4 is in S_3, and p_5 is in S_4. For p_6, the nearest c_i is c_2 and p_6 is set to be in S_2. In STEP 5, c_2' is ((4+4.5+6)/3=4.83, (1+1.2+2)/3=1.4) and $c_1'=c_1$, $c_3'=c_3$, $c_4'=c_4$. In STEP 6, the algorithm ends.

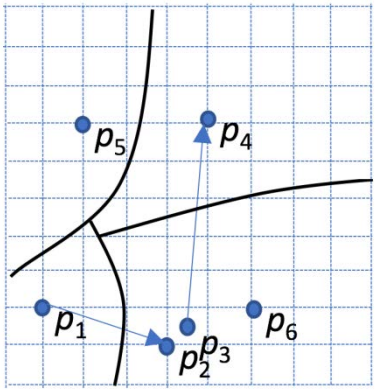

Fig. 5. An example for flexible area recognition.

3.4 Generalized *k*-knock code

In this section, we describe the generalized graphical authentication algorithm: *k*-knock code. This scheme encompasses 2x2 knock code and 1x2 knock code. Also, with flexible area recognition, this scheme provides high user convenience.

Suppose that the screen is divided into k ($2 \leq k$) areas. Note that this scheme does not limit the way to divide, e.g., border line can be in arbitrary shape. **Fig. 6** shows 2 examples for k=3 and k=4.

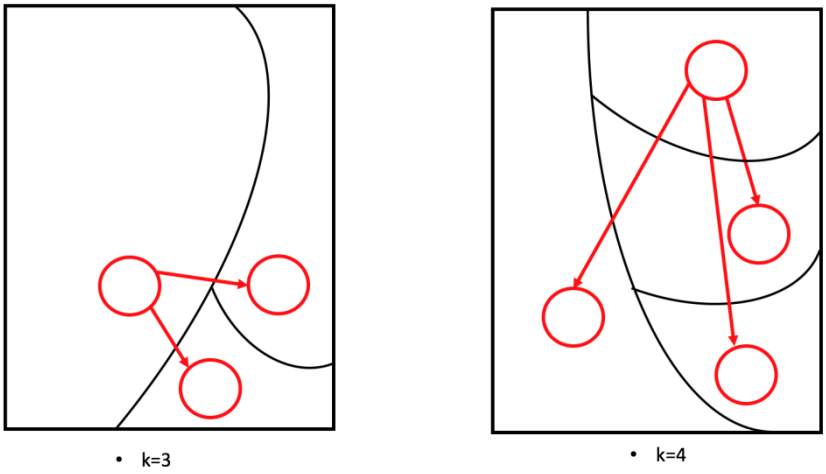

Fig. 6. 2 examples for *k*-knock code (k=3, k=4)

For each area, the user can do one of the following operations: tapping in the area or sliding to another area. Hence, there are k cases for the tapping and k-1 cases for the sliding (i.e., sliding to the same area is not allowed). If the user finishes the touch/sliding operations, this scheme automatically recognizes k areas using the algorithm described in Section 3.3 and then determines authentication success/failure.

In *k*-knock code, for each move (i.e., each touch/sliding operation) totally $2k$-1 cases exist. If the number of moves is from n_1 to n_2, the size of the password space is $\sum_{k=n_1}^{n_2}(2k-1)^k$.

Note that k=2, it is the same as 1x2 knock code with flexible area recognition. If k=4, this scheme includes 4x4 knock code, i.e., the user can use the same password as in 4x4 knock code with the same security.

4. Analysis results: security and convenience

In this section, we analyze security of the proposed schemes. First, in Section 4.1 we compare the size of the password space in the original Knock Code and the proposed schemes. Then, we explain resistance against smudge attacks in Section 4.2. In Section 4.3 we provide user study results on convenience and security of the proposed schemes.

4.1 Size of the password space

The security of the authentication scheme can be quantified as the total number of different passwords that can be created. As mentioned in Section 2, the original Knock Code has totally 4^n passwords where the length of the password is n.

In 1x2 knock code, for each move (i.e., each touch/sliding operation) there are 4 different ways possible and the number of passwords is 4^n when the length of the password is n. For 2x2 knock code, the number of passwords is 16^n. In generalized k-knock code, it is $\sum_{k=n_1}^{n_2}(2k-1)^k$. **Table 1** shows the comparison results between the original Knock Code and the proposed schemes under the same password length. In **Table 1**, we set the password length to be 6 to 8, which is the widely used value in the original Knock Code. In 1x2 knock code, even though the number of areas is reduced by half, the security level is the same.

If the password length is the same, 2x2 knock code provides significantly larger password space, e.g., n=8, the original Knock code can have 65,536 different passwords while 2x2 knock code can set more than 4.2 billion passwords.

Fig. 7 shows the comparison results for each scheme under the same password length. Y-axis represents the number of possible passwords, which is shown in logarithm-scale. Under the password length, 1x2 knock code has the same password length as that of the original Knock Code. In 2x2 knock code, the password space is much larger than that of the original Knock Code. If we set the same password space, the password length of 2x2 knock code is reduced by half.

Table 1. Number of possible passwords for each authentication method (under same password length).

Password Length	LG's Knock Code	1x2 knock code	2x2 knock code	Generalized k-knock code
6	4,096	4,096	16,777,216	$(2k-1)^6$
7	16,834	16,834	268,435,456	$(2k-1)^7$
8	65,536	65,536	4,294,967,296	$(2k-1)^8$

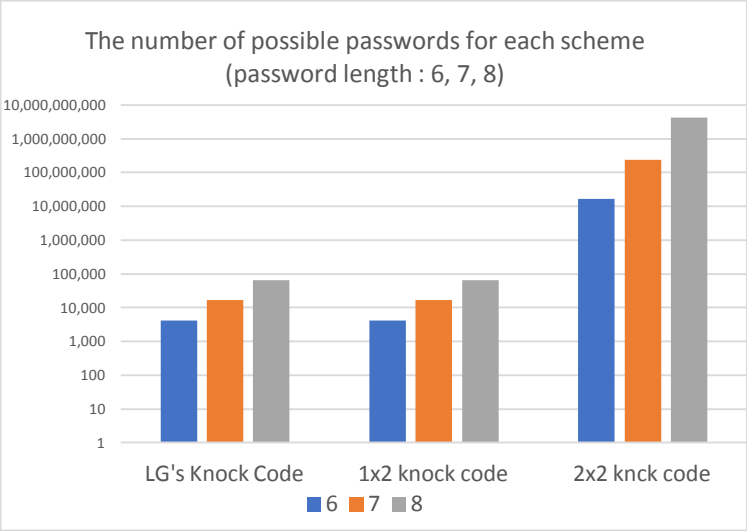

Fig. 7. Comparison results for the number of passible passwords for each authentication method (under the same password length.)

4.2 Resistance analysis against the smudge attack

In this section, we show that compared with the original Knock Code, the proposed method has some resistance to smudge attacks due to the slide operations.

Fig. 8-a) shows an example of smudge left after authentication when the password length is 6. Generally, the original Knock Code leaves 4~8 tapping marks. E.g., in **Fig. 7-a)**, 5 marks are shown. In this case, it is likely to be analyzed easily.

Fig. 8-b) and **Fig. 8-c)** are examples for smudges with length 6 of 1x2, 2x2 knock code respectively. In **Fig. 8-b)**, the left and right slide inputs were done, which removed all remaining smudges during authentication. In **Fig. 7-c)** of 2x2 knock code, most of smudges were erased by diagonal sliding operations. In this way, if the user adds the slide in the password, especially at the end of the password, the smudge can be erased, which shows some resistance against the smudge attacks.

Also, the user study in Section 4.3 shows the similar results: compared with the original Knock Code and Android pattern lock, our scheme is more strongly resistant against the smudge attack.

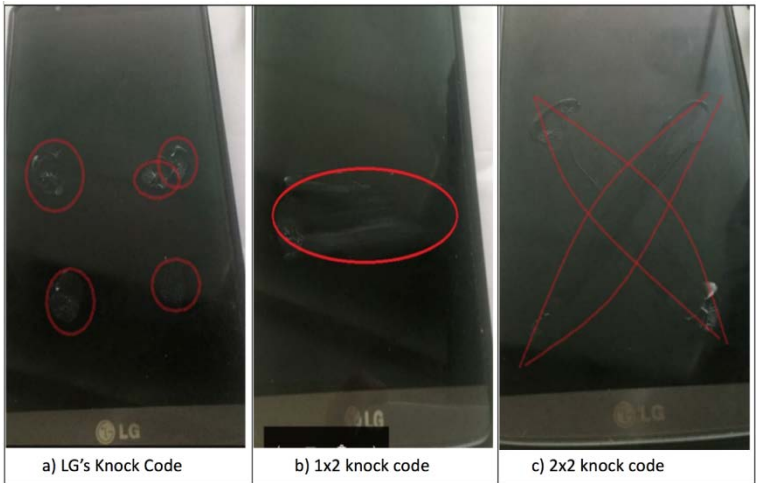

Fig. 8. Examples for smudge in
a) LG's Knock Code, b) 1x2 knock code c) 2x2 knock code

In order to further increase resistance against the smudge attack, our scheme can be slightly modified by enforcing a virtual wheel to be drawn at the end after drawing the pattern, which is used in TinyLock [17].

4.3 User study: usability and security

4.3.1. Design, participants

Experiments were conducted on 20 university students (male: 16, female: 4), who already have experience with Android pattern lock. Among them, 2 are left-handed people and the others are right-handed. Also, they have knowledge of the original Knock Code for unlocking. The smartphone devices that were used in the experiments are Google Nexus 6.

First, we measured the elapsed time for unlocking the phone using the original Knock Code, 1x2 knock code, and 2x2 knock code. Then, we counted the number of success and that of failure for each method. Thirds, we conducted survey regarding to convenience and security.

For the password length of the original Knock Code, we set as 6, 8, and 10. For 2x2 knock code, we set the length as 3,4, and 5 to compare under the same security level. For 1x2 knock code, we set the length as 6, 8, and 10.

4.3.2. Evaluation procedure

Before proceeding with the experiment, we gave brief explanation on the experiments and then conducted basic training. We demonstrated how to unlock the smartphone using the original Knock Code, 2x2 knock code, 1x2 knock code, and Android pattern lock. Then, we had the question and answer time before starting the experiments.

First, all participants were asked to give personal information (name and age). Then, they signed the consent form. Then, participants received the smartphone devices to setup the lock and to conduct unlocking. Also, we measured the number of success/failure for unlocking and the elapsed time for unlocking. The detailed procedure is as follows.

In the experiments, there is the practice time where participants can test/practice the device using diverse locking methods. Then, the participants set the original Knock Code where the password length is 6. Then, they perform the unlock test for 10 times. Then, we repeat this procedure for different password length: 8 and 10.

Next, we conducted experiments for 1x2 knock code, where the procedure is the same for that of the original Knock Code. Then, we conducted experiments for 2x2 knock code. Finally, for Android pattern lock, we repeated the procedure.

After finishing all the tests, participants are asked to fill out the questionnaire regarding to the following questions: “Usability: this method is easy to use?”, “Security: this method is secure enough w.r.t. smudge attack?” Finally, participants were given a certain commodity as compensation for the experiment.

4.3.3. Hypotheses

We use the following hypotheses on the experiments.

- (H1-1) Enhanced 1x2 knock code is more convenient than the original Knock Code.
- (H1-2) Enhanced 2x2 knock code is more convenient than the original Knock Code.
- (H2-1) Enhanced 1x2 knock code is more convenient than the Android pattern lock.
- (H2-2) Enhanced 2x2 knock code is more convenient than the Android pattern lock.
- (H3-1) Enhanced 1x2 knock code is faster than the original Knock Code.
- (H3-2) Enhanced 2x2 knock code is faster than the original Knock Code.
- (H4-1) Enhanced 1x2 knock code is faster than the Android pattern lock.
- (H4-2) Enhanced 2x2 knock code is faster than the Android pattern lock.
- (H5-1) Enhanced 1x2 knock code is more secure than the original Knock Code.
- (H5-2) Enhanced 2x2 knock code is more secure than the original Knock Code.
- (H6-1) Enhanced 1x2 knock code is more secure than the Android pattern lock.
- (H6-2) Enhanced 2x2 knock code is more secure than the Android pattern lock.

4.3.4. Results

1) Entry time

The entry time means the elapsed time for unlocking the screen to use the smartphone, i.e., from the event where the participant first touches the screen to the event where the unlocking is successfully finished or failed. In the experiment, the participant repeated unlocking 10 times for each scheme (Android pattern, LG knock code, 1x2 knock code, 2x2 knock code) and then the average value was calculated.

Table 2. Numerical results of the unlocking time.

Method	Median	Mean	Min	Max	Std
Android pattern lock	2.41s	2.38s	2.01s	2.74s	0.202
Original Knock Code (length=6)	2.03s	2.04s	1.91s	2.17s	0.080
Original Knock Code (length=8)	2.27s	2.28s	2.10s	2.46s	0.102
Original Knock Code (length=10)	2.43s	2.43s	2.20s	2.57s	0.100
Enhanced 1x2 knock code (length=6)	1.98s	1.97s	1.84s	2.10s	0.072
Enhanced 1x2 knock code (length=8)	2.21s	2.19s	1.96s	2.35s	0.099
Enhanced 1x2 knock code (length=10)	2.31s	2.30s	2.12s	2.44s	0.091
Enhanced 2x2 knock code (length=3)	1.40s	1.39s	1.09s	1.60s	0.130
Enhanced 2x2 knock code (length=4)	1.40s	1.43s	1.27s	1.69s	0.097
Enhanced 2x2 knock code (length=5)	1.48s	1.49s	1.30s	1.67s	0.108

Table 2 shows the experimental results. For Android pattern lock, the fastest time to unlock was 2.01 seconds and the slowest time was 2.74 seconds. The average value was 2.38 seconds. In LG Knock Code with 6 digits, the fastest time for a player to unlock was 1.91 seconds and the slowest time was 2.17 seconds. On average, it took 2.04 seconds. For the 10 digits, the fastest time was 2.20 second, the slowest time was 2.57 second, and the average them was 2.43 second.

For 1x2 knock code, it can be seen that the entry time is a little bit faster than that of LG's Knock Code.

In 2x2 knock code for 3 digits, the fastest time, the slowest time, and the average time were 1.09, 1.60, and 1.39 seconds respectively. For 5 digits, the fastest time, the slowest time, and the average time were 1.30, 1.67, and 1.49 seconds, respectively. Compared with the original Knock Code/Android pattern lock, those are approximately 1.6 times faster. This implies that the entry time for 2x2 knock code is faster than that of LG's Knock Code and that of Android pattern lock. With respect to these results, Hypotheses (H3-2) and (H4-2) can be accepted. However, for 1x2 knock code, the speed is slightly faster than the original Knock Code and Android pattern lock, but the difference is not so significant. From this observation, Hypotheses (H3-1) and (H4-1) cannot be accepted.

2) Error rate

We measured how many times the participant failed during the test (totally 10 attempts). Some participants have successfully unlocked in a single attempt, but most participants have failed at least one time. None of the participants has failed unlocking ten times.

Fig. 9 shows the error rates for the enhanced knock code, the original Knock Code, and Android pattern lock. These are the average values where the number of unsuccessful cases is divided into 10 for 2x2 knock code, 1x2 knock code, LG Knock Code and Android pattern lock. This figure shows that 2x2 knock code has the smallest error rates while Android pattern lock has the most.

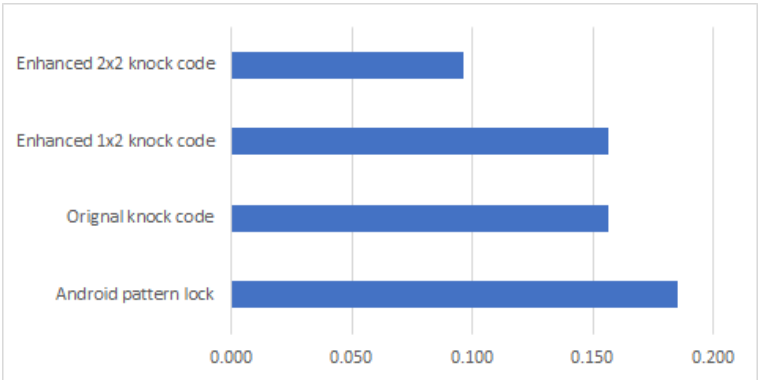

Fig. 9. Error rates for Enhanced knock code, the original knock code, and Android pattern lock.

3) Usability and Security

Usability and security were evaluated based on participants' questionnaires. After the experiment to measure the time required to perform each unlock, participants were asked to fill in the questionnaire given in Section 4.3.2. **Fig. 10** summarizes the results. We used the paired-samples *t*-test to analyze the data.

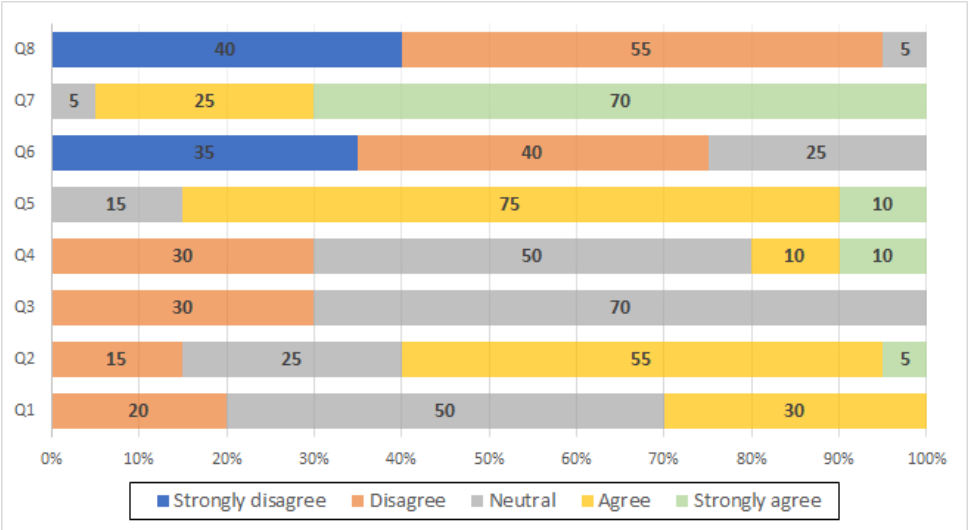

Fig. 10. Score rates regarding the questionnaire. (Questions (Q1, Q3, Q5, Q7): Usability – this method is easy to use?; Security (Q2, Q4, Q6, Q8) – this method is secure enough w.r.t. smudge attack?)

With respect to Hypotheses (H1-1) ~ (H6-2), we prepared 8 questions, which are as follows.

- Q1: Is the Android pattern lock convenient to use?
- Q2: Can the Android pattern lock be guessed by observing fingerprint smudge?
- Q3: Is the original Knock Code convenient to use?
- Q4: Can the original Knock Code be guessed by observing fingerprint smudge?
- Q5: Is the enhanced 1x2 knock code convenient to use?
- Q6: Can the enhanced 1x2 knock code be guessed by observing fingerprint smudge?
- Q7: Is the enhanced 2x2 knock code convenient to use?
- Q8: Can the enhanced 2x2 knock code be guessed by observing fingerprint smudge?

Above questions can be divided into usability and security. Questions about usability include Q1, Q3, Q5 and Q7 while those about security include Q2, Q4, Q6, and Q8. For the answers, Likert-type scale was used for ratings from 1 (strongly disagree) to 5 (strongly agree).

After survey was done, we conducted paired-samples *t*-test to measure statistically significant changes. We use the confidence level of 0.01. First, a paired-samples *t*-test was calculated to compare the convenience of the enhanced 1x2 knock code and the original Knock Code. The analysis produced a significant *t* value ($t(19) = -8.75$, $p=2.148E-08$, $p < 0.01$), which implies that (H1-1) can be accepted. Second, a paired-samples *t*-test was calculated to compare the convenience of the enhanced 2x2 knock code and the original Knock Code. The analysis produced a significant *t* value ($t(19) = -12.71$, $p=4.902E-11$, $p < 0.01$), which implies that (H1-2) can be accepted. From this observation, it can be considered that our enhanced knock code is more convenient than the original Knock Code.

Third, a paired-samples *t*-test was calculated to compare the convenience of the enhanced 1x2 knock code and the Android pattern lock. The analysis produced a significant *t* value ($t(19) = -4.07$, $p=3.243E-04$, $p < 0.01$), which implies that (H2-1) can be accepted. Fourth, a paired-samples t-test was calculated to compare the convenience of the enhanced 2x2 knock code and the Android pattern lock. The analysis produced a significant *t* value ($t(19) = -8.40$, $p=4.056E-08$, $p < 0.01$), which implies that (H2-2) can be accepted. From this observation, it can be considered that our enhanced knock code is more convenient than the Android pattern lock.

Fifth, a paired-samples *t*-test was calculated to compare the security of the enhanced 1x2 knock code and the original Knock Code. The analysis produced a significant *t* value ($t(19) = 4.22$, $p=2.310E-04$, $p < 0.01$), which implies that (H5-1) can be accepted. Sixth, a paired-samples *t*-test was calculated to compare the security of the enhanced 2x2 knock code and the original Knock Code. The analysis produced a significant *t* value ($t(19) = 6.47$, $p=1.683E-06$, $p < 0.01$), which implies that (H5-2) can be accepted. From this observation, w.r.t. the smudge attack, it can be considered that our enhanced knock code is more secure than the original Knock Code.

Seventh, a paired-samples *t*-test was calculated to compare the security of the enhanced 1x2 knock code and the Android pattern lock. The analysis produced a significant *t* value ($t(19) = 6.02$, $p=4.259E-06$, $p < 0.01$), which implies that (H6-1) can be accepted. Eighth, a paired-samples *t*-test was calculated to compare the security of the enhanced 2x2 knock code and the Android pattern lock. The analysis produced a significant *t* value ($t(19) = 8.86$, $p=1.766E-08$, $p < 0.01$), which implies that (H6-2) can be accepted. From this observation,

w.r.t. the smudge attack, it can be considered that our enhanced knock code is more secure than Android pattern lock.

5. Conclusion

In this paper, we proposed the enhanced knock code schemes that support slides as inputs. Our schemes have flexible area recognition, which provides high user convenience. We mathematically analyze security of the proposed schemes, which shows that in 2x2 knock code, the password length is reduced by half under the same security level. With flexible area recognition, 1x2 knock code provides high user convenience. Under the same password length, the total number of passwords in 2x2 knock code is more than 4.5 billion, which is overwhelmingly larger than the original Knock Code. We describe the generalized k-knock code, which encompasses 1x2 and 2x2 knock code. The user study is provided to show convenience and security of the proposed schemes.

References

[1] Robert Biddle, Sonia Chiasson, P.C. van Oorschot “Graphical Passwords: Learning from the First Twelve Years,” *ACM Computing Surveys*, vol. 44, no. 4, pp. 1-41, August. 2012.
[2] Guixin Ye, Zhanyong Tang, Dingyi Fang, Xiaojiang Chen, Kwang In Kim, Ben Taylor, and Zhen Wang, “Cracking Android Pattern Lock in Five Attempts,” in *Proc. of NDSS’2017*, 2017. Article (CrossRef Link)
[3] Arash Habibi Lashkari, Samaneh Farmand, Dr. Omar Bin Zakaria, and Dr. Rosli Saleh “Shoulder Surfing attack in graphical password authentication,” *International Journal of Computer Science and Information Security*, Vol. 6, No. 2, 2009.
[4] M. Kumar, T. Garfinkel, D. Boneh, and T. Winograd, "Reducing Shoulder-surfing by Using Gaze-based Password Entry," in *Proc. of Symposium on Usable Privacy and Security*, pp.13-19, 2007. Article (CrossRef Link)
[5] Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith “Smudge attacks on Smartphone Touch Screens,” in *Proc. of 4th USENIX Workshop on Offensive Technologies*, August. 2010.
[6] LG Electronics, “LG V20 Knock on and Knock Code,” available at http://www.lg.com/us/support/product-help/CT10000025-20150217113217-activation.
[7] M. Rutnik, “LG grabs record 20% of US smartphone market,” available at https://www.androidauthority.com/lg-grabs-record-smartphone-market-share-us-770033/, 2017 (accessed October 8 2017).
[8] LG Electronics, “Key LG Smartphones to get Knock Code Upgrade,” 03-26-2014, available at http://www.lg.com/ae/press-release/key-lg-smartphones-to-get-knock-code-upgrade.
[9] Ian Jermyn, Alain Mayer, Fabian Monrose, Michael K. Reiter, and Aviel D. Rubin “The design and analysis of graphical password,” in *Proc. of 8th USENIX Security Symposium*, August. 1999.
[10] Paul Dunphy and Jeff Yan, “Do Background image improve ‘draw a secret’ graphical passwords?,” in *Proc. of the 14th ACM Conference on Computer and Communications Security, CCS’07*, pp.36-47, 2007. Article (CrossRef Link)
[11] Hai Tao and Carlisle Adams, “Pass-go: A proposal to improve the usability of graphical passwords,” *International Journal of Network Security*, vol.7, no. 2, pp.273-292, 2008.
[12] H. Gao, X. Guo, X. Chen, L. Wang and X. Liu, “YAGP: Yet Another Graphical Password Strategy,” in *Proc. of ACSAC’98*, 2008. Article (CrossRef Link)
[13] C. Varenhorst, M. V. Kleek, and L. Rudolph, “Passdoodles: A Lightweight Authentication Method,” available at http://people.csail.mit.edu/emax/public_html/papers/varenhorst.pdf, 2004.
[14] R. Weiss and A. De Luca, “PassShapes: Utilizing Stroke Based Authentication to Increase Password Memorability,” in *Proc. of Nordic Conference on Human-Computer Interaction*

(NordiCHI), 2008. Article (CrossRef Link)
[15] Gmalto, "GrIDsure: One-time password (OTP) without hardware tokens or software applications," 2015, available at Article (CrossRef Link).
[16] Hsin-Yi Chiang, Sonia Chiasson, "Improving user authentication on mobile devices: A Touchscreen Graphical Password," in *Proc. of International Conference on MobileHCI*, pp. 251-260, January. 2013.
[17] Taekyoung Kwon and Sarang Na, "TinyLock: Affordable defense against smudge attacks on smartphone pattern lock systems," *Computers and Security*, Vol. 42, pp. 137-150, May. 2014.
[18] Ashley Colley, Tobias Seitz, Tuomas Lappalainen, Matthias Kranz, and Jonna Häkkilä1 "Extending the Touchscreen Pattern Lock Mechanism with Duplicated and Temporal Codes," *Advances in Human-Computer Interaction*, vol. 2016, November. 2016.
[19] Harshal Tupsamudre, Vijayanand Banahatti, and Sachin Lodha "Pass-O: A Proposal to Improve the Security of Pattern Unlock Scheme," in *Proc. of ASIACCS'2017*, April. 2017. Article (CrossRef Link)
[20] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-Means Clustering Algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100-108, 1979. Article (CrossRef Link)

Yun-Hwan Jang received the B.E. degree in Computer Science from Hongik University, South Korea, in 2015. He currently is a M.S. student in Information Security from Hanyang University, Seoul, Korea. His main research interests include android security, web Security, malware analysis, and network security.

Yongsu Park received the B.E. degree in Computer Science from Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 1996. He received the M.E. degree and the Ph.D. degree in Computer Engineering from Seoul National University in 1998 and 2003, respectively. He is currently a professor in the Department of Computer Science at Hanyang University, Seoul, Korea. His main research interests include computer system security, malware analysis, operating system security, and network security.