SysPal: *Sys*tem-guided *Pa*ttern *L*ocks for Android

Geumhwan Cho[1], Jun Ho Huh[2], Junsung Cho[1], Seongyeol Oh[1], Youngbae Song[1], Hyoungshick Kim[1]
[1]Department of Computer Science and Engineering, Sungkyunkwan University, South Korea
[2]Software R&D Center, Samsung Electronics, South Korea
Email: {geumhwan, js.cho, seongyeol, youngbae, hyoung}@skku.edu
etpfest@gmail.com

***Abstract*—To improve the security of user-chosen Android screen lock patterns, we propose a novel *system-guided* pattern lock scheme called "SysPal" that mandates the use of a small number of randomly selected points while selecting a pattern. Users are given the freedom to use those mandated points at any position.**

We conducted a large-scale online study with 1,717 participants to evaluate the security and usability of three SysPal policies, varying the number of mandatory points that must be used (upon selecting a pattern) from one to three. Our results suggest that the two SysPal policies that mandate the use of one and two points can help users select significantly more secure patterns compared to the current Android policy: 22.58% and 23.19% fewer patterns were cracked. Those two SysPal policies, however, did not show any statistically significant inferiority in pattern recall success rate (the percentage of participants who correctly recalled their pattern after 24 hours). In our lab study, we asked participants to install our screen unlock application on their own Android device, and observed their real-life phone unlock behaviors for a day. Again, our lab study did not show any statistically significant difference in memorability for those two SysPal policies compared to the current Android policy.

I. Introduction

To help Android users select memorable and secure authentication secrets, Google introduced a graphical password scheme in 2008 based on "Pass-Go" [23]. Android users are asked to select and remember a graphical pattern on a 3×3 grid – this scheme is interchangeably referred to as "Android pattern lock," "Android pattern unlock," or "Android screen lock pattern." The Android pattern lock scheme, with its strong usability [27], has quickly emerged as the most popular screen locking method for Android devices [25]. Graphical patterns are also used to protect mobile payment applications. For example, PayNow (http://www.paynow.co.kr) is a popularly used mobile payment application (with more than one million users) that requires users to prove their identity using a pattern before processing a payment.

However, as with any user-chosen password scheme, people still choose easy-to-remember patterns that are also easy to guess. Previous studies [22, 24] showed that the actual password space of user-chosen patterns is much smaller than the theoretical space of 389,112 possible patterns. Even though the real-world pattern space is probably larger than the real-world 4-digit PIN space [22], user-chosen patterns are still weak against guessing attacks [4].

To mitigate guessing attacks, Android enforces a policy that only allows up to 20 consecutive fail unlock attempts. However, even with the fail-attempt limit policy used on Android (allowing maximum 20 guesses), attackers could still successfully guess a significant portion of user-chosen patterns (about 16.7% [4]). Pattern strength meters [22] and the use of a bigger grid (e.g., 4×4) layout [4] have been suggested as a way to help users select stronger patterns. The impact of such solutions are limited though: about 20% of user patterns were still cracked when a 4×4 grid was used [4], and even with the pattern strength meter in place, user behavior in choosing the starting (first position) point was still strongly biased [22].

To overcome the security limitations of user-chosen patterns, we propose a novel, system-guided pattern selection scheme called "SysPal" (**Sys**tem-guided **Pa**ttern **l**ocks). Our goal was to improve the security of existing Android screen lock patterns against guessing attacks while following four design principles: (i) minimize additional memorability burden on users, (ii) keep the authentication or unlock time similar, (iii) make them easy to learn and use, and (iv) avoid significant software or hardware changes. SysPal *mandates* users to use a few randomly chosen points (at any position) upon selecting a pattern, guiding users to deviate from their normal, biased pattern selection behaviors. Various user persuasion techniques [10, 12, 15, 21] for improving password security have been proposed before. Such persuasion techniques and SysPal share a common goal of helping users move away from their biased password selection behaviors (that are easy to guess). Nevertheless, SysPal differentiates itself from other techniques by *mandating* the use of random points but also giving users the freedom to use those points *at any position* upon selecting a pattern (i.e., not necessarily as a starting point). SysPal aims to improve the security of an *existing* graphical password scheme that is popularly used with minimal compromise in usability.

To evaluate the security and usability of SysPal patterns, we conducted a large-scale online user study, recruiting a total of 1,717 participants through Amazon Mechanical Turk. We tested our approach by varying the "number of mandated points" that must be used upon selecting a pattern from one to three. We compared the security and recall success rate (percentage of users who successfully recalled their pattern after approximately 2 minutes, 15 minutes, and 24 hours) of those SysPal patterns against both the randomly-generated patterns and original Android patterns.

Our evaluation results suggest that SysPal significantly improves the security of patterns compared with the original Android patterns while producing similar recall success rates. Our lab study, conducted on 46 participants, confirmed this observation, showing that the memorability of SysPal patterns

DOI 10.1109/SP.2017.61

are not statistically inferior compared to original patterns. To achieve strong ecological validity, we implemented an Android screen lock application that uses the SysPal policies, and asked the participants to use it on their own phone for a day. Whenever the participants tried to unlock their phone, the application asked them to enter a pattern created under the SysPal policies, and recorded success and fail attempts.

Meanwhile, the performance of our Markov model-based guessing attack significantly decreased, cracking 32.55% of original patterns compared to cracking just 9.36% of SysPal patterns when two random points were mandated. Interestingly, increasing the number of mandated points did not improve the security against guessing attacks. This implies that the number of mandated points needs to be decided with caution.

We summarize the key contributions of this paper as follows:

1) We proposed SysPal, a novel system-guided pattern selection scheme for improving the security of Android patterns.
2) We performed a large-scale empirical evaluation of three SysPal policies, which suggests that SysPal can significantly improve pattern security with just small compromise in recall success rates.
3) We also performed a lab study to test SysPal under real-life unlock scenarios, which suggests that the difference in memorability between SysPal patterns and original Android patterns is not statistically significant.
4) We performed a large-scale empirical comparison of randomly-generated patterns with the original Android patterns. Our results showed that the recall success rate of randomly-generated patterns is about 21.80% lower.

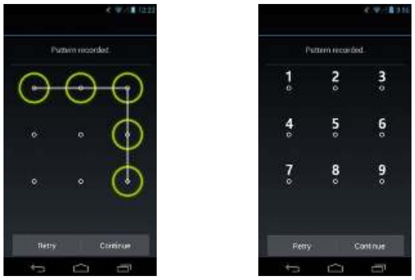
The rest of paper is organized as follows. Section II describes an attack scenario. Section III describes how the user study was designed. Sections IV and V present the key usability and security results, respectively. Section VI presents the results of a separate lab study. Section VII discusses how the results match up to our hypotheses. Related work is covered in Section VIII, and our conclusions are in Section IX.

II. Background

A. Android screen lock patterns

Android screen lock pattern is one of the most widely adopted graphical password schemes [2]. Users have to remember a pattern consisting of consecutive segments (lines connecting two points) on a 3×3 grid. During registration, users are asked to select a secret pattern. Then users are asked to recall their secret pattern upon authentication, and draw it on the grid to unlock their Android device (see Figure 1(a)).

For notational convenience, the following conventions are adopted throughout the paper: the 9 points on the grid are numbered (indexed) from 1, starting with the point located at the top left corner, to 9, which is the point located at the bottom right corner of the grid (see Figure 1(b)). A segment in a pattern is defined as "a line that connects two points" together. The original Android pattern policy requires a pattern to consist of at least four points, and prevents a point from being used more than once in a given pattern.



(a) Android pattern (b) Numbers on 3×3 grid

Fig. 1: Android screen lock pattern example, and the indexing numbers assigned to each of the 9 points on the 3×3 grid.

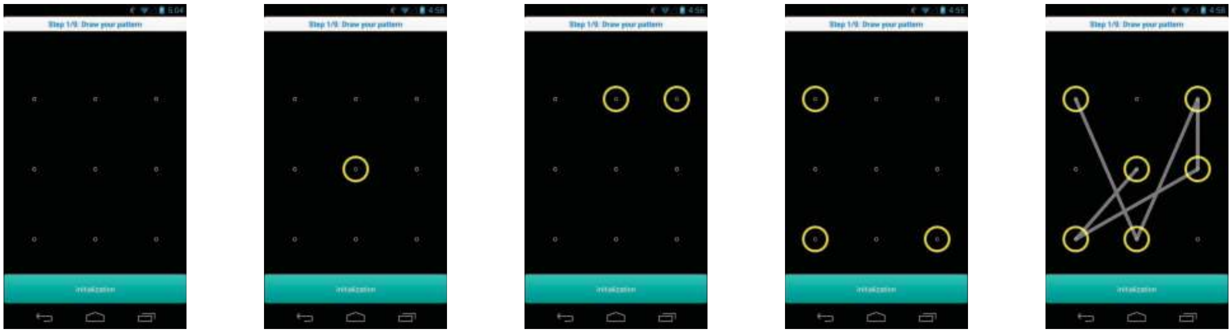
B. Threat model for Android patterns

There are many known attacks against Android patterns, including smudge attacks [2, 5, 9], sensor-based side channel attacks [6], shoulder surfing attacks [26, 29], and guessing attacks [4, 22, 24]. This paper only focuses on evaluating the robustness of SysPal patterns against a pattern dictionary-based guessing attack that involves an attacker getting access to an Android device, and trying out most likely patterns first to unlock it.

In theory, the total number of all possible patterns is 389,112 ($\approx 2^{18}$), which is much larger than the password space of 10,000 4-digits PINs. Despite this relatively larger password space, users still choose weak patterns that are easily guessable [22, 24]. To mitigate guessing attacks performed on such weak patterns, Android only allows up to 20 consecutive fail unlock attempts. If a user fails to draw the correct unlock pattern within 5 attempts, the device is temporally locked for 30 seconds; after 20 consecutive fail attempts, Android displays the "Too many pattern attempts" error message, and asks the user to log in with a Google account to unlock the device. That is, the attacker cannot try more than 20 attempts. Thus, we assume that the attacker's goal is to unlock a target device within 20 trials. Intuitively, if the attacker has no information about the pattern being used, the best attack strategy would be to try the top 20 most commonly used patterns first. The attacker could use a probabilistic password model [19] such as the n-gram Markov model (this is explained further in Section V), training a Markov model using a real-world set of Android patterns to find the top 20 patterns.

III. Methodology

This section lists our research questions and hypotheses, and explains the user study design. Our work was motivated by the following research question: "Can we design security policies for Android screen lock patterns to improve their security without significantly compromising their usability?"



(a) Original (b) 1-Point (c) 2-Point (d) 3-Point (e) Random

Fig. 2: Initial setup screen examples for the five policies tested.

Based on the research question, we defined the following three hypotheses: (H1) The security of SysPal patterns strengthens with the increase in the number of mandated points; (H2) The memorability of SysPal patterns decreases with the increase in the number of mandated points; (H3) A SysPal policy that shows no statistically significant difference in memorability against the original Android patterns has *better security* than those original patterns. The user study and experiments were designed to validate the above hypotheses. In Section VII, we discuss how the study results match up to those hypotheses.

A. SysPal policies

Each SysPal policy defines *the number of randomly-selected mandated point(s) that must be used once* upon selecting a pattern (see Table I). To test Hypotheses 1 and 2, we created SysPal policies from `1-Point` to `3-Point`, increasing the number of mandated points. We only focused on those three SysPal policies because mandating too many points could potentially reduce the overall password space. We evaluated the performance of those SysPal policies compared against the original Android policy: policy `Original` was created to replicate the real-world Android pattern policy that requires a pattern to consist of at least four points, and a point to be used only once. It was used to test hypothesis 3. Finally, we considered randomly generated patterns named `Random` as the most ideal pattern policy for security. We were also interested in comparing the security of SysPal patterns against system-generated, random patterns. Figure 2 shows initial setup screen examples for the five policies we experimented with. For `Random` patterns (see Figure 2-(e)), starting points and ending points are obvious as users are presented with animations showing how patterns start and end. In our experiments, each policy was used as a separate experimental condition.

B. User study design

TABLE I: Description of the five policies we experimented with.

Policy	*Description*
`Original`	Users should choose a pattern based on the following rules: (1) at least four points need to be used, (2) a point can only be used once, and (3) an unused point can not be jumped over.
`1-Point`	Users should choose a pattern that uses *one* randomly assigned point.
`2-Point`	Users should choose a pattern that uses *two* randomly assigned points.
`3-Point`	Users should choose a pattern that uses *three* randomly assigned points.
`Random`	Given a system-generated random pattern, users should remember that pattern.

We evaluated the effectiveness of the five policies (see Table I) through quantitative experiments conducted using Amazon Mechanical Turk. Before starting the study, participants were asked to acknowledge a consent form, which explained the study purposes and instructions, and informed that participations are voluntary and confidential, and they have the right to terminate the study any time without penalty. Data were collected confidentially only for the purposes of conducting statistical analyses. Ethical perspective of our research was validated through an institutional review board (IRB) at a university.

To evaluate the SysPal policies in a realistic setting, we developed an Android application that simulates the real-world Android pattern setup and unlock tasks. Using that application, we collected the participants' behavioral data to examine how they choose a pattern and use it to unlock their device. We only recruited individuals who own an Android device. Only when a participant agreed with our consent form, we asked the participant to install the application on their Android device. After starting the application, the participant was asked to select a pattern under one of the five policies (randomly assigned). We used between-subject comparisons of the five policies to avoid order effects. While selecting a pattern, the participants were given an option to "initialize" (reset) the grid unlimited number of times to start from the beginning if they wanted to. For the SysPal policies, the mandated points were fixed though, and did not change upon initialization. For those assigned to the `Random` policy, initialization allowed them to completely reset a given random pattern if they wanted to use a different one.

Our user study was designed following the Atkinson-Shiffrin dual memory model [3]. This model postulates that human memories initially reside in a "short-term" memory for a limited time (20 to 30 seconds). Short-term memory has limited capacity and older items are wiped as new items enter. Further, rehearsing or recalling items while they are in the short-term memory causes the items to stay longer in the short-term memory. Based on Atkinson-Shiffrin memory model, each participant was first asked to complete two training tasks (for rehearsing) to help the participant remember the selected pattern (for associating long-term memory with that pattern). Next, the participant was asked to complete a graphical puzzle, which was intended to wipe out the participant's short-term memory of the selected pattern information (with new items) during the process. After solving the puzzle, the participant was asked to complete three pattern recall tests (at different time intervals) to check whether the participant can remember the selected pattern over time.

The following paragraphs present details of the data collection procedures in the order participants were asked to complete them.

1. Pattern setup: Each participant was randomly assigned to one of the five policies in Table I. For all SysPal policies, randomly selected points were highlighted with yellow circles (see Figure 2). Participants assigned to the SysPal policies were asked to generate a pattern that must use all of those highlighted points. Those assigned to the `Original` policy were asked to generate a pattern based on the original Android pattern setup rules (see Table I). Those assigned to the `Random` policy were asked to remember the given randomly generated pattern as is.

2. Pattern memorization: Each participant was asked to draw the correct pattern two times to help with memorization. If incorrect patterns were entered five times consecutively, the correct pattern was revealed again so that the participant would have another chance to memorize it.

3. Puzzle: Each participant was asked to complete a moderately challenging lexical and graphical puzzle, which takes about 2 minutes to complete.

4. Demographics questions: Each participant was asked demographic questions such as ethnicity, age, gender and level of education.

5. Draw pattern: Each participant was asked to draw his or her chosen pattern within five attempts (In Android, if a user fails to draw the correct pattern within five attempts, his or her device is temporally locked for 30 seconds).

6. Survey questions: After completing the pattern recall test (at step 5), participants were asked to answer the survey questions listed in Table II. Only those who correctly recalled their pattern were invited to the next pattern recall test.

Steps 5 and 6 were repeated after 15 minutes for the second recall test, and repeated again after 24 hours for the third recall test. To remind the participants about those two tests, our Android application was designed to send out vibration notifications at appropriate times. Each participant also received an email 24 hours after completing the second recall test, inviting him or her to the third recall test. In our study, the two 15 minute and 24 hour break periods were selected to reflect on the real-world smartphone lock and unlock frequencies.

TABLE II: Survey questions asked after each pattern recall test. For Q2, the five-level Likert item format was "very difficult," "difficult," "neutral," "easy," and "very easy."

#	*Question*	*Answers*
`Q1`	Did you use an external storage (e.g., a sheet of paper or a text file) to write down your pattern?	yes/no
`Q2`	How difficult was it for you to remember your pattern?	Likert scale
`Q3`	Did you use any special technique (e.g., creating character patterns, forming dictionary words) to help you create and remember your pattern?	yes/no
`Q4`	If you answered "Yes" to Q4, what was the special technique that you used?	Open ended

According to the results presented in [14], the average daily number of interactions with smartphones was 57 times. Assuming that the daytime is about 14 hours, the mean break duration between unlock sessions is about 15 minutes. We also used a break of 24 hours for the third recall test based on the assumption that a typical user would unlock his or her device at least once a day. Even when people are on vacation, it is likely that they would carry their phone with them, and unlock their phone at least once a day. Those usage scenarios and break durations are quite different to the way people would use their passwords for logging into websites.

To prevent the participants from simply taking a snapshot of their pattern and cheating, we disabled the screen capture feature from our application. Before running the real user study, we conducted several pilot studies with a total of 1,118 participants to fix bugs, and address unclear instructions and descriptions.

C. User data collected

Throughout the steps of the user study described in Section III-B, we recorded the following information:

Selected pattern and pattern policy: For each participant, we recorded the selected pattern and the assigned pattern policy.

Number of initialization attempts: For each participant, we recorded the number of times a participant tried to reset the grid during pattern setup.

Number of unlock attempts: For all three pattern recall tests, we recorded the number of attempts each participant made in drawing the selected pattern.

Time taken for pattern setup: We measured the time it took each participant to set up his or her pattern, starting from when the participant first saw the pattern screen view and ending when the participant successfully selected a pattern. To complete the setup process, each participant had to select a pattern that conformed to the given pattern policy.

Time taken to authenticate: For all of the recall tests, we measured the time it took each participant to complete an authentication attempt, starting from when the participant first saw the login screen and ending when the participant either drew the correct pattern or failed to draw the correct pattern within five attempts.

Pattern recall results: For all of the pattern recall tests, we recorded the results of the recall tests (i.e., whether a correct pattern was entered) for each attempt made.

Survey answers: We recorded the participants' answers to the survey questions in Table II.

D. Mechanical Turk

To conduct a large-scale study, we used Amazon Mechanical Turk, recruiting a sufficiently large number of participants to perform meaningful statistical analyses. Participants had to be located in the United States, and at least 18 years of age. The participants who completed the first and second recall tests were rewarded with USD 1.50. The participants who came back and completed the third recall test were rewarded with additional USD 0.50.

E. Statistical tests

Without making any assumptions on data distributions, we performed the Fisher's Exact Test (FET) to compare the proportion of cracked patterns, starting/end/overall point(s), and pattern recall success rate based on survival rates for the five policies in Table I. The statistical confidence in the pattern setup time, authentication time and attempts between the five policies were tested using two-tailed unpaired t-test due to the collected data was normally distributed (tested by Shapiro-Wilk's test). Recall difficulty was tested using unpaired Mann-Whitney U test (MW U test) because a skewed distribution was found in the participants' Likert scale responses. Post-hoc comparisons were corrected for multiple-testing using the Bonferroni correction estimation when appropriate.

IV. Results: usability

This section presents the key usability results from the user study, discussing pattern recall success rate and authentication time.

A. Demographics

A total of 1,717 participants completed the first pattern recall test. 1,603 came back to complete the second pattern recall test, and 1,236 came back complete the third pattern recall test. Most of the participants were white (71%), and the majority were in the 18–29 (61%) and 30–39 (29%) age groups. About 51% were male. 54% had a university degree, and 37% had a high school diploma. The details of the demographics are presented in Table XIII (see Appendix A).

B. External storage usage

After completing each pattern recall test, we asked the participants about the use of an external storage (see `Q1` in Table II). There were only 44 participants who used an external storage for at least one of the tests. 54.5% of such participants were assigned with the `Random` policy. Most of them (84.1%) used a piece of paper as an external storage. We discarded those participants' records to precisely measure pattern "recall success rate." Therefore we, in the rest of paper, used this filtered dataset excluding the participants who used an external storage. We paid all the participants regardless of their answer to this question though.

C. Recall success rate

We analyze the number of participants who successfully recalled their pattern in the three recall tests to compare the recall effects of the five policies presented in Table I.

1) Survival rates: First, we simply counted the number of the remaining participants who successfully recalled their pattern in all tests against the number of all initial participants. We note that there were many participants who did not return to complete the second or third test. 21.93–25.68% of those assigned to the SysPal policies, 24.74% of those assigned to the `Original` policy, and 16.77% of those assigned to the `Random` policy did not return to complete the second or third test. If we categorize those participants as failed participants, the formula shown in Table III, (# remaining participants)/(# initial participants), can be applied.

TABLE III: (# remaining participants)/(# initial participants) with 95% binomial confidence intervals across five policies.

Policy	*First Test*	*Second Test*	*Third Test*
`Original`	382/384 99.48% 0.98, 1.00	365/384 95.05% 0.92, 0.97	278/384 72.40% 0.68, 0.77
`1-Point`	326/331 98.49% 0.97, 1.00	317/331 95.77% 0.93, 0.98	232/331 70.09% 0.65, 0.75
`2-Point`	340/342 99.42% 0.98, 1.00	330/342 96.49% 0.94, 0.98	252/342 73.68% 0.69, 0.78
`3-Point`	320/326 98.16% 0.96, 0.99	312/326 95.71% 0.93, 0.98	231/326 70.86% 0.66, 0.76
`Random`	276/334 82.63% 0.78, 0.87	265/334 79.34% 0.75, 0.84	169/334 50.60% 0.45, 0.56

The first test results show the participants' recall success rate soon after solving a puzzle (which takes about 2 minutes). Interestingly, about 17.37% of the `Random` participants failed to recall their pattern in the first test. In all other policies, only 0.52–1.84% failed the first test. The first-test recall success rate of all SysPal policies (98.16–99.42%) and `Original` policy (99.48%) were better than `Random` policy (82.63%), showing statistically significant differences (all $p < 0.001$, corrected FET). However, there was no statistically significant difference between the SysPal polices and `Original` policy (all $p = 1.0$, corrected FET).

In the second test (taken at least 15 minutes after the first test) the recall success rate difference between the `Random` policy and all other policies increased. For all SysPal and `Original` participants, only 3.51–4.95% failed to recall their pattern; 2.19–4.45% did not return to the second test even though they had successfully recalled their patterns in the first test. All SysPal policies (95.71–95.77%) and `Original` policy (95.05%) showed statistically significant superiority over Random policy (79.34%) in the second-test recall success rate (all $p < 0.001$, corrected FET).

Similarly, after the third test (taken at least 24 hours after the second test), we noticed further increase in the recall success rate difference between the Random policy and all other policies. For all SysPal and Original participants, 26.32–29.91% failed to recall their patterns; 20.30–24.59% did not return to the third test even though they had successfully recalled their pattern in the second test. All SysPal policies (70.09–73.68%) and Original policy (72.40%) showed statistically significant superiority over Random policy (50.60%) in the third-test recall success rate (all $p < 0.001$, corrected FET). Again, we did not find any statistically significant difference in recall success rate between the Original and all SysPal policies (all $p = 1.0$, corrected FET).

2) Excluding those who dropped out without failing: Even though we tried our best to bring back the participants to subsequent tests – sending notifications and offering bonuses – some participants still dropped out without failing. To accommodate for such drop out rates, we used another recall success rate metric that excludes those who dropped out without failing (see Table IV). We used the formula of (# remaining participants)/(# returned participants + # dropped out after failing one of previous tests).

TABLE IV: (# remaining participants)/(# returned participants + # dropped out after failing one of previous tests) with 95% binomial confidence intervals across five policies.

Policy	*First Test*	*Second Test*	*Third Test*
Original	382/384	365/(365+2)	278/(287+2)
	99.48%	99.46%	96.19%
	0.98, 1.00	0.98, 1.00	0.93, 0.98
1-Point	326/331	317/(318+5)	232/(240+6)
	98.49%	98.14%	94.31%
	0.97, 1.00	0.96, 0.99	0.90, 0.97
2-Point	340/342	330/(332+2)	252/(263+4)
	99.42%	98.80%	94.38%
	0.98, 1.00	0.97, 1.00	0.91, 0.97
3-Point	320/326	312/(313+6)	231/(236+7)
	98.16%	97.81%	95.06%
	0.96, 0.99	0.96, 0.99	0.92, 0.97
Random	276/334	265/(275+58)	169/(210+68)
	82.63%	79.58%	60.79%
	0.78, 0.87	0.75, 0.84	0.55, 0.67

Compared to the first metric, this second metric computed similar survival rates for the first two tests across all policies. But it computed much higher third-test recall success rate, showing more significant differences in the third-test recall success rate between the Random policy (60.79%) and all other policies (94.31–96.19%).

For all three tests, we did not find any statistically significant difference in recall success rate between the Original and all SysPal policies. In fact, the overall recall success rate of SysPal patterns was not too different from Original patterns. But all of those policies demonstrated statistically significant superiority over Random (all $p < 0.001$, corrected FET).

D. Authentication time and attempts made

We measured the authentication time (time taken to unlock a device) by adding the preparation time (time taken until the first touch) and the input time (time measured after the first touch until a device is unlocked) [28]. Table V shows the *mean* time taken to complete the authentication process for all three recall tests. Appendix B visually compares the authentication times between all the policies.

TABLE V: Mean time (sec) taken to complete authentication across the five policies (μ: mean, σ: standard deviation).

Policy	*First Test*		*Second Test*		*Third Test*	
	μ	σ	μ	σ	μ	σ
Original	4.60	3.56	4.73	3.64	6.31	5.13
1-Point	4.26	2.76	4.07	2.76	6.53	6.75
2-Point	4.17	2.94	4.38	3.95	5.97	5.32
3-Point	4.47	4.30	4.52	4.77	5.79	6.13
Random	12.90	10.70	9.15	7.59	13.65	12.77

In the first test, all SysPal policies (4.17–4.47 seconds) and Original policy (4.60 seconds) outperformed Random policy (12.90 seconds) with statistical significance (all $p < 0.001$, corrected two-tailed unpaired t-test). Between all SysPal policies and Original policy, there was no statistically significant difference in authentication time ($p = 1.0$, $p = 0.74$, $p = 1.0$ for 1-Point, 2-Point, and 3-Point, respectively, corrected two-tailed unpaired t-test). The second-test authentication time results were not too different for the SysPal policies and Original policy ($p = 0.76$, $p = 1.0$, $p = 1.0$ for 1-Point, 2-Point, and 3-Point, respectively, corrected two-tailed unpaired t-test), but the mean authentication time for the Random policy significantly decreased. In the third test, the mean authentication time for all policies increased. This seems natural since the third test was taken at least 24 hours after the second test. In the third test, all SysPal policies (5.79–6.53 seconds) and Original policy (6.31 seconds) still outperformed Random policy (12.77 seconds) with statistical significance (all $p < 0.001$, corrected two-tailed unpaired t-test).

TABLE VI: Mean number of authentication attempts made across the five policies (μ: mean, σ: standard deviation).

Policy	*First Test*		*Second Test*		*Third Test*	
	μ	σ	μ	σ	μ	σ
Original	1.14	0.50	1.11	0.36	1.24	0.74
1-Point	1.18	0.61	1.11	0.28	1.22	0.74
2-Point	1.12	0.47	1.10	0.40	1.29	0.83
3-Point	1.19	0.65	1.08	0.34	1.20	0.64
Random	2.26	1.52	1.70	1.15	2.16	1.57

Table VI shows the *mean* number of attempts made to complete the authentication process for all three recall success rate tests. The number of attempts made by those who failed the tests (i.e., 5 attempts) were included in the average. In the first test, the mean number of attempts for all SysPal policies (1.12–1.19) and Original policy (1.14) were less than that of the Random policy (2.26), showing statistically significance differences (all $p < 0.001$, corrected two-tailed unpaired t-test). This shows that the Random participants had to draw their pattern twice on average to unlock their device. This could be a significant usability issue for using random patterns. The mean number of attempts for all policies reduced slightly in the second test but increased again in the third test.

E. Pattern setup time

We also measured the time taken to set up a pattern, starting from when a participant is first given the pattern selection screen, and ending when a pattern is finally selected. The time spent on initializing the grid was included in this setup time as well.

As shown in Table VII, the mean time taken to set up a pattern was similar across all policies except for the Random policy. Random showed the longest mean setup time of 36.9 seconds, and its inferiority against all other policies was statistically significant (all $p < 0.001$, corrected two-tailed unpaired t-test). Again, we did not find any statistically significant difference between Original and all SysPal policies (all $p = 1.0$, corrected two-tailed unpaired t-test).

TABLE VII: Mean time (sec) taken to set up a pattern, and mean number of pattern initialization (reset) performed while setting up a pattern (μ: mean, σ: standard deviation).

Policy	*Setup time*		*# Initialization*	
	μ	σ	μ	σ
Original	22.05	15.63	0.07	0.27
1-Point	22.04	14.93	0.07	0.31
2-Point	22.02	17.49	0.08	0.29
3-Point	22.50	18.97	0.08	0.12
Random	36.91	22.01	0.12	0.58

We allowed the participants to initialize (reset) the grid unlimited number of times while setting up a pattern, and start again from the beginning if they wanted to – for SysPal policies the mandated points were fixed though, and did not change after initialization. During pattern setup, the Original participants initialized the grid 0.07 times on average (see Table VII), and all of the SysPal participants also initialized the grid between 0.07 and 0.08 times on average. The differences between the SysPal policies and the Original policy were not statistically significant ($p = 1.0$, $p = 0.55$, and $p = 0.09$ for 1-Point, 2-Point, and 3-Point, respectively, corrected two-tailed unpaired t-test). The Random participants performed initialization (resetting the given random pattern) most number of times on average with 0.12. Its inferiority against Original and all SysPal policies were statistically significant (all $p < 0.001$, corrected two-tailed unpaired t-test). We note that this frequent initialization is the reason why the point usage ratios shown later in Figures 4, 5, and 6 for Random patterns were not evenly distributed.

F. Recall difficulty

Based on the participants' responses to Q2 in Table II, we estimated pattern "recall difficulty" across different policies. Responses were repeatedly collected after the first test, second test, and third test. We only present the third test results as shown in Figure 3 because the results were not too different between the three tests. The majority of the SysPal and Original participants felt their patterns were easy to remember. But relatively more Random participants felt that their patterns are difficult to remember. For the Original and all SysPal policies, the median value was "5" ("very easy"), and for Random it was "4" ("easy"). Again, Original and all SysPal policies demonstrated statistically significant superiority in the self-reported recall difficulty over Random (all $p < 0.001$, unpaired corrected MW U test).

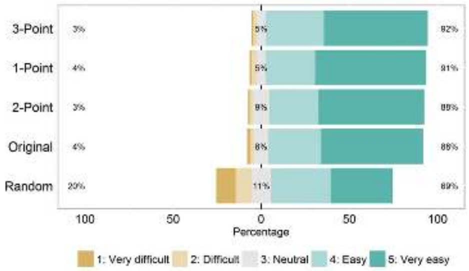


Fig. 3: Results for the third-test recall difficulty.

G. Remembrance techniques

We also asked the participants "Did you use any special technique (e.g., forming a shape) to help you create and remember your pattern?" in Q3 of Table II. 22.48% of the participants (across all policies) used a technique for creating and remembering their pattern. 69.34% of them formed a shape, and about half of those participants used a letter (e.g., 'Z' or 'S') as a shape.

A few participants created a number sequence (like PINs) and remembered that number sequence to help them recall their patterns, explaining that number sequences are easy to remember. A few mentioned that they "practiced several times," and a few mentioned that they "created a pattern that starts from a top left position and ends at a bottom right position."

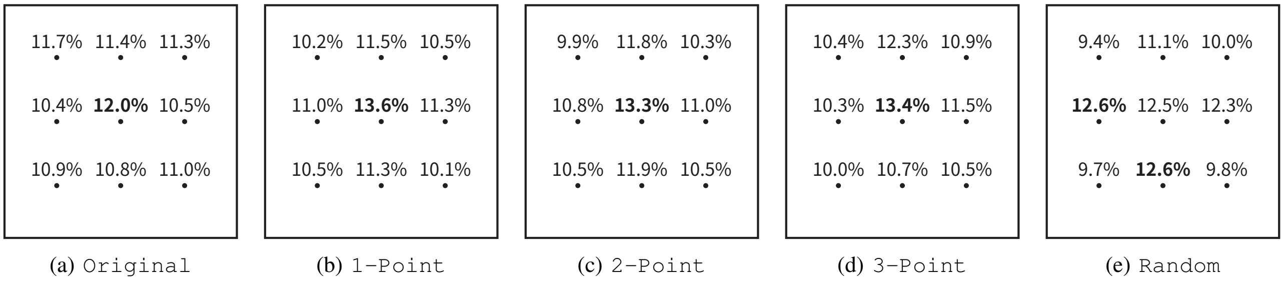
H. 15 minute wait period

After completing the first recall success rate test, we asked the participants to wait for 15 minutes before continuing with the second test. To get a sense of what they were doing during this wait period, we asked an additional question, "Since the last test, what did you spend most of your time doing?"

Most of the participants responded that they were surfing the Internet (36.99%) or watching TV (15.93%). Intriguingly, 12.50% responded that they participated in another survey on MTurk. There were a few participants who responded that they checked emails, read a book, cooked, or ate something, such activities adding up to 16.75% in total.

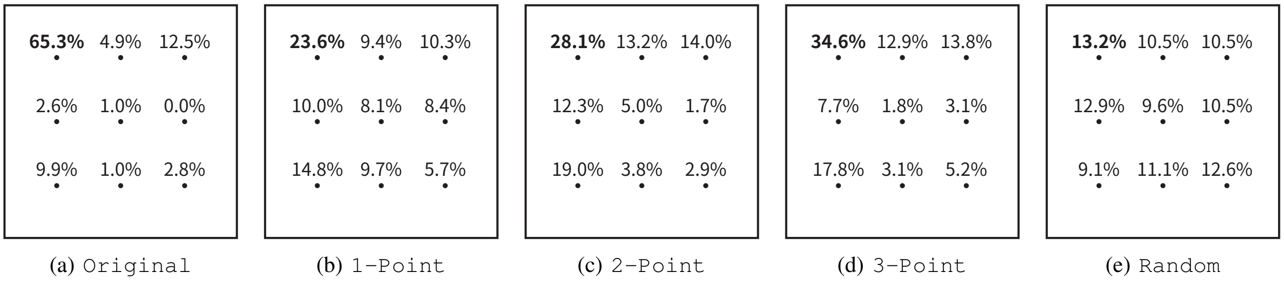
V. Results: security

In this section, we present the key security results for the three SysPal policies, comparing their security against the Original and Random policies.



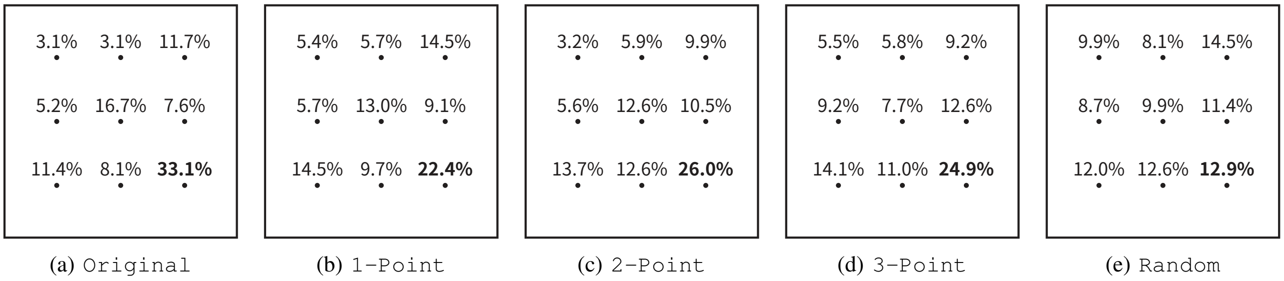
(a) `Original` (b) `1-Point` (c) `2-Point` (d) `3-Point` (e) `Random`

Fig. 4: Usage ratios for each of the 9 points.



(a) `Original` (b) `1-Point` (c) `2-Point` (d) `3-Point` (e) `Random`

Fig. 5: Ratio of each point being used as the starting point.



(a) `Original` (b) `1-Point` (c) `2-Point` (d) `3-Point` (e) `Random`

Fig. 6: Ratio of each point being used as the ending point.

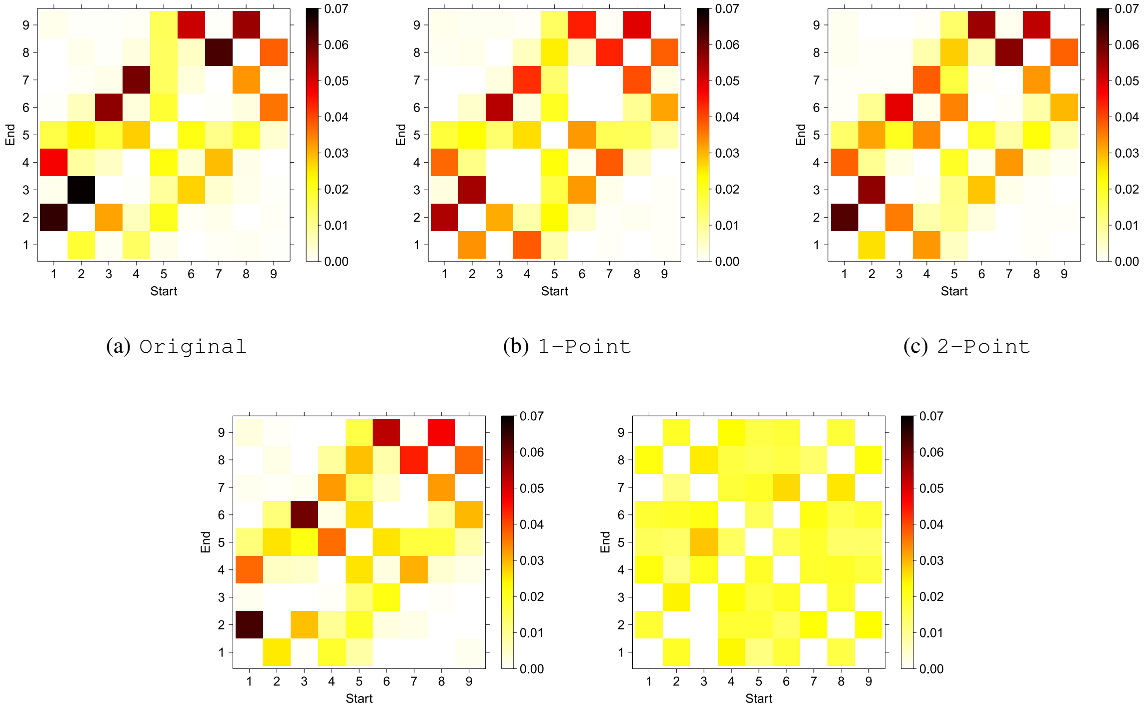
A. *Pattern characteristics*

1) Points used in patterns: Previous studies [22, 24] demonstrated that people tend to select weak patterns that have some common (guessable) characteristics; e.g., people prefer using certain starting points and ending points.

To identify such common characteristics that could potentially weaken the pattern security, we first analyzed the usage frequencies of each of the nine points in the 3×3 grid. Figure 4 shows the usage ratios of all 9 points across all five policies. In all policies except for the `Random` policy, the most frequently used point was 5 – it was more frequently used in all SysPal policies (13.3–13.6%) compared to the `Original` policy (12.0%). Overall, however, the usage frequencies of all points across all policies (including `Random`) were somewhat evenly distributed, and we did not find any statistically significant difference between point usage frequencies (all $p = 1.0$, corrected FET).

We also analyzed the starting and ending point ratios in patterns across all five policies. As shown in Figure 5, the starting points for `Original` patterns were strongly biased towards the upper leftmost point (point "1"), showing a ratio of 65.3% (this percentage is similar to what was reported in [22]). The most popularly used starting points in all of the SysPal policies were also the upper leftmost point, but it was used much less compared to `Original`; those differences between `Original` and all SysPal policies in the distribution of the starting points were statistically significant (all $p < 0.05$, corrected FET). Further, all SysPal policies except `1-Point` showed significantly different distributions of the starting points against `Random` (all $p < 0.001$, corrected FET). `1-Point` did not show statistically significant difference against `Random` ($p = 1.0$, corrected FET).

Figure 6 shows that the usage of the ending points was also biased toward the lower rightmost point (point "9") for all policies except `Random`. However, we did not find any statistically significant difference between all SysPal policies and `Random` (all $p = 1.0$, corrected FET). Further, there was no statistically significant difference between `Original` and all SysPal policies (all $p = 1.0$, corrected FET).



(a) `Original` (b) `1-Point` (c) `2-Point`

(d) `3-Point` (e) `Random`

Fig. 7: Usage ratios for all possible segments.

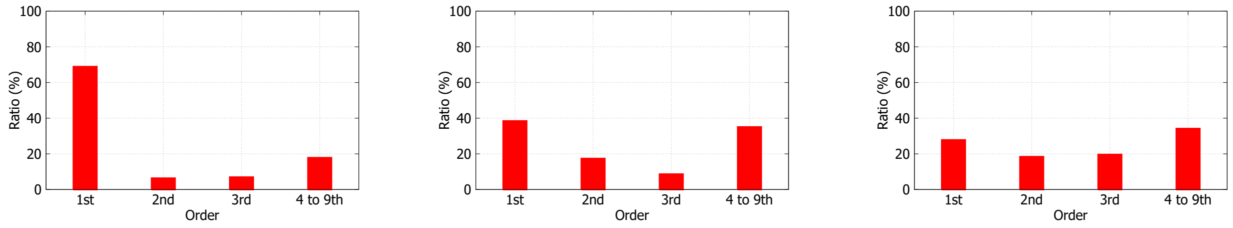
Unexpectedly, the points used in the `Random` patterns were slightly biased toward a few points as shown in Figures 4, 5 and 6. Our statistical tests show that the usage frequencies of the 9 points for the `Random` patterns were not uniformly distributed ($p < 0.001$, chi-square test). We surmise that this may be due to one of the pattern setup instructions that allowed `Random` participants to initialize (reset) given random pattern unlimited number of times until they got a pattern they wanted to use.

2) Segments used: Next, we analyzed the usage ratio of frequently used segments. A segment is a line that connects two points together. Figure 7 shows the usage ratios for all possible segments: darker the color, higher the number of segments used. The total number of segments used in `Original` was 2,596, and the most frequently used segment was (2, 3). It was used 175 times (6.92%). A few diagonal lines, such as (9, 4) and (1, 8), were never used. All SysPal policies showed similar characteristics: the most frequently used segments were (2, 3) and (1, 2), and (i, $i+1$) was more frequently used than ($i+1$, i) for all $i = 1, 2, 4, 5, 7$ and 8, implying that most patterns were drawn from left to right.

We computed Shannon entropy on the usage frequency of segments. The entropy value of usage frequency distribution for segments of `Original` patterns was 5.074. This value was relatively lower than the corresponding entropy values for the SysPal policies, which were 5.185, 5.129, and 5.184, for policies `1-Point`, `2-Point`, and `3-Point`, respectively. This implies that the segments used in the SysPal patterns (for all three policies) were more evenly distributed than the segments used in the `Original` patterns. The segments used in the `Random` patterns had the highest entropy value of 5.753.

3) Positions of mandated points: SysPal policies require users to use randomly assigned points upon selecting a pattern but users can freely choose positions (indexes) in which those points are used. This section analyzes how those mandated points were used. Figure 8 shows the usage frequency ratios of the mandated points used in each of the 9 positions.

The `1-Point` participants used the mandated point mostly in the first position of their patterns (68.9%). 38.5% of the `2-Point` participants used one of the two mandated points in the first position, and 17.4% used one of the mandated points in the second position. The use of the mandated points in different pattern positions for `2-Point` seems to be more evenly distributed than that of `1-Point` but this may be simply because there was one more mandated point that had to be used. Interestingly, the usage frequency ratio for the fourth position was quite high for `2-Point`. This may be because patterns with length 4 were frequently selected by the `2-Point` participants (about 30%), and one of the mandated points could have been used frequently as an ending point in those patterns. The `3-Point` participants used mandated points less in the first position, and used them more in the



(a) 1-Point (b) 2-Point (c) 3-Point

Fig. 8: Proportion of mandated points used in each pattern position. "4 to 9th" represents the accumulated proportion of mandated points used across positions 4 to 9.

TABLE VIII: Comparison of the mean distance of the Mandated group and Normal group (μ: mean, σ: standard deviation).

		Length													
Policy		4		5		6		7		8		9		Total	
		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
2-Point	Normal	1.630	0.181	1.942	0.143	2.271	0.110	2.635	0.097	2.934	0.093	3.310	0.077	2.696	0.625
	Mandated	1.851	0.905	2.519	1.287	3.209	1.533	3.308	1.937	4.789	2.504	4.153	2.698	2.906	1.955
3-Point	Normal	1.695	0.332	1.942	0.248	2.318	0.189	2.667	0.181	3.035	0.131	3.314	0.113	2.720	0.624
	Mandated	1.639	0.332	2.135	0.580	2.393	0.757	2.667	1.089	2.708	1.092	3.546	1.246	2.286	0.985

second and third positions. Across all SysPal policies, there seems to be a tendency to use the mandated points as the starting point of their pattern.

4) Usage of mandated points: Next, we analyzed how mandated points were used upon selecting a SysPal pattern. First, we looked at how often mandated points were used *adjacent* to each other. For example, given 1 and 2 as mandated points, if a pattern is 1–2–3–6, we considered those two mandated points to be adjacently located. If a pattern is 1–4–5–2, those two mandated points were not considered as adjacently located points. For 2-Point, 33.6% of the participants selected patterns that consisted of adjacently located mandated points. Surprisingly, for 3-Point, 82.5% of the participants selected patterns with at least two adjacently located mandated points. Interestingly, 33.0% of the 3-Point patterns had all three mandated points adjacently located to each other.

Second, we analyzed the number of patterns that used the mandated points as the starting point and ending point. 32.8% of the 2-Point participants and 36.4% of the 3-Point participants created their patterns that way. Overall, our results indicate that the participants' pattern selection behaviors, including the way they choose starting and ending points, were significantly affected by the randomly-assigned mandated points.

5) Distance between mandated points and other points: To further analyze the usage behaviors of mandated points, we compared the mean distances between the mandated points and normal points (see Table VIII). For example, given 1 and 2 as mandated points, if a pattern is 1–2–3–6, the distance between the mandated points (i.e., "1" and "2") is 1. If a pattern is 1–4–5–2, the distance between the mandated points is 3. For our analysis, we created two groups, one group called "Normal" that comprises of segments that contain at least one normal point, and another group called "Mandated" that comprises of segments with two mandated points. Since by the definition given above the Mandated group will be empty for 1-Point, we only analyzed policies 2-Point and 3-Point.

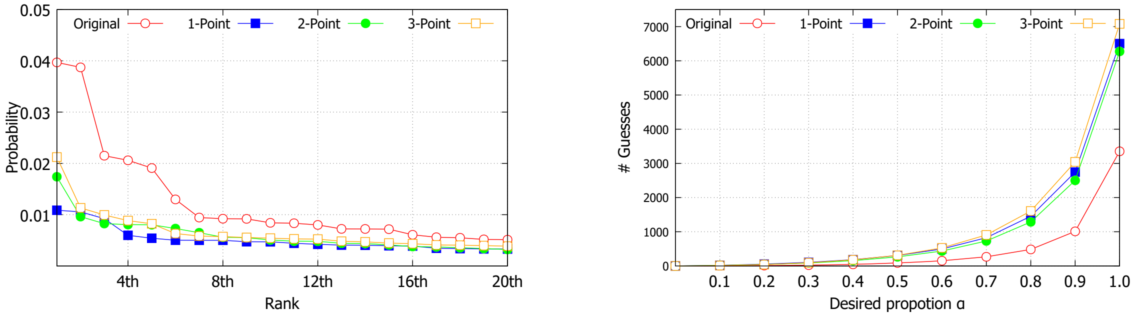
The mean distance of two points in a segment in the Mandated group for 2-Point was 2.906 with a standard deviation of 1.955. This was longer than the mean distance of two points in a segment in the Normal group, which was 2.696 with a standard deviation of 0.625. There was a statistically significant difference between the two groups ($p < 0.001$, corrected two-tailed unpaired t-test).

For policy 3-Point, however, the mean distance of two points in a segment in the Normal group ($\mu = 2.720$ and $\sigma = 0.624$) was longer than that of the Mandated group ($\mu = 2.286$ and $\sigma = 0.985$). But the difference between the two groups was not statistically significant ($p = 0.37$, corrected two-tailed unpaired t-test).

In 32.7% of the 2-Point patterns, and 36.2% of the 3-Point patterns, mandated points were used as starting and ending points at the same time. In 33.6% of the 2-Point patterns, and 33.1% of the 3-Point patterns, all the mandated points were directly connected (i.e., there was no intermediary point in between). Furthermore, in 49.7% of the 3-Point patterns, at least two mandated points were directly connected. It is possible that, given mandated points, many participants simply tried to select a pattern that connects the mandated points with a minimal drawing effort.

B. Guessing entropy

To compare the robustness of the five policies against guessing attacks, we calculated *partial guessing entropy* estimates [8] because some attackers may only be interested in stealing just a fraction of an entire password set. This is a popularly used technique for estimating the average number of trials needed to successfully guess a *fraction* (α) of an entire password set. For $0 \leq \alpha \leq 1$, let $\mu_\alpha = min\left\{j \middle| \sum_{i=1}^{j} p_i \geq \alpha\right\}$



(a) Top 20 Probabilities

(b) Guessing entropy

Fig. 9: Probability estimates for the top 20 patterns and α–guessing entropy.

TABLE IX: Comparison of bits of information with α across all policies.

Policy	α									
	0.1	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*	*1.0*
`Original`	5.04	5.82	6.54	7.19	7.86	8.50	9.20	9.97	11.00	12.71
`1-Point`	7.54	8.19	8.67	9.16	9.67	10.21	10.82	11.57	12.44	13.67
`2-Point`	7.16	7.91	8.40	8.92	9.47	10.02	10.65	11.39	12.30	13.62
`3-Point`	6.95	7.81	8.52	9.12	9.69	10.29	10.96	11.71	12.59	13.79
`Random`	11.20	11.84	12.44	13.02	13.58	14.11	14.60	15.04	15.44	15.81
`Random Patterns` (U_{389112})	18.57	18.57	18.57	18.57	18.57	18.57	18.57	18.57	18.57	18.57
`Real-world 4-digit PINs` [17]	5.19	7.04	8.37	9.38	10.08	10.63	11.08	11.44	11.70	11.83
`Random 4-digit PINs` (U_{10000})	13.29	13.29	13.29	13.29	13.29	13.29	13.29	13.29	13.29	13.29
`Real-world 6-digit PINs`	10.71	13.32	14.03	14.50	14.92	15.36	15.86	16.49	17.14	17.53
`Random 6-digit PINs` ($U_{1000000}$)	19.93	19.93	19.93	19.93	19.93	19.93	19.93	19.93	19.93	19.93

where p_i is the probability of i^{th}element occurring in non-increasing order, and let $\lambda_{\mu_\alpha} = \sum_{i=1}^{\mu_\alpha} p_i$, which is the actual fraction covered. With those notations, partial guessing entropy is defined as $G_\alpha(\chi) = (1 - \lambda_{\mu_\alpha}) \cdot \mu_\alpha + \sum_{i=1}^{\mu_\alpha} i \cdot p_i$ where the traditional guessing entropy is a special case of partial guessing entropy with $\alpha = 1$.

Because our collected set of patterns only represent a small portion of the theoretically possible password space, we employed the 3-gram Markov model to estimate the occurrence likelihood of every possible pattern. A separate Markov model was constructed for each policy. To cover rare n-gram cases, we particularly used the *Laplace smoothing* approximation technique – the frequency of each n-gram is incremented by one. The Markov model is one of most representative probabilistic password models to evaluate the guessability of passwords [19].

The estimated probabilities of occurrence likelihood of patterns are sorted in descending order, and the probabilities of the top 20 patterns are plotted in Figure 9(a). The probability graphs for the SysPal policies are flatter than the graph for the `Original` policy. Compared to the SysPal policies, the pattern distribution for `Orignal` is skewed toward a small number of commonly used patterns. Appendix C shows the top 20 most likely used patterns in the SysPal policies and `Original` policies, respectively.

As described in our attack scenario (see Section II-B), a policy that restricts the number of allowed unlock fail attempts (e.g., Android allows 20 consecutive attempts) could be applied to effectively mitigate real-time, online guessing attacks. For more intuitive comparison of entropy estimates, entropy estimates can be represented in “bits of information.” This conversion can be done as follows:

$$\tilde{G}_\alpha(\chi) = \log\left(\frac{2 \cdot G_\alpha(\chi)}{\lambda_{\mu_\alpha}} - 1\right) + \log\frac{1}{2 - \lambda_{\mu_\alpha}}$$

The converted results are shown in Table IX. As for the real-world 4-digit PINs, we used a PIN dataset consisting of 204,508 PINs that was collected through an iPhone application [17]. As for the real-world 6-digit PINs, we extracted 383,914 6-digit PINs from the popularly known “RockYou” (32.6 million) and “Yahoo” (0.5 million) password datasets. We constructed a 5-gram Markov model with those PINs to estimate the guessing entropy of 6-digit PINs.

As expected, policy `Random` showed the highest guessing entropy estimates in bits of information. All SysPal policies showed higher entropy estimates than `Original` policy. Among the SysPal policies, policy `1-Point` showed a higher guessing entropy estimate than other SysPal policies between $\alpha = 0.1$ and 0.4. Interestingly, policy `3-Point` showed the highest guessing entropy estimates from $\alpha = 0.5$ onwards.

Next, based on those occurrence likelihood probabilities, we computed partial guessing entropy estimates. Our entropy results are plotted in Figure 9(b). As α increases, the differences between `Original` and all SysPal policies increases significantly, clearly demonstrating that SysPal patterns are much more robust against guessing attacks even when α is large. Unlike `Original` policy, all SysPal policies seem to provide higher resistance to guessing attacks than real-world

TABLE X: Comparison of the percentage of cracked patterns across all policies.

	Original	1-Point	2-Point	3-Point	Random
Mean # of guessing attempts	5,472.97	3,803.01	2,993.18	3,740.18	47,445.51
Mean # of guessing attempts (≤ 20)	6.31	10.44	7.29	11.74	0.00
Mean % of cracked patterns (≤ 20)	32.55%	9.97%	9.36%	14.11%	0.00%

4-digit PINs when α is less than 0.3. However, the guessing entropies of the SysPal policies are still significantly lower than that of the real-world 6-digit PINs.

C. Pattern cracking

1) k-fold cross validation: As another metric for evaluating the security of SysPal, we used a pattern cracking technique that we designed using the 3-gram Markov model (see V-B). We calculated the probabilities of all possible patterns, and sorted them in descending order. This list of ordered patterns were used as an organized, smart dictionary to efficiently crack patterns.

To statistically generalize pattern cracking results, we applied the 10-fold cross validation method on each pattern set. That is, the patterns collected for each policy was equally divided into 10 subsets. One of the subsets was used as a test set, and the remaining 9 subsets were used as the training data. We repeated this validation process 10 times, where every subset was used once as a test set. We then averaged the ratios of cracked patterns on those 10 rounds.

2) Cracking results: The cracking results are summarized in Table X. None of the Random patterns were cracked in our experiments, demonstrating their robustness against sorted dictionary-based guessing attacks. In contrast, 32.55% of the Original patterns were successfully cracked within 20 guessing attempts, which was the largest percentage across all policies. Much smaller portion of SysPal patterns were cracked though: 9.36%, 9.97%, and 14.11% for 2-Point, 1-Point, and 3-Point, respectively. All SysPal policies showed statistically significant superiority over Original in resisting our cracking technique (all $p < 0.001$, corrected FET). However, there was no statistically significant difference between all SysPal policies (all $p > 0.9$, corrected FET).

VI. Lab study

As shown in Table IV, the recall success rate for Original and SysPal policies was around 94-99% in all three tests. Those results showed that people can recall SysPal patterns just as well as Original patterns. However, since most people unlock their phones more frequently in real life [13] (and not based on the three artificial recall test durations we experimented with), it is hard for us to make strong claims about the memorability solely based on the first study results.

To address this limitation of the first study, we conducted a separate lab study to collect screen unlock data that would closely resemble real-life usage scenarios. To achieve strong ecological validity, we implemented a screen unlock application that mimics the current Android screen lock pattern mechanism and supports SysPal policies. We asked the participants to install it on their own Android phone, and use it for a day. The goal of the lab study was to measure and analyze the participants' real-life unlock behaviors when SysPal policies are used.

A. Methodology

Instead of using Mechanical Turk, this study was conducted in a laboratory, where the participants (before doing the study) were explained the study purpose, instructions, and how SysPal patterns can be created (using the same examples given to the Mechanical Turk participants). We selected three representative policies to experiment with: policy Original was chosen as the real-world reference, and two SysPal policies 1-Point, 2-Point were chosen as the best performing SysPal policies in the first study that did not show statistically significant difference against policy Original in the recall success rate.

The participants were welcome to participate at anytime between 10am and 10pm. In one day period, we recruited 46 participants from a university campus. Each participant was sequentially assigned to one of the three policies. We followed the methodology of the first study (see Section III), with the modifications explained bellow.

The participants were first briefed on the purpose of the study, and asked to sign a consent form. They were then asked to submit demographics, and download and install our screen unlock application. After installation, they were asked to select a pattern based on the assigned policy, and use this pattern for a day. To ensure that we do not affect their normal phone usage behaviors, we did not give any additional instruction. For those who were already using a screen lock mechanism, we asked them to switch to using our application for a day. An unlock attempt was marked as a "failed" attempt if a participant failed to draw the correct pattern within 5 guesses. A reminder email was sent 24 hours after the study started, notifying the participants to come back and complete a survey (the responses for Q2 and Q3 as described in Table II are presented in Appendix D).

B. Demographics

We recruited a total of 46 Android users during a three-day period. We carefully excluded two responses, one from a participant who used an external storage, and another from a participant who tried to disable our unlock application during the study, leaving us with 44 valid responses. The participants' age ranged from 19 to 45 years. The majority of the participants (62.2%) were graduate students, and about 73.3% were male.

C. Memorability

We first estimated the 24-hour survival rate for each policy based on the number of participants who correctly entered

TABLE XI: Lab study memorability results. Mean time (sec) taken to complete authentication, mean number of unlock attempts made during the 24 hour period (μ: mean, σ: standard deviation), % of participants who survived the study, and % of successful unlock trials across all participants.

Policy	# Participants	# Unlock trials	# Failed	% Survived participants	% Successful unlock trials	Authentication time		# Attempt	
						μ	σ	μ	σ
`Original`	15	873	1	93.3%	99.9%	2.16	2.53	1.07	0.34
`1-Point`	13	730	1	92.3%	99.9%	2.55	3.50	1.18	0.52
`2-Point`	16	965	1	93.8%	99.9%	2.54	4.08	1.11	0.40

their pattern in all unlock trials made (each participant would have tried different number of times to unlock his or her phone depending on their daily usage behaviors) during the 24 hour period. If a participant failed to unlock his or her phone screen within 5 attempts during the 24 hour period, we marked it as a "failed" trial, and stopped the application.

As Table XI shows, the survival rates for all policies were high, ranging between 92.3% and 93.8%. For each policy, only one participant failed to unlock his or her phone within 5 trials. As with our recall success rate results (see Section IV-C), policy `Original` did not show statistically significant superiority in the survival rate against `1-Point` and `2-Point`. In fact, `2-Point` showed a higher survival rate.

For each policy, we also computed the percentage of successful unlock trials (entered a correct pattern within 5 attempts) across all participants (note, each participant tried unlocking his or her phone different number of times). Those successful unlock rates were equally very high at 99.9% across all policies. Again, there was no statistically significant difference between the three policies (all $p = 1.0$, corrected FET).

D. Authentication time and the number of attempts made

As shown in Table XI, in contrast to the result of the large-scale online study (see Table V), `Original` policy (2.16 seconds) outperformed `1-Point` (2.55 seconds) and `2-Point` (2.54 seconds) with a statistically significant difference in authentication time (all $p < 0.001$, corrected two-tailed unpaired t-test). However, we did not find statistically significant difference between `1-Point` and `2-Point` ($p = 1.0$, corrected two-tailed unpaired t-test).

We also compared the mean number of attempts made for each unlock trial. The `Original` policy had the lowest mean value at 1.07 times compared with 1.18 times for `1-Point` and 1.11 times for `2-Point`. `Original` showed statistically significant superiority against `1-Point` ($p < 0.001$, corrected two-tailed unpaired t-test) but not against `2-Point` ($p = 0.2$, corrected two-tailed unpaired t-test). Interestingly, `2-Point` outperformed `1-Point` with a statistically significant difference ($p < 0.001$, corrected two-tailed unpaired t-test).

E. Pattern setup time

We also analyzed pattern setup time and the number of pattern initialization (grid reset) performed upon setting up a pattern (see Table XII). The `Original` participants took 12.02 seconds on average to set up a pattern, showing statistically significant superiority against the `2-Point` participants, who took 32.27 seconds on average ($p < 0.01$, unpaired corrected MW U test); the `1-Point` participants took 27.14 seconds on average, again, showing inferiority against `Original` participants ($p = 0.06$, corrected unpaired MW U test). The difference in average setup time between `1-Point` and `2-Point` was not statistically significant ($p = 0.58$, corrected unpaired MW U test).

TABLE XII: Lab study setup time results. Mean time (sec) taken to set up a pattern, and mean number of pattern initialization (grid reset) performed during pattern setup (μ: mean, σ: standard deviation).

Policy	*Setup time*		*# Initialization*	
	μ	σ	μ	σ
`Original`	12.02	5.16	0.20	0.54
`1-Point`	27.14	30.93	0.15	0.36
`2-Point`	32.27	31.13	0.38	0.70

As explained in Section IV-E, we allowed the participants to initialize (grid reset) the grid unlimited number of times during pattern setup, and start again from the beginning if they wanted to.

We did not find any statistically significant differences in the mean number of initialization performed between the three policies, which ranged between 0.15–0.38 times (all $p > 0.6$, corrected unpaired MW U test).

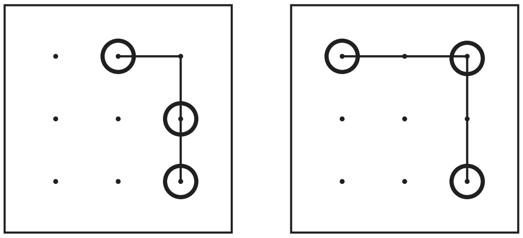
VII. Discussion

We discuss our findings with respect to the hypotheses we set up in Section III.

A. Security improvements

The first hypothesis states that "the security of SysPal patterns strengthens with the increase in the number of mandated points." As shown in Table IX and X, however, increasing the number of mandated points from one to three did not improve the security of SysPal patterns. We did, however, demonstrate the superiority of all SysPal policies over `Original` policy in both the guessing entropy estimates and the percentage of patterns cracked by the 3-gram Markov model.

Unlike our expectations, policies `1-Point` and `2-Point` showed a lower percentage of cracked patterns (lower by 4.14% and 4.75%, respectively) compared to policy `3-Point`. Those differences, however, were not statistically significant. Hence, based on our security analysis results, we cannot accept the first hypothesis.



(a) 2–3–6–9 (b) 1–2–3–6–9

Fig. 10: Most frequently cracked patterns in `3-Point`.

The different characteristics of `2-Point` and `3-Point` patterns provide some possible explanations about the weaknesses found in `3-Point` patterns. As shown in Table VIII, the mean distance between mandated points (2.286) was *less* than the mean distance between normal points (2.720) in `3-Point`, while the mean distance between mandated points (2.906) was *greater* than the mean distance between normal points (2.696) in `2-Point`. This indicates that when users are given more than two mandated points, they may have a tendency to directly join the mandated points together – possibly because the chance of creating a pattern with length 4 or longer, by simply connecting three mandated points directly, is quite high (which is impossible when there are just two mandated points). This could have encouraged the `3-Point` participants to merely connect three mandated points directly to create a pattern that conforms to the pattern length requirement. We surmise that such tendency may have introduced some pattern selection bias, and weakened the security of `3-Point` patterns. Figure 10 demonstrates two most frequently cracked patterns for `3-Point` (each being cracked 5 times).

B. Recall success rate and memorability effects

The second hypothesis states that "the memorability of SysPal decreases with the increase in the number of mandated points." As shown in Table III and IV, however, the effects of increasing the number of mandated points is not clear. Both the recall success rate results in Section IV-C and memorability results in Section VI-C did not show statistically significant difference between all SysPal policies. Hence, we do not have sufficient evidence to accept the second hypothesis. Intriguingly, all SysPal policies did not show statistically significant inferiority in recall success rate and memorability against the `Original` policy. Overall, our results show that the SysPal patterns have the potential to be just as memorable as the `Original` patterns.

C. Replacing the original Android policy

The third hypothesis states that "a SysPal policy that shows no statistically significant difference in memorability against the original Android patterns has *better security* than those original patterns." As shown in Section IV, none of the SysPal policies showed statistically significant inferiority against the `Original` policy in the third-test recall success rates (70.09–73.68% vs. 72.40%). Our lab study under real-life unlock scenarios also showed that there is no statistically significant difference in memorability between those policies.

In terms of security (see Section V), all SysPal policies significantly outperformed the `Original` pattern policy in both the partial guessing entropy estimates (6.95–7.54 vs. 5.04 when $\alpha = 0.1$), and percentage of cracked patterns (9.36%–14.11% vs. 32.55%). Hence, our results accept the third hypothesis.

In contrast to the online study results (see Table V, VI and VII), the lab study results showed superiority of `Original` over `1-Point` and `2-Point` with respect to authentication time, number of authentication attempts, and setup time (see Table XI and XII). Those results were somewhat expected though, as most of the participants were already familiar with the current Android policy. The SysPal policies, on the other hand, are new policies that the participants had to learn and try for the first time. We believe the usability of the SysPal policies can improve over time as people become more familiar with SysPal.

Based on our analysis, SysPal policies can potentially replace the current Android policy without compromising too much usability. In fact, SysPal is highly compatible with the existing Android lock scheme, and only requires small software level upgrade to fully support it. However, our recommendation is to use `1-Point` or `2-Point` (that were more secure than `3-Point`) to have extra robustness against guessing attacks.

D. Usability of random patterns

Random patterns obviously had the highest partial guessing entropy estimates and the lowest percentage of cracked patterns (0%). However, as expected, the third-test recall success rate was significantly lower (at 50.60%) than all other policies. This demonstrates a clear memorability limitation in adopting purely random patterns. To the best of our knowledge, we are the first group to analyze the usability of random patterns.

E. Implications on graphical passwords

Previous studies [22, 24, 30] have shown that the actual password spaces of user-chosen graphical password schemes are much smaller than their theoretical password spaces. Just like textual passwords, people choose easy-to-remember graphical passwords that are vulnerable against guessing attacks. However, unlike textual passwords, complexity policies have not yet been implemented nor thoroughly evaluated in the context of graphical passwords.

The SysPal policies were designed specifically to improve the security of graphical patterns by artificially adding and mandating some randomness to the pattern selection process. Our evaluation results indicate that such policies can significantly enhance the security of patterns without compromising too much memorability. We speculatively generalize those findings, and indicate that such policies may be effective in other types of graphical password schemes as well. For instance, with the picture gesture-based authentication system [30], a gesture, randomly selected from the three "tap," "line," and "circle" gestures, can be mandated to help users deviate from their normal gesture selection behaviors, and choose stronger passwords.

As part of future work, we plan to generalize SysPal policies, apply the concepts to other types of graphical password schemes, and evaluate their effectiveness.

VIII. Related work

Graphical passwords have been studied intensively in academia [7] but they have not been so popular in real-world systems. In 2008, however, a graphical pattern-based password scheme (modified from Pass-Go [23]) was deployed on Android devices as a screen unlock mechanism. It quickly became the most popularly used screen lock mechanism on Android [25].

However, Android patterns are vulnerable to smudge attacks [2, 5, 9], accelerometer-based side channel attacks [6], shoulder surfing attacks [26, 29], and guessing attacks [4, 22, 24]. Uellenbeck et al. [24] conducted a large-scale study, analyzing common (biased) user pattern selection behaviors that could weaken pattern security. For example, the majority of the participants used the upper leftmost point as the starting point. Andriotis et al. [2] also identified such biased behaviors through an online survey. Song et al. [22] confirmed those findings through the analysis of real-world patterns collected through an Android application. Our research was motivated by the practical challenge of changing users' such biased pattern selection behaviors.

To improve the pattern security, Andriotis et al. [1] proposed the use of a pattern security meter based on features such as pattern length, overlapping points, and knight moves, and showed that 23.3% of the participants changed their initially selected pattern when their meter was present. Song et al. [22] also analyzed the effects of a pattern strength meter by comparing two independent user groups, one group with a meter in place, and another group without a meter in place. Their guessing entropy results showed that their pattern strength meter is indeed effective in improving the pattern security against guessing attacks. However, even with the meter being present, they still identified biased pattern selection behaviors. Aviv et al. [4] suggested the use of a bigger grid (4×4) layout. Again, their bigger grid layout did not really affect users' pattern selection behaviors. Even when the 4×4 grid was used, 19% of users' patterns were cracked by guessing attacks.

Being mindful of such limitations, SysPal was designed to help users think and behave differently while selecting a pattern by randomly assigning a few points. It was also our goal to design very *practical* Android security policies that can significantly enhance pattern security with just small compromise in usability. In any given system, this is a challenging usable security goal to achieve.

To strengthen the security of *textual* passwords, the effects of password composition policies (e.g., restricting number of digits, lowercase, uppercase, and symbols) have been intensively studied [11, 16, 18, 20]. User persuasion techniques for graphical passwords [10, 21] and textual passwords [12, 15] were introduced to help users move away from their normal password selection behaviors. Chiasson et al. [10] proposed a persuasive cued click-point technique that forces users to choose points from a series of randomly chosen areas in a given picture. Siadati et al. [21] proposed a persuasive technique to suggest a random starting point that a user can *optionally* use. In contrast, SysPal policies *mandate* the use of given random points at any position. Their memorability evaluation was conducted with a small number of students in a lab environment, experimenting with one short recall interval of 20 minutes. They used a web user interface rather than conducting the study on the actual Android screen lock user interface, which could have affected participants' behavior in selecting and using patterns, and the security and usability results. In contrast, to achieve strong ecological validity, we developed an actual Android screen lock app (with SysPal policies) that the participants installed on their own smartphones, and recorded and studied the participants' *real-life unlock behaviors* for a day. We also studied several different policies by varying the number of mandated points to generalize our observations, and find an optimal SysPal policy.

Such persuasive techniques and SysPal are designed based on a common goal to persuade users to move away from their normal selection behaviors, and introduce more randomness as a result. Nevertheless, SysPal is a new, and fully evaluated Android pattern selection policy that mandates the use of a few points, but at the same time allows the freedom of using those mandated points at any position. It was designed to incrementally improve the security of an existing, popularly used graphical password scheme. Those characteristics clearly differentiate SysPal from previous persuasive techniques.

IX. Conclusions and future work

We proposed a novel system-guided pattern scheme for Android called SysPal, which mandates the use of a few randomly selected points upon choosing a pattern. The idea is to help users choose more secure patterns by deviating them from their normal, biased pattern selection behaviors.

Our large-scale online study showed that SysPal patterns and original Android patterns have similar pattern recall success rate. Our lab study, which was designed to closely resemble real-life unlock scenarios, again showed that SysPal patterns are just as memorable as original Android patterns. Yet, SysPal patterns are much more robust against guessing attacks, and have higher entropy values. Hence, it is our recommendation to replace the existing Android policy with the SysPal policy that mandates either one or two points. SysPal policies may also be used as guidelines to help design similar security policies for other graphical password schemes, artificially injecting some randomness to passwords.

As part of future work, we plan to explore other pattern features like "pattern lengths," and design other types of password composition policies for pattern-based authentication.

ACKNOWLEDGMENT

This work was supported by NRFK (No. 2014R1A1A1003707) and ITRC (IITP-2016-R0992-16-1006). The authors would like to thank all the anonymous reviewers for their valuable feedback. Note that Hyoungshick Kim is the corresponding author.

REFERENCES

[1] P. Andriotis, T. Tryfonas, and G. Oikonomou, "Complexity Metrics and User Strength Perceptions of the Pattern-Lock Graphical Authentication Method," in *Proceedings of the 2nd Conference on Human Aspects of Information Security, Privacy, and Trust*, 2014.

[2] P. Andriotis, T. Tryfonas, G. Oikonomou, and C. Yildiz, "A pilot study on the security of pattern screen-lock methods and soft side channel attacks," in *Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2013.

[3] R. C. Atkinson and R. M. Shiffrin, "Human memory: A proposed system and its control processes," *The psychology of learning and motivation*, vol. 2, 1968.

[4] A. J. Aviv, D. Budzitowski, and R. Kuber, "Is Bigger Better? Comparing User-Generated Passwords on 3x3 vs. 4x4 Grid Sizes for Android's Pattern Unlock," in *Proceedings of the 31st ACM Annual Computer Security Applications Conference*, 2015.

[5] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge Attacks on Smartphone Touch Screens," in *Proceedings of the 4th USENIX Workshop on Offensive Technologies*, 2010.

[6] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith, "Practicality of accelerometer side channels on smartphones," in *Proceedings of the 28th ACM Annual Computer Security Applications Conference*, 2012.

[7] R. Biddle, S. Chiasson, and P. Van Oorschot, "Graphical passwords: Learning from the first twelve years," *ACM Computing Surveys*, vol. 44, no. 4, 2012.

[8] J. Bonneau, "The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords," in *Proceedings of the 33rd IEEE Symposium on Security and Privacy*, 2012.

[9] S. Cha, S. Kwag, H. Kim, and J. H. Huh, "Boosting the Guessing Attack Performance on Android Lock Patterns with Smudge Attacks," in *Proceedings of the 12nd ACM Asia Conference on Computer and Communications Security*, 2017.

[10] S. Chiasson, A. Forget, R. Biddle, and P. C. van Oorschot, "Influencing users towards better passwords: persuasive cued click-points," in *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction-Volume 1*, 2008.

[11] D. Florêncio and C. Herley, "Where do security policies come from?" in *Proceedings of the 6th ACM Symposium on Usable Privacy and Security*, 2010.

[12] A. Forget, S. Chiasson, P. C. van Oorschot, and R. Biddle, "Improving text passwords through persuasion," in *Proceedings of the 4th ACM Symposium on Usable privacy and security*, 2008.

[13] M. Harbach, E. von Zezschwitz, A. Fichtner, A. De Luca, and M. Smith, "It's a hard lock life: A field study of smartphone (un) locking behavior and risk perception," in *Proceedings of the 10th USENIX Symposium On Usable Privacy and Security*, 2014.

[14] D. Hintze, R. D. Findling, M. Muaaz, S. Scholz, and R. Mayrhofer, "Diversity in locked and unlocked mobile device usage," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, 2014.

[15] J. H. Huh, S. Oh, H. Kim, K. Beznosov, A. Mohan, and S. R. Rajagopalan, "Surpass: System-initiated User-replaceable Passwords," in *Proceedings of the 22nd ACM Conference on Computer and Communications Security*, 2015.

[16] P. G. Inglesant and M. A. Sasse, "The true cost of unusable password policies: password use in the wild," in *Proceedings of the 28th ACM Conference on Human Factors in Computing Systems*, 2010.

[17] H. Kim and J. H. Huh, "PIN selection policies: Are they really effective?" *Computers & Security*, vol. 31, no. 4, 2012.

[18] S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman, "Of passwords and people: measuring the effect of password-composition policies," in *Proceedings of the 29th ACM Conference on Human Factors in Computing Systems*, 2011.

[19] J. Ma, W. Yang, M. Luo, and N. Li, "A Study of Probabilistic Password Models," in *Proceedings of the 35th IEEE Symposium on Security and Privacy*, 2014.

[20] R. Shay, S. Komanduri, P. G. Kelley, P. G. Leon, M. L. Mazurek, L. Bauer, N. Christin, and L. F. Cranor, "Encountering stronger password requirements: user attitudes and behaviors," in *Proceedings of the 6th ACM Symposium on Usable Privacy and Security*, 2010.

[21] H. Siadati, P. Gupta, S. Smith, N. Memon, and M. Ahamad, "Fortifying Android Patterns using Persuasive Security Framework," in *Proceedings of the 9th ACM Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2015.

[22] Y. Song, G. Cho, S. Oh, H. Kim, and J. H. Huh, "On the Effectiveness of Pattern Lock Strength Meters: Measuring the Strength of Real World Pattern Locks," in *Proceedings of the 33rd ACM Conference on Human Factors in Computing Systems*, 2015.

[23] H. Tao and C. Adams, "Pass-Go: A Proposal to Improve the Usability of Graphical Passwords," *International Journal of Network Security*, vol. 7, no. 2, 2008.

[24] S. Uellenbeck, M. Dürmuth, C. Wolf, and T. Holz, "Quantifying the security of graphical passwords: the case of android unlock patterns," in *Proceedings of the 20th ACM Conference on Computer and Communications Security*, 2013.

[25] D. Van Bruggen, S. Liu, M. Kajzer, A. Striegel, C. R. Crowell, and J. D'Arcy, "Modifying smartphone user locking behavior," in *Proceedings of the 9th ACM Symposium on Usable Privacy and Security*, 2013.

[26] E. von Zezschwitz, A. De Luca, P. Janssen, and H. Hussmann, "Easy to Draw, but Hard to Trace?: On the Observability of Grid-based (Un)Lock Patterns," in *Proceedings of the 33rd ACM Conference on Human Factors in Computing Systems*, 2015.

[27] E. von Zezschwitz, P. Dunphy, and A. De Luca, "Patterns in the Wild: A Field Study of the Usability of Pattern and Pin-based Authentication on Mobile Devices," in *Proceedings of the 15th ACM International Conference on Human-Computer Interaction with Mobile Devices and Services*, 2013.

[28] C. Winkler, J. Gugenheimer, A. De Luca, G. Haas, P. Speidel, D. Dobbelstein, and E. Rukzio, "Glass unlock: enhancing security of smartphone unlocking through leveraging a private near-eye display," in *Proceedings of the 33rd ACM Conference on Human Factors in Computing Systems*, 2015.

[29] N. H. Zakaria, D. Griffiths, S. Brostoff, and J. Yan, "Shoulder Surfing Defence for Recall-based Graphical Passwords," in *Proceedings of the 7th ACM Symposium on Usable Privacy and Security*, 2011.

[30] Z. Zhao, G.-J. Ahn, J.-J. Seo, and H. Hu, "On the security of picture gesture authentication," in *Proceedings of the 22nd USENIX Security Symposium*, 2013.

APPENDIX A
LARGE-SCALE STUDY DEMOGRAPHICS

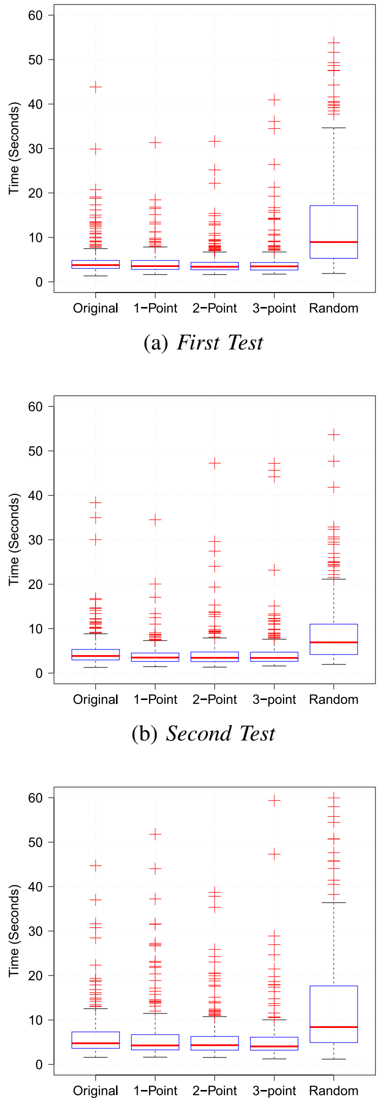
The demographics of the first large-scale study with 1,717 participants are presented in Table XIII.

TABLE XIII: The demographics of the first study participants.

Gender	
Female	833 (48.51%)
Male	878 (51.14%)
No answer	6 (0.34%)
Age group	
18–29	1,059 (61.68%)
30–39	500 (29.12%)
40–49	131 (7.63%)
50–59	18 (1.05%)
60 and over	6 (0.35%)
No answer	3 (0.17%)
Education	
Less than high school	8 (0.47%)
High school	632 (36.81%)
University	928 (54.05%)
Masters	113 (6.58%)
Doctoral	11 (0.64%)
Professional	19 (1.10%)
No answer	6 (0.35%)
Ethnicity	
African American	175 (10.19%)
Asian	118 (6.87%)
White	1,225 (71.35%)
Hispanic	134 (7.80%)
Other	42 (2.45%)
No answer	23 (1.34%)

APPENDIX B
VISUALIZATION OF AUTHENTICATION TIME FOR THE ONLINE USER STUDY

Figure 11 visually compares the authentication time results between all the policies based on the results collected through the online study. All the policies except for `Random` showed similar authentication time.



(a) *First Test*

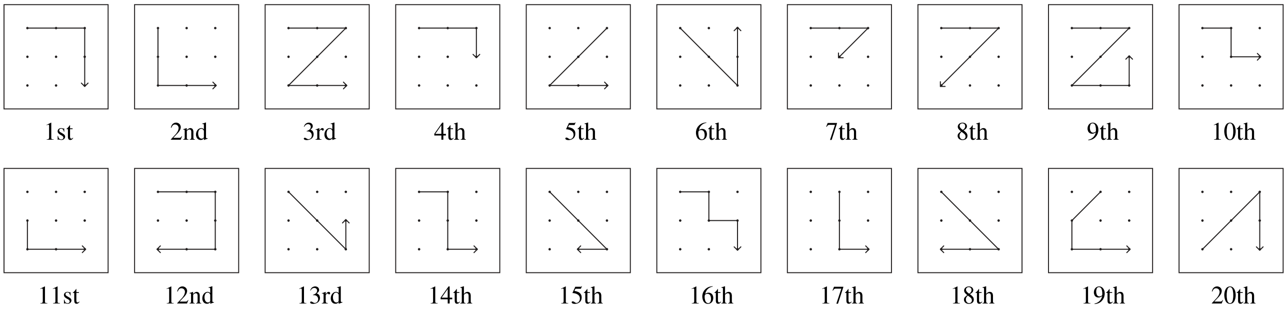
(b) *Second Test*

(c) *Third Test*

Fig. 11: Authentication time in the large-scale online user study.

APPENDIX C
TOP 20 MOST LIKELY USED PATTERNS IN THE SYSPAL AND `ORIGINAL` POLICIES

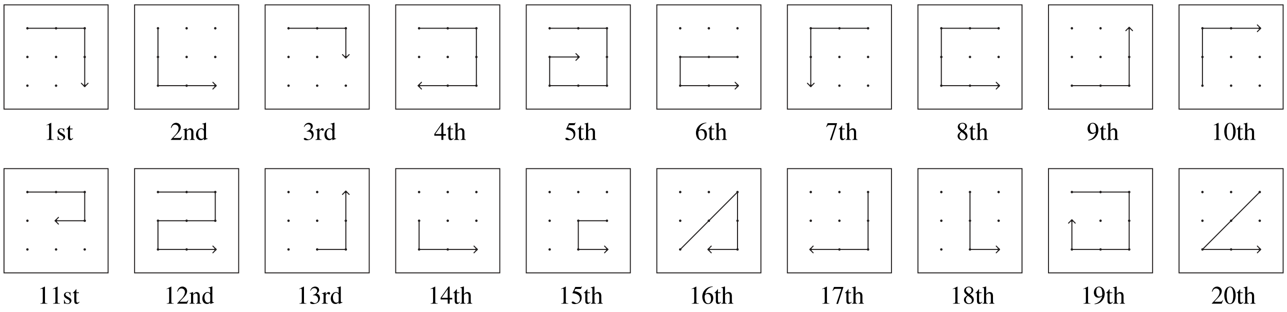
We used the 3-gram Markov model to find the top 20 most likely used patterns in the SysPal policies and `Original`



1st 2nd 3rd 4th 5th 6th 7th 8th 9th 10th

11st 12nd 13rd 14th 15th 16th 17th 18th 19th 20th

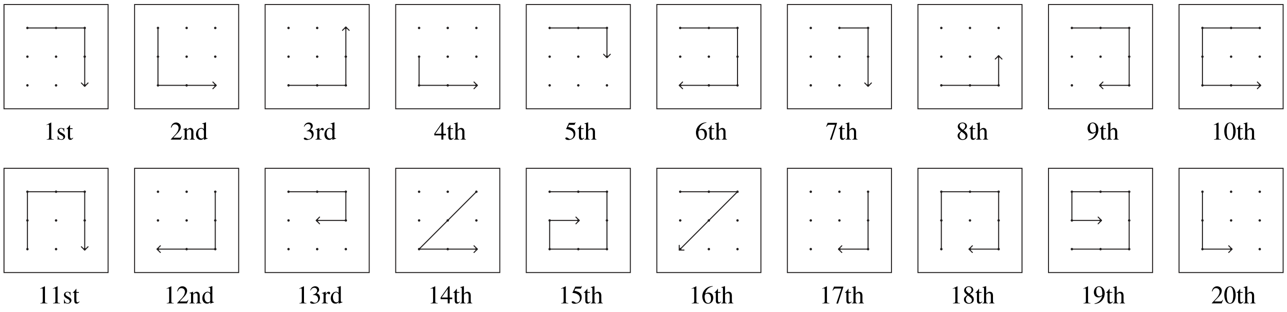
Fig. 12: Top 20 most likely used patterns in `Original`.



1st 2nd 3rd 4th 5th 6th 7th 8th 9th 10th

11st 12nd 13rd 14th 15th 16th 17th 18th 19th 20th

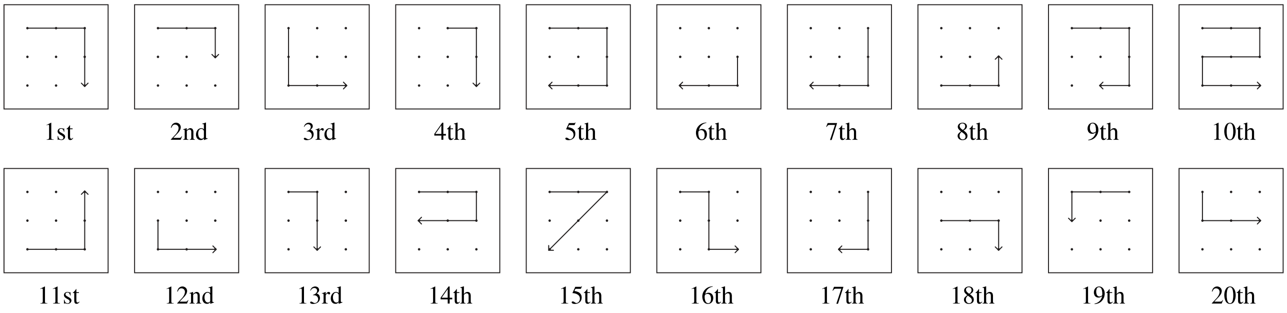
Fig. 13: Top 20 most likely used patterns in `1-Point`.



1st 2nd 3rd 4th 5th 6th 7th 8th 9th 10th

11st 12nd 13rd 14th 15th 16th 17th 18th 19th 20th

Fig. 14: Top 20 most likely used patterns in `2-Point`.



1st 2nd 3rd 4th 5th 6th 7th 8th 9th 10th

11st 12nd 13rd 14th 15th 16th 17th 18th 19th 20th

Fig. 15: Top 20 most likely used patterns in `3-Point`.

policies, respectively. Figure 12, 13, 14 and 15 show the results.

These results show that even when SysPal policies are in place, users still tend to choose patterns that are somewhat easy to draw.

Among the top 20 most likely used patterns, we analyzed the number of patterns that would commonly exist in given two policies. This result is presented in Table XIV. Between 2-Point and 1-Point, over 55% of the top 20 patterns overlapped; between 2-Point and 3-Point, over 60% of such patterns overlapped. Between all the SysPal polices and Original, however, less than 40% of such patterns overlapped.

TABLE XIV: The number of overlapped patterns among each policy.

	Original	1-Point	2-Point	3-Point
Original	-	**7 (35%)**	**7 (35%)**	**9 (40%)**
1-Point	**7 (35%)**	-	11 (55%)	8 (40%)
2-Point	**7 (35%)**	11 (55%)	-	12 (60%)
3-Point	**9 (40%)**	8 (40%)	12 (60%)	-

APPENDIX D
RESPONSES OF PARTICIPANTS IN THE LAB STUDY

After participating in the lab study for a day, participants were asked several survey questions. First, we asked them "How difficult was it for you to remember your pattern?" (Q2 in Table II). Most of them answered that it was easy to set up their patterns (93.3% for Original, 92.3% for 1-Point, 81.3% for 2-Point, respectively).

We also asked Q3 in Table II. 16 out of the 44 participants answered that they used shapes that are "easy to generate" or "easy to draw." Such results were similar to those that were observed from the online study (see Section IV-G).