GraphNeighbors: Hampering Shoulder-Surfing Attacks on Smartphones

Irfan Altiok, Sebastian Uellenbeck, Thorsten Holz

Horst Görtz Institute for IT-Security
Ruhr-University Bochum
Universitätsstrasse 150
44810 Bochum
irfan.altiok@rub.de
sebastian.uellenbeck@rub.de
thorsten.holz@rub.de

Abstract: Today, smartphones are widely used and they already have a growing market share of more than 70 % according to recent studies. These devices often contain sensitive data like contacts, pictures, or even passwords that can easily be accessed by an attacker if the phone is not locked. Since they are mobile and used as everyday gadgets, they are susceptible to get lost or stolen. Hence, access control mechanisms such as user authentication are required to prevent the data from being accessed by an attacker. However, commonly used authentication mechanisms like PINs, passwords, and Android Unlock Patterns suffer from the same weakness: they are all vulnerable against different kinds of attacks, most notably shoulder-surfing. A promising strategy to prevent shoulder-surfing is to only enter a derivation of the secret during the authentication phase.

In this paper, we present a novel authentication mechanism based on the concept of *graphical neighbors* to hamper shoulder-surfing attacks. Results of a usability evaluation with 100 participants show that our implementation called GRAPHNEIGHBORS is applicable in comparison to commonly used authentication mechanisms.

1 Introduction

Smartphones are among the most popular gadgets available on the market today. According to recent studies, their popularity is expected to grow even more in the near future. Since smartphones are not only used for calling but also for taking pictures, sending text messages, surfing the web, or online banking, they contain a lot of sensitive and private data like contact information, personal messages, or even passwords. Due to their mobility, they are carried around by users and consequently, they are an interesting target for attackers, who can easily access all sensitive data if the smartphone is not locked. Hence, smartphones offer different authentication mechanisms like passwords, PINs, or *Android Unlock Patterns*. Since passwords and PINs are cumbersome to enter into the device, alternative solutions are needed. Several studies showed that visual information like pictures are easier to recall than textual information [SCH70] and thus Android Un-

lock Patterns seem to be the solution. However, it also has been shown that the entropy of user-chosen Android Unlock Patterns is rather low [UDWH13]. In the same way, all three methods suffer from the same weakness: they are vulnerable against shoulder-surfing attacks [TOH06]. In this paper, we present a concept and an implementation of GRAPHNEIGHBORS, that counters plain shoulder-surfing attacks while serving usability as well.

Ethical Considerations

As part of our work, we asked 100 people to authenticate using the developed prototypes. All users were informed before participating that they were to take part in a scientific study and that their data was used to analyze the properties of an authentication system. All data collected during our study was used in an anonymized way so that there is no link between collected data and individual participants. Our institute does not fall under the jurisdiction of an IRB or similar ethics committee.

Paper Outline

The rest of this paper is structured as follows. In Section 2, we discuss related work in particular on shoulder-surfing resilient user authentication. We explain our approach in detail and discuss our prototype implementation in Section 3. In Section 4, we present the results of a user survey with 100 participants with focus on the usability of GRAPHNEIGHBORS. We introduce four attacker scenarios and argue how GRAPHNEIGHBORS performs in contrast to legacy authentication approaches like PIN, password, and Android Unlock Patterns in Section 5. Finally, we conclude this paper in Section 6 with a discussion of the results and potential future research questions.

2 Related Work

In recent years, a lot of work has been published about user authentication on mobile devices but only a few aims at hampering shoulder-surfing attacks.

Tan et al. proposed a spy-resistant keyboard that should allow for more secure password entry on public touch screens [TKC05]. Their keyboard is composed of 42 character tiles, two inter-actor tiles, a feedback text box, a backspace button, and an enter button. Each tile randomly contains a lower letter, an upper letter, and a digit or a symbol. Instead of having a fixed shift state, each tile has a random assigned shift state defined by a red line under one of the three characters. By pressing one of the inter-actor tiles, the user can change the shift state of all tiles simultaneously. Having located the correct character as well as the appropriate shift state, he has to drag an inter-actor tile on the character tile to select it. While dragging to the correct tile, all shift states disappear. Balzarotti et al. showed that this schema can easily be broken by a single shoulder-surfing session since

the password is always entered in clear [BCV08]. GRAPHNEIGHBORS improves on this since the login session can be observed several times without revealing the secret.

Roth et al. presented a PIN entry method to mitigate shoulder surfing [RRF04]. On a keypad, they colored the numbers into black and white groups and the user has to select the group his PIN's digit is part of in a round-based fashion. A shoulder surfer would only obtain a subset of the potential PIN, but she could reconstruct the PIN with an intersection analysis if she would be able to watch the entry process several times. In contrast to our approach, the authors assume that the attacker's resources are bounded by the cognitive capabilities of a human. Again, this only holds if the attacker is not allowed to take notes or record the authentication procedure, for example with a (smartphone) camera [BCV08, MVG+11]. Otherwise, she could intersect the chosen subsets to obtain the secret PIN within a few rounds. GRAPHNEIGHBORS resists such attacks.

S3PAS is a system implemented by Zhao and Li to prevent shoulder-surfing attacks [ZL07]. Here the secret is a list of three characters where each item forms a virtual and invisible triangle in a 10×10 grid of characters. To authenticate, the user has to touch inside the triangle in a row-based fashion. Again, this scheme is not secure against sophisticated shoulder-surfing attacks where the attacker records a few sessions [BCV08, MVG+11]. By intersecting the possible password candidates she can afterwards obtain the secret.

3 Methodology and Implementation

Currently, Android offers four mechanisms to unlock a smartphone. The most secure in terms of theoretical password space but also least usable method are passwords. The user can choose an arbitrary string containing digits (0-9), letters (a-zA-Z), and special characters (e. g., !,?,;,:) to form a login secret on a hardware or a software keyboard. There is only one constraint: The login secret has to contain at least 4 characters and at most 16. Although this results in a total password space of about 2^{107}, most users do not want to utilize such a cumbersome approach since entering a secret on a softkeyboard is error-prone due to small buttons and also takes some time.

The PIN login mechanism is a special case of the aforementioned passwords. Here, the user has a limited character set of only digits with at least 4 and at most 16 digits that leads to approximately 2^{53} different secrets. Since the buttons to enter the secret are much larger in comparison to the softkeyboard used for entering password, this authentication method is faster and less error-prone.

The third alternative are Android Unlock Patterns, a simplified variant of the Pass-Go scheme [TA08] to increase usability and to adapt for the small screens found on typical devices running Android. It uses nine points arranged in a 3×3 grid and the user needs to select a path through these points according to the following rules:

1. At least four points must be chosen,
2. no point can be used twice,

3. only straight lines are allowed, and
4. one cannot jump over points not visited before.

Since the authentication can be performed very quickly, this method is a powerful replacement for password and PIN. However, all three mechanisms suffer from the very same weakness: they are susceptible to shoulder-surfing attacks. Besides this, Aviv et al. have shown that the secret can often be obtained by an attacker through analyzing the oily residue left by the user's touch [AGM+10] (so called *smudge attacks*).

The last authentication mechanism is Android's Face Login. To make use of this approach, the user has to take a picture of his face and afterwards look into the camera in order to authenticate. On the one hand, this leads to resistance against classic shoulder-surfing since the secret cannot be easily eavesdropped by an attacker. On the other hand, it has been shown that an attacker only needs a picture of the smartphone's owner to authenticate. A countermeasure has been implemented in Android 4.1 where the user has to blink in order to login. Albeit, we found that even this can easily be fooled by using two pictures (one with open eyes and another one with closed eyes).

In summary, Face Login is not secure against simple picture attacks whereas password, PIN, and Android Unlock Patterns suffer from shoulder-surfing attacks. Thus there is a need for an authentication method that solves these shortcomings. Multiple extensions for the PIN system have been proposed in recent years [DLFBH09, DLHH10, DLWD07, BOKK11, RRF04] that attempt to hamper shoulder-surfing. While recalling visual information like pictures is easier than recalling passwords or PINs [SCH70, BCVO12, Chi08], there are only a few approaches that make use of graphical secrets that cover this attack vector [FCB10].

We propose GRAPHNEIGHBORS, a graphical authentication mechanism that fixes these drawbacks. To evaluate different settings, we develop three approaches as prototypes that we describe in the following. We show that plain shoulder-surfing attacks do not reveal the login secret and additionally that smudge attacks are useless against our scheme. Like other authentication mechanisms, GRAPHNEIGHBORS utilizes different figures as secrets. We denote this set of figures as $\mathbb{F}$. Additionally, GRAPHNEIGHBORS uses a set of colors $\mathbb{C}$ as well as a set of optional positions $\mathbb{P}$. The user's secret is based on figures having each figure $f_i \in \mathbb{F}$ being concatenated with a user chosen color $c_i \in \mathbb{C}$ and a user chosen position $p_i \in \mathbb{P}$. The position refers to the selection the user has to choose when authenticating. In contrast to most existing graphical logins, the user does not select his secret directly to authenticate. Instead, he chooses the figure *besides* his partial secret to prevent shoulder-surfing attacks. The previously selected position is the key to decide which of the neighbors to choose. Therefore, a four-digit user's secret $\mathbb{S} = (s_1, s_2, s_3, s_4)$ can be described as the following ordered set:

$$(f_1 \circ c_1 \circ p_1, f_2 \circ c_2 \circ p_2, f_3 \circ c_3 \circ p_3, f_4 \circ c_4 \circ p_4)$$

with $s_i = f_i \circ c_i \circ p_i$. Note that $\circ$ denotes the concatenation operator. To configure the login, the user has to choose a secret $\mathbb{S}$ beforehand.

As a first approach, we choose six different geometric figures (`circle` ◯, `square` □, `triangle` △, `rhombus` ◊, `pentagon` ⬠, and `star` ★), four different colors (`blue`,

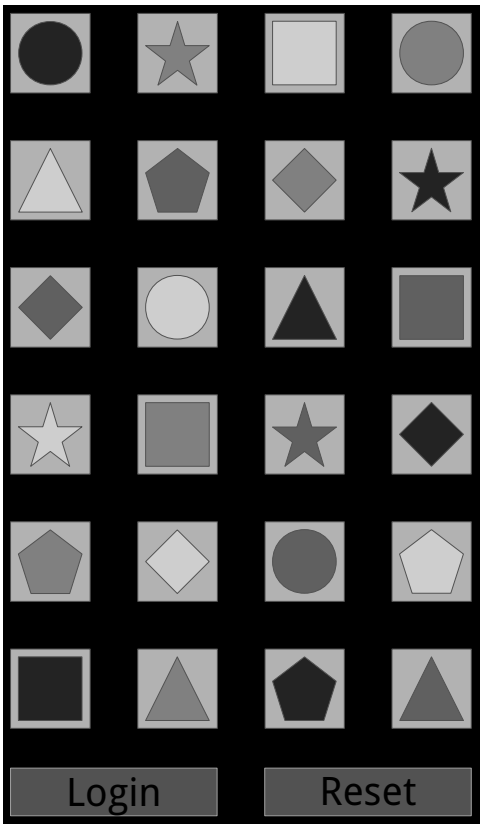

Figure 1: View for approach A and B. The user has to choose at least four objects as secret. In contrast to approach B, approach A additionally uses directions as secret for each object.

`green`, `yellow`, and `red`), and five different locations (`above`, `right`, `below`, `left`, and `none`). The login view for this approach can be seen in Figure 1 presenting a matrix of 6×4 figures randomly located. The authentication process is round-based: For each round, the user has to find the figure that matches his chosen secret regarding shape and color and select the figure corresponding to his position selection beside the located figure. If $f_i \circ c_i$ is located at the upper left edge of the screen and the previously chosen position p_i is left, he has to choose the most right figure in the same row. If p_i is above, the user has to select the bottommost figure in the same column. Other cases match accordingly. Like for other methods that have a user defined secret's length, the user has to press a button after having inserted the full secret $\mathbb{S}$.

To give a concrete example, we explain the authentication process based on Figure 1. Let the first partial secret be $f_1 \circ c_1 \circ p_1$ with $f_1 = \bigcirc$, $c_1 = \texttt{red}$, and $p_1 = \texttt{below}$. Given the aforementioned figure, the user has to select the `blue` $\pentagon$ since this figure is right below the red circle. Let the second partial secret be $f_2 \circ c_2 \circ p_2$ with $f_2 = \square$, $c_2 = \texttt{blue}$, and $p_2 = \texttt{left}$. For the second round, the user has to select the red triangle since the square

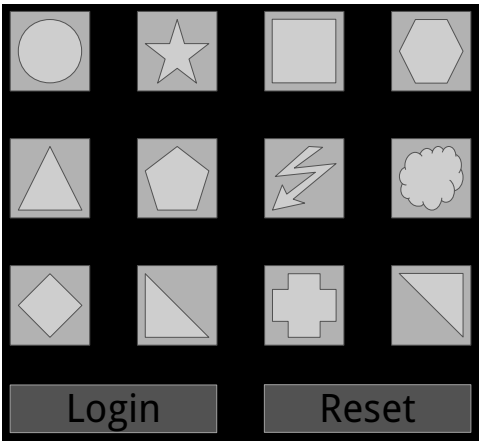

Figure 2: View for approach C. The user has to choose at least four objects as secret. Additionally, directions for each object must be chosen.

is the leftmost figure in the last row.

Our second approach is very similar to the first one: We also choose six different geometric figures and four colors, but the user cannot choose different locations on his own. Thus, the automatic selection for this property is `none`. Since we omit the neighbor option, this method is not secure against shoulder-surfing attacks. By offering this approach, we intended to evaluate whether more difficult approaches like approach A are usable in comparison to omitting this feature.

Our final approach differs from the first and second one by the number of figures and the number of colors (c. f. Figure 2). We implemented twelve different figures (`circle` ◯, `star` ★, `square` □, `hexagon` ⬡, upward directed isosceles `triangle` △, `cloud` ☁, `pentagon` ⬠, `lightning` 🗲, `rhombus` ◊, `cross` ✜, left downward directed isosceles `triangle` ◺, and right upward directs isosceles `triangle` ◹) with five different locations as in approach A, but used only a single color. We considered this approach since we wanted to find out whether the concept of neighbors works with only one color but more figures. On a smartphone, these figures can easily be taken from the user's gallery since there is no need to colorize them. Note that this is only feasible for approach C since some pictures might be indistinguishable when being colored as needed for the approaches A and B. For our prototype, we used geometric figures due to the easy comparability. Pictures instead of figures might be more familiar for the smartphone's owner, but less comparable for a study.

We implemented all three approaches as prototypes in a tool called GRAPHNEIGHBORS for the Android operating system. A summary of the method's properties can be found in Table 1. In the next section, we evaluate and discuss the usability of them.

Table 1: Comparison of the properties of different approaches.

	Figures	Colors	Neighbors	Displayed Objects
Approach A	6	4	5	24
Approach B	6	4	1	24
Approach C	12	1	5	12

4 Usability Evaluation

To evaluate GRAPHNEIGHBORS, we conducted a user study with 100 participants. In the beginning, we explained the GRAPHNEIGHBORS to each participant face to face and answered upcoming questions. We further asked them to select and remember a secret for each approach individually. After five minutes, we requested each participant to login with the previously chosen secrets having three tries for each approach. Finally, we asked them to fill out a questionnaire.

We learned that $74\,\%$ of all participants where able to authenticate using approach A within three attempts. $86\,\%$ of all participants could login when using approach B while $96\,\%$ could authenticate by means of approach C. Table 2 outlines details about successful logins and the number of attempts participants needed to authenticate. By intersecting the sets of successful participants we discovered that $68\,\%$ could authenticate using each single approach and that almost all subjects ($98\,\%$) could authenticate using at least one of the approaches.

We further asked our participants to compare all three approaches of GRAPHNEIGHBORS with common login methods and name the method where they could remember the secret best. $27\,\%$ stated that they could remember secrets for approach C best, while $25\,\%$ could remember PINs best. $18\,\%$ liked passwords most followed by $15\,\%$ preferring Android Unlock Patterns. Approach A was chosen by only $11\,\%$ and approach B by only $4\,\%$. Note that approach C performs better than all legacy authentication methods while being secure against simple shoulder-surfing attacks (c. f. Section Security Evaluation).

We also asked which authentication method they would prefer as default for their smartphone. $34\,\%$ gave approach C as preferred method followed by PIN with $19\,\%$ and approach A as well as other mechanisms with $14\,\%$ each. Passwords achieved $10\,\%$ while approach B got $5\,\%$ and Android Unlock Patterns $4\,\%$.

Table 2: Number of attempts to successfully login for 100 participants.

	Approach A	Approach B	Approach C
1^{st} try	42	59	94
2^{nd} try	28	25	2
3^{rd} try	4	2	-
Σ	74	86	96

5 Security Evaluation

In this section, we introduce four attacker scenarios and argue how GRAPHNEIGHBORS performs in contrast to legacy authentication approaches like PIN, password, and Android Unlock Pattern. We concentrate on the average case since worst or best case scenarios do not point out reality.

The first attack we discuss is guessing the secret without knowing the secret's length. Since $\sum_{i=4}^{n} b^i$ is the password space for passwords with at least 4 and at most n characters having a basis of b different characters, we need on average $\frac{\sum_{i=4}^{n} b^i}{2}$ tries to reveal a password. Given that, we get $\frac{\sum_{i=4}^{16} 95^i}{2} \approx 1.09 \times 2^{104}$ as concrete number. Transferred to the PIN authentication, we obtain $\frac{\sum_{i=4}^{16} 10^i}{2} \approx 1.23 \times 2^{52}$ for $b = \{0, \ldots, 9\}$. For Android Unlock Patterns an exact number cannot be given in a closed equation since login patterns have a lot of restrictions, e. g., points cannot be used twice and not every path is allowed. Considering all restrictions, we nearly get 400.000 possible patterns. Therefore, we need on average $200.000 \approx 1.53 \times 2^{17}$ tries to expose the login secret. Comparing this to approach A of GRAPHNEIGHBORS, we have a base of $6 \times 4 = 24$ different object/color combinations. Although we additionally can choose five different directions, this does not change the security of this approach because all possible objects are already element of these 24 objects. Hence, we obtain $\frac{\sum_{i=4}^{16} 24^i}{2} \approx 1.34 \times 2^{72}$ as average count of attacks to guess the secret. For approach B, we get the very same number since we do not need to take the number of neighbors into account. This differs for approach C where we have only 12 different objects. Given this, we get $\frac{\sum_{i=4}^{16} 12^i}{2} \approx 1.40 \times 2^{56}$ as average number of attacks to guess the secret.

As second attack, we consider guessing the secret with knowledge of it's length. For our analysis, we assume a secret's length of four. Taking a look at passwords, we obtain $95^4 \approx 1.21 \times 2^{26}$ different possibilities. Therefore, an attacker needs approximately 2^{25} tries to expose the secret. For four digit PINs, we get $\frac{10^4}{2} \approx 1.22 \times 2^{12}$ possibilities. Android Unlock Patterns containing exactly four points only offer less than $\frac{9 \times 8 \times 7 \times 6}{2} \approx 1.47 \times 2^{10}$ since the first point can be chosen arbitrarily but further points depend on the previously chosen ones. Calculating this for GRAPHNEIGHBORS, we obtain $\frac{24^4}{2} \approx 1.27 \times 2^{17}$ for approach A and B, while we need on average $\frac{12^4}{4} \approx 1.27 \times 2^{14}$ tries to get the secret for approach C.

The third attack is guessing the user's secret after shoulder-surfing. When considering password, PIN, and Android Unlock Patterns, we conclude that all these approaches can easily be broken by conducting shoulder-surfing once. An attacker having observed the login procedure knows the secret and can unlock the smartphone afterwards. Since approach B of GRAPHNEIGHBORS does not make use of the neighbor option, this method can also be attacked by simple shoulder-surfing. However, approach A and C make use of the neighbor option and are therefore secure against single shoulder-surfing attacks since the attacker does not attain the secret. Albeit, they are not resilient against sophisticated shoulder-surfing attacks because an attacker conducting multiple shoulder-surfing sessions can perform an intersection attack to obtain the secret in the end. To do so, the

attacker captures the user's inputs including the possible neighbors and creates a list of password candidates. Having four rounds and five possibilities per round, the list consists of $5^4 = 625$ candidates. Due to the randomization of the figures on the touchscreen, each login session offers other candidates. By collecting and intersecting multiple candidate sets, the attacker can eventually obtain the secret.

Finally, we consider smudge attacks [AGM+10]. In this context, the oily residue on the smartphone's touchscreen is analyzed to obtain the login secret. Here, an attacker can inspect the touchscreen after the user has tapped or wiped on the touchscreen to enter the secret. The Android pattern login is most vulnerable against this attack vector. Since the user has to slide his finger to enter the secret, residues form a path on the touchscreen. This path has exactly two designated points: on the one hand the start point and on the other hand the end point. Start and end point can be interchanged but this only leads to two possible secrets. Having five tries to authenticate using this method, we can consider that plain login patterns are highly vulnerable. By looking at password and PIN, attackers can see where a user has touched the touchscreen. Therefore, they know the touched position but have no information about the order of touches. For a secret with four different characters (numbers or letters) we get $4! = 24$ possible secrets leading to 1.5×2^3 tries on average. Since all approaches of GRAPHNEIGHBORS show the geometric figures randomly arranged, smudge attacks do not lead to any hint of the secret.

Limitations

While being more secure against shoulder-surfing attacks than PIN, password, and Android Unlock Patterns, GRAPHNEIGHBORS also has at least one limitation. Since the number of selectable neighbors is limited, observing the authentication reveals a hint of the login secret. By intersecting the possible password candidates, this can lead in the long run to the secret password. This attack can only be mitigated by changing the password on a regular basis. Nevertheless, GRAPHNEIGHBORS hampers shoulder-surfing attacks since even a sophisticated shoulder-surfing attack performed once does not lead to the login secret.

6 Conclusion and Future Work

In this paper, we introduced a novel graphical authentication method that makes use of neighbors to achieve resistance against plain shoulder-surfing attacks. We implemented three different approaches as a prototype for the Android OS. In a usability evaluation, we asked 100 people to choose secrets for each approach and verified that almost all participants ($98\,\%$) where able to remember the secret after a short period of time of at least one approach. In a succeeding survey we found that $34\,\%$ preferred using our proposed neighbor option with 12 different figures but without additional colors. Our security evaluation shows that graphical neighbors improve the security against shoulder-surfing on the

one hand and smudge attacks on the other compared to commonly used authentication methods.

In future work, we plan to focus on a large-scale and long-term evaluation to verify that people can remember not only figures, but also positions over a long period of time.

References

[AGM+10] Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith. Smudge Attacks on Smartphone Touch Screens. In *USENIX Workshop on Offensive Technologies (WOOT)*, 2010.

[BCV08] Davide Balzarotti, Marco Cova, and Giovanni Vigna. ClearShot: Eavesdropping on Keyboard Input from Video. In *IEEE Symposium on Security and Privacy*, pages 170–183. IEEE Computer Society, 2008.

[BCVO12] Robert Biddle, Sonia Chiasson, and P.C. Van Oorschot. Graphical Passwords: Learning From the First Twelve Years. *ACM Computing Surveys*, 44(4):19:1–19:41, September 2012.

[BOKK11] Andrea Bianchi, Ian Oakley, Vassilis Kostakos, and Dong-Soo Kwon. The Phone Lock: Audio and Haptic Shoulder-Surfing Resistant PIN Entry Methods for Mobile Devices. In *Tangible and Embedded Interaction*, 2011.

[Chi08] Sonia Chiasson. *Usable Authentication and Click-based Graphical Passwords*. PhD thesis, Carleton University, 2008.

[DLFBH09] Alexander De Luca, Bernhard Frauendienst, Sebastian Boring, and Heinrich Hussmann. My Phone is My Keypad: Privacy-Enhanced PIN-Entry on Public Terminals. In *Annual Conference of the Australian Computer-Human Interaction Special Interest Group*, 2009.

[DLHH10] Alexander De Luca, Katja Hertzschuch, and Heinrich Hussmann. ColorPIN: Securing PIN Entry Through Indirect Input. In *ACM Conference on Human Factors in Computing Systems (CHI)*, 2010.

[DLWD07] Alexander De Luca, Roman Weiss, and Heiko Drewes. Evaluation of Eye-Gaze Interaction Methods for Security Enhanced PIN-Entry. In *Australasian Conference on Computer-Human Interaction: Entertaining User Interfaces*, 2007.

[FCB10] Alain Forget, Sonia Chiasson, and Robert Biddle. Shoulder-Surfing Resistance with Eye-Gaze Entry in Cued-Recall Graphical Passwords. In *ACM Conference on Human Factors in Computing Systems (CHI)*, 2010.

[MVG+11] Federico Maggi, Alberto Volpatto, Simone Gasparini, Giacomo Boracchi, and Stefano Zanero. A Fast Eavesdropping Attack Against Touchscreens. In *7th International Conference on Information Assurance and Security (IAS)*, 2011.

[RRF04] Volker Roth, Kai Richter, and Rene Freidinger. A PIN-Entry Method Resilient Against Shoulder Surfing. In *ACM Conference on Computer and Communications Security (CCS)*, 2004.

[SCH70] Lionel Standing, Jerry Conezio, and Ralph N. Haber. Perception and Memory for Pictures: Single-trial Learning of 2500 Visual Stimuli. *Psychonomic Science*, 19(2):73–74, 1970.

[TA08] H. Tao and C. Adams. Pass-Go: A Proposal to Improve the Usability of Graphical Passwords. *International Journal of Network Security*, 7(2):273–292, 2008.

[TKC05] Desney S. Tan, Pedram Keyani, and Mary Czerwinski. Spy-Resistant Keyboard: More Secure Password Entry on Public Touch Screen Displays. In Ash Donaldson, editor, *OZCHI*, volume 122 of *ACM International Conference Proceeding Series*. ACM, 2005.

[TOH06] Furkan Tari, A. Ant Ozok, and Stephen H. Holden. A Comparison of Perceived and Real Shoulder-Surfing Risks Between Alphanumeric and Graphical Passwords. In *Symposium on Usable Privacy and Security (SOUPS)*, 2006.

[UDWH13] Sebastian Uellenbeck, Markus Dürmuth, Christopher Wolf, and Thorsten Holz. Quantifying the Security of Graphical Passwords: The Case of Android Unlock Patterns. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.

[ZL07] Huanyu Zhao and Xiaolin Li. S3PAS: A Scalable Shoulder-Surfing Resistant Textual-Graphical Password Authentication Scheme. In *AINA Workshops (2)*, pages 467–472. IEEE Computer Society, 2007.