Keep Others from Peeking at Your Mobile Device Screen!

Chun-Yu (Daniel) Chen, Bo-Yao Lin, Junding Wang, and Kang G. Shin
CSE/EECS, University of Michigan
Ann Arbor, MI, USA
{chunyuc,boyaolin,jundwang,kgshin}@umich.edu

ABSTRACT

People use their mobile devices anywhere and anytime to run various apps, and the information shown on their device screens can be seen by nearby (unauthorized) parties, called *shoulder surfers*. To mitigate this privacy threat, we have developed `HideScreen` by utilizing the human vision and optical system properties to hide the users' on-screen information from the shoulder surfers.

Specifically, `HideScreen` discretizes the device screen into grid patterns to neutralize the low-frequency components so that the on-screen information will "blend into" the background when viewed from the outside of the designed range. We have developed and evaluated several ways of hiding both on-screen texts and images from shoulder surfers. Our extensive experimental evaluation of `HideScreen` demonstrates its high protection rates (>96% for texts and >99% for images) while providing good user experience.

CCS CONCEPTS

• **Security and privacy → Privacy protections**.

KEYWORDS

Shoulder surfing; mobile privacy; privacy protection; human machine interaction

ACM Reference Format:
Chun-Yu (Daniel) Chen, Bo-Yao Lin, Junding Wang, and Kang G. Shin. 2019. Keep Others from Peeking at Your Mobile Device Screen!. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19), October 21–25, 2019, Los Cabos, Mexico.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3300061.3300119

MobiCom '19, October 21–25, 2019, Los Cabos, Mexico

ACM ISBN 978-1-4503-6169-9/19/10…$15.00
https://doi.org/10.1145/3300061.3300119

1 INTRODUCTION

People use mobile devices everywhere they go, even in public areas. The information shown on their device screens could be personal or sensitive (e.g., bank account information and text messages to/from personal friends), and hence the users would not want others to see it. However, people nearby can easily see the information displayed on the device screen by peeking at (or *shoulder surfing*) the screen, as shown in Fig. 1.

Figure 1: A common case of shoulder surfing in a train/bus and the effect of applying our proposed solution, `HideScreen`**.**

Although users will take proper defensive actions when they beware of someone else's peeking at their device screens, they are reported to beware of only 7% of shoulder surfing incidents [1]. Moreover, shoulder surfers are reported to succeed in obtaining a 6-digit PIN with a 10.8% probability by taking just one peek [2]. Users may try not to view sensitive/private information in public areas, but cannot always help it. For example, the Justice Secretary of Philippines, Vitaliano Aguirre II, was enraged at the leakage of his text messages by someone who had peeked at, and taken a picture of, his smartphone screen during a Senate hearing [3].

The most popular defense against shoulder surfing is to attach a privacy film on the device screen, which limits the visible range of screen to a certain viewing angle to hide the on-screen information (OSI). Even though it is an effective way to protect OSI when the shoulder surfer is outside of the visible range, the privacy film provides little protection inside the visible range, *e.g.,* the shoulder surfer is right behind the user (Section 8.5). Also, it requires users to beware of the privacy risks and take appropriate actions before viewing any sensitive OSI. This requires users to buy and attach a privacy film, or buy devices equipped with privacy films (*e.g.,* HP Sure View [4]), incurring additional cost and/or effort.

Researchers and IT companies have been seeking software solutions that can *proactively* protect users' OSI from the end of information provider (*e.g.,* Google's shoulder surfer detection [5]), or let users hide their OSI without requiring other hardware protection (*e.g.,* BlackBerry's Privacy Shade [6]). They usually focus on the protection of (i) authentication secrets, which can be used to "unlock" devices and provide access to their contents, and (ii) other general information, including the on-screen texts and images displayed by applications. The former only focuses on the protection of authentication secrets, while the latter usually blocks the information altogether, which also prevents the intended user from viewing the OSI (Section 11).

One of the most recent and promising ways of preventing shoulder surfing is IllusionPIN [7]. It utilizes the concept of *hybrid image* [8] to hide the real and the decoy keypads in high and low spatial frequency bands, respectively. That way, users can read the real keypad while shoulder surfers cannot. Even though IllusionPIN is able to protect general images, as the authors of [7] stated, the parameters used need to be appropriately tuned for *each* particular task to achieve the required protection (Section 11).

Considering the possible leakage of sensitive OSI (SOSI) and the lack of their effective protection, we would like to enable information senders/providers to *proactively* protect SOSI instead of *passively* relying on the awareness and presence of protection at the receivers, such as use of a privacy film. We meet this goal by developing a novel solution, called `HideScreen`, for SOSI protection without requiring any additional hardware. It (i) can protect the SOSI without compromising users' intended tasks/apps, and (ii) is simple enough to implement and run on commodity mobile devices while consuming as little resources (*e.g.*, computing power and energy) as possible to support good user experience.

We have developed `HideScreen` with the goal of providing the user a special "private view" that can be deployed as an API, a built-in function in the device OS, or a stand-alone app (Section 10.1). Developers or information providers can choose to display sensitive information (with protection in the private view) on the screen that can still be seen correctly by the user. Before displaying the protected information, `HideScreen` will first acquire the information — either via direct input with the API call, or image processing — to be displayed, generate and then display the protected version.

`HideScreen` is tailored to meet the need of apps that display some short but sensitive information — such as PIN, account/password, and partially-personal messages — and protect the OSI from unauthorized parties located outside of the designed viewing range. Specifically, `HideScreen` focuses on the protection of short texts on the screen which can also protect the texts shown on soft keypad/keyboard. When key shuffling is used, `HideScreen` will prevent a shoulder surfer from acquiring sensitive information by observing the on-screen locations the user touches. Other than on-screen texts, some images, such as personal photos or the security picture used for bank account login, can also be privacy/security-sensitive, and hence `HideScreen`'s protection is extended to on-screen images.

Unlike IllusionPIN that utilizes the spatial frequency of information itself, `HideScreen` injects high spatial frequency components into the information to be displayed, thereby neutralizing the low-frequency components that are easily viewable by shoulder surfers. This mechanism is equivalent to moving the low-frequency components to the higher-frequency space that cannot be seen by shoulder surfers (Fig. 2). Specifically, based on optical system properties and human vision characteristics (Section 3), we propose use of grid patterns to display information so that the information to be protected can only be viewed within a designed range (Section 4). Furthermore, since `HideScreen` does not rely on the spatial frequency of the OSI itself to provide protection, the viewable range of the protected information can be adjusted dynamically and automatically based on the user's viewing distance by changing the grid parameters, thus broadening its use for various applications with different requirements.

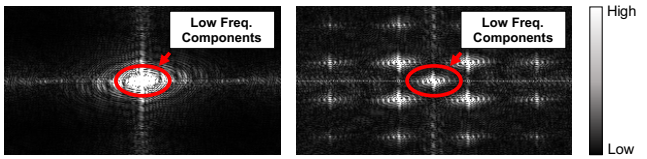
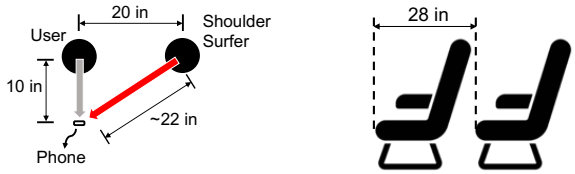


Figure 2: This example shows the spatial frequency spectrum of a single character 'a' before and after `HideScreen` is applied.

The only requirement for deploying `HideScreen` is that the displayed colors must be calibrated (Section 7). Therefore, the figures shown throughout this paper may not reveal the designed effect because the colors/images displayed may vary with devices. This calibration is needed only once per device and takes only several minutes, which can be done by device manufacturers or users. Like all the protection schemes, `HideScreen` inevitably degrades the legitimate user's readability of OSI (Section 10.2). So, we have also developed mechanisms to enhance the user experience while maintaining the level of protection. Furthermore, since `HideScreen` is a software-based approach, it can be implemented to give users the options to enable/disable the protection in the private view at any time, depending on their needs.

This paper makes the following contributions:



(a) Top-down view of Fig. 1 (b) Seat pitch between rows

Figure 3: Threat model settings

- Proposal of grid-based display for OSI protection (Section 4), based on optical system properties and human vision characteristics (Section 3);
- Development of text (Section 5) and image (Section 6) protection, and demonstration of their effectiveness in protecting texts/images (Section 8) at a low rate of information leakage ($\leq$ 3.8% and $\leq$ 0.9%, respectively).
- Demonstration of `HideScreen`'s practical usability by evaluating its readability, energy consumption, and use-case study (Section 9).

2 THREAT MODEL

The goal of adversaries (shoulder surfers) is to acquire the information shown on the user's device screen. The information that shoulder surfers interested in would be texts and static images associated with applications such as messaging/texting and account login. Since the reported/known shoulder surfing events are mostly casual and opportunistic [1], we assume the most common case in which shoulder surfers (SSs) use their eyes or smartphone cameras to acquire/comprehend the on-screen information.

Smartphones/tablets/laptops are assumed to be viewed by their users from a distance up to 24" (equal to the length of human arms [9]). If and when the SS tries to acquire OSI with his own eyes, the difference between the user's and the SS's viewing distance is assumed to be greater than 12" because a SS would not want to be caught by the device user that he is peeking at the screen. As shown in Fig. 3(a), this is also the difference between the user's and the SS's viewing distance when the SS is sitting right next to the user while the distance between the user and the SS is 20" (*i.e.*, the average shoulder width [9]) and the viewing distance of the user is 10" (*i.e.*, average smartphone viewing distance [10]).

We assume that the SS may also try to acquire the user's OSI by using his smartphone camera, for example, when he is sitting in a seat behind the user. In such a case, the distance between the device and the SS will be the size of the seat pitch (Fig. 3(b)). Since the seat pitch ($\geq$28" [11]) on an airplane tends to be smaller than other transportation vehicles, we use it as the setting of our threat model, erring on the conservative side.

As mentioned before, we also assume casual and opportunistic shoulder surfers, but not malicious professionals with special equipments, such as binoculars and digital cameras, who target a specific individual for specific information. This excludes the case in which the shoulder surfers use a camera to video-record the user's device screen and then process the video to extract the sensitive information. This exclusion should not diminish the value of `HideScreen`, since users are unlikely to view their confidential information in public areas and the attackers have other ways to obtain the target information than shoulder surfing, such as implanting/installing malware in the targeted user's device. Nevertheless, we will discuss the protection against the attacker's use of special equipments in Section 8.

3 RESOLVING POWER AND HUMAN VISION

Before detailing `HideScreen`, we first introduce some necessary background.

3.1 Resolving Power of Optical Systems

An optical system is capable of producing or perceiving light. Here we focus on optical systems that perceive light. The *resolving power* of an optical system is defined as its ability to distinguish two adjacent light sources [12]. It is usually represented by the minimum angle with which the system can distinguish two separate light sources as individual ones (*i.e.*, *minimum resolvable angle*).

The limit of the resolving power of an optical system is determined by the diffraction of light. According to Rayleigh Criterion, an optical system with an aperture can resolve two separate point light sources if the first principal diffraction minimum (PDMin) of one light source coincides with the principal diffraction maximum (PDMax) [12]. Figs. 4(a), 4(b), and 4(c) show the conditions for two light sources to be resolvable, just resolvable (*i.e.*, Rayleigh Criterion), and unresolvable, respectively. This minimum angle separation is then given by

$$\theta_{min} = 1.22\lambda/D, \tag{1}$$

where λ is the wavelength of light, and D is the aperture diameter of the optical system [12]. As shown in Eq. (1), the resolving power of an optical system is determined by its aperture. That is, the larger the aperture, the greater the resolving power.

3.2 Human Vision Characteristics

Human vision is nothing but an optical system and hence is subject to Rayleigh Criterion. However, humans' perception of images/patterns has some special characteristics. The authors of [13] conducted experiments to determine humans' perception sensitivity with respect to spatial frequencies.

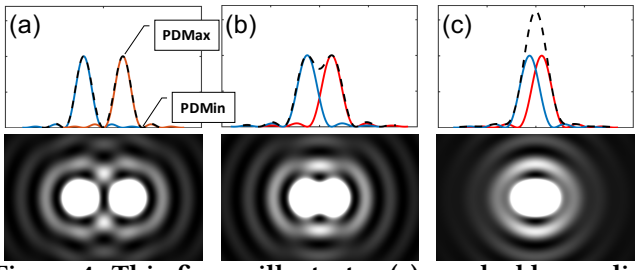


Figure 4: This figure illustrates (a) resolvable condition, (b) Rayleigh Criterion, and (c) unresolvable condition.

Human vision is shown to have the highest sensitivity when the spatial frequency is around 8 *cycles per degree* (c/d) and the perception of patterns is cut off around 60 c/d (*i.e.*, a person cannot recognize that there is a pattern).

4 SYSTEM DESIGN

4.1 Design Overview

While the information embedded in the texts is their meaning, the information embedded in the images is their spatial pattern. To deal with their unique characteristics, we introduce different schemes for different protection targets. `HideText`, `HideImage`, and `SelImage` are the three protection schemes employed in `HideScreen`.

These three schemes are compared and summarized in Table 1. `HideText` focuses on the protection of texts, and the other two protect images. All of these are designed to protect information by viewing distance and angle, meaning that a shoulder surfer will not be able to read the information correctly from the outside of the designed viewing range.

The two image protection schemes differ in loss or no loss of information. `HideImage` protects the images at the cost of some content loss (*i.e.*, not showing the original image on the screen), while `SelImage` protects the images without loss of content, thus allowing a SS to be able to identify the real information with some probability.

	`HideText`	`HideImage`	`SelImage`
Protection Target	Text	Image	Image
Protection by Viewing Distance	✓	✓	✓
Protection by Viewing Angle	✓	✓	✓
Protection by User Interaction	×	×	✓
Plain Text/Image Protection	✓	✓	×
Lossless Protection	✓	×	✓

Table 1: Comparison of protection schemes in `HideScreen`

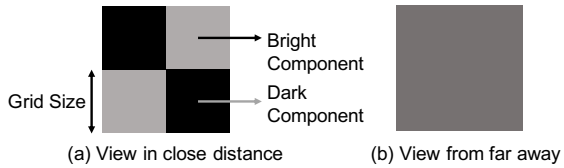


Figure 5: Basic concept of `HideScreen`

All schemes are built on one basic idea: if a user views a grid pattern within the designed range, he will see the grid (Fig. 5a), else he will only see an area of single color (Fig. 5b). `HideScreen` captures the information to be shown on the screen and transforms it into a *grid image*, which is an image composed of grids. Before discussing the details of a grid image, we first introduce the properties of a grid.

4.2 Characteristics of a Grid

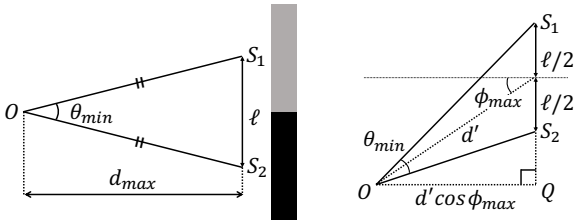
There are three main characteristics of a grid (Fig. 5a). The first is *grid size*, which is defined as the size of a single color square. Since we can approximate a single-colored square as a single point light source, the grid size determines the range that users can view the information (denoted as *visible range*). The larger the grid size, the longer the visible range. The second and third characteristics are the colors of bright (H_{bright}) and dark (H_{dark}) components in the grid, respectively.[1] These two characteristics determine the color a user will perceive when viewing the grid. We use $H \approx H_1 \oplus H_2$ to mean that a grid with (H_1, H_2) components looks the same as a single color H when it is viewed from far away. Note that the colors may look slightly different on different screens even if they are displaying a color with the same color value. So, to use `HideScreen`, the displayed colors must be calibrated *a priori* only once for each device. We will discuss how this calibration is done in Section 7.2.

4.3 Calculation of Visible Distance and Angle for General Optical Systems

Let us consider how to calculate the visible distance and angle for a given grid pattern with grid size ℓ. Suppose we are using an optical system, such as a camera, to view the grid pattern, and its aperture/lens size is D. The maximum resolution [12] is determined by the minimum resolvable angle (Eq. (1)).

Fig. 6(a) shows the relationship between the minimum resolvable angle (θ_{min}) and the visible distance (d_{max}). After

[1]Each color H can be represented as (r, g, b), where r, g, and b are the values of red, green, and blue components, respectively. It can also be represented by a single integer (#000000 - #FFFFFF) when 24-bit color coding is used.



(a) Minimum resolvable angle θ_{min} and visible distance d_{max}

(b) Minimum resolvable angle and visible angle ϕ_{max}

Figure 6: Illustration of visible range calculation

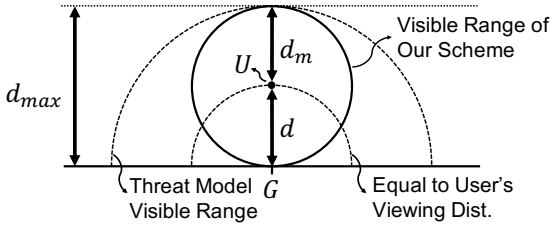


Figure 7: This figure shows the visible range of a user U whose viewing distance is d and d_{max} is designed to be $d + d_m$ inches.

obtaining θ_{min}, we can calculate the visible distance as:

$$d_{max} = \frac{\ell}{2\tan(\frac{1}{2}\theta_{min})} = \frac{\ell}{2\tan(0.66\frac{\lambda}{D})}. \tag{2}$$

Now, let's consider the case when a user is viewing the grid from distance $d' < d_{max}$ as shown in Fig. 6(b). If the user keeps moving away from the perpendicular line, the maximum visible angle ϕ_{max} will occur when the viewing angle between S_1 and S_2 is θ_{min}. We can obtain ϕ_{max} by solving Eq. (3) = Eq. (5):

$$area(\Delta OS_1S_2)$$

$$= \frac{1}{2} \times |\overline{S_1S_2}| \times |\overline{OQ}| = \frac{1}{2}\ell d' \cos\phi_{max} \tag{3}$$

$$= \frac{1}{2}\sin(\angle S_1OS_2) \times |\overline{OS_1}| \times |\overline{OS_2}| \tag{4}$$

$$= \frac{1}{2}\sin\theta_{min} \times \sqrt{(d'\cos\phi_{max})^2 + (d'\sin\phi_{max} + \frac{\ell}{2})^2}$$

$$\times \sqrt{(d'\cos\phi_{max})^2 + (d'\sin\phi_{max} - \frac{\ell}{2})^2}. \tag{5}$$

If $d' \gg \ell$ and ϕ_{max} is not close to 0, we have $\phi_{max} \approx \cos^{-1}(d'\theta_{min}/\ell)$.

4.4 Viewing Distance Calculation for Human Vision

As mentioned in Section 3.2, human vision has the perception cutoff at around 60 c/d. Therefore, the minimum resolvable angle for human vision is

$$\theta_{H,min} \approx 1/60^\circ \approx 3 \times 10^{-4}(rad). \tag{6}$$

Note that the equations in Section 4.3 can also be applied to human vision if we replace θ_{min} with $\theta_{H,min}$. The reason for this substitution is that the result obtained in Section 4.3 depends only on the resolving power, *i.e.*, the limit of spatial frequency that certain optical systems can resolve. So, it can be applied to any optical systems as long as their resolving power is known. The visible distance for human vision is then given as

$$d_{H,max} = \frac{\ell}{2\tan(\frac{1}{2}\theta_{H,min})} \approx 3333\ell. \tag{7}$$

4.5 Visible Distance and Range

Once the grid size ℓ is set, the visible distance of the grid is determined by:

$$d' \approx \frac{\ell\cos\phi}{\theta_{min}}, \tag{8}$$

where ϕ is the viewing angle, θ_{min} is the minimum resolvable angle, and d' is the corresponding visible distance. It shows that the visible distance is proportional to the cosine of viewing angle for a given grid size. Hence, the visible range is a circular area with the diameter equal to d_{max} (Fig. 7). Here we design

$$d_{max} = d_{H,max} = d + d_m, \text{where } d_m = 12". \tag{9}$$

The rationale behind this design, instead of setting $d_{max} = d$, is to leave the margin (d_m) for errors in the measurement of viewing distance, errors of grid quantization, and individual vision differences. This design also ensures that the protected information has better readability, since human vision generally has better sensitivity at lower spatial frequencies. $d_m = 12"$ is chosen as it is the minimum difference of viewing distances between the user and the shoulder surfer defined in our threat model.

4.6 Information Protection by Grids

We can utilize grids for hiding OSI because if a person views the grid from the outside of the visible range, he cannot resolve the bright and dark components into two individual sources. Therefore, he will only see the "mixture" of the two light sources. By utilizing this property, we can use the grid to create a pattern P for a designated visible distance d_{max}. What remains is to find a background B that has the same color as P when viewing from the outside of the visible range. We can then create a grid image $G = P + B$, so that only the person within the visible distance can see the pattern correctly. Fig. 8(a) shows an example of applying the protection

(a) Example of basic protection by grids

(b) Example of HideText

Figure 8: Text protection examples

by grids to a text, where the word "book" is the P component that is replaced by grids and the gray background is the B component.

5 TEXT PROTECTION

We now discuss how to utilize the background knowledge and the system design introduced in Sections 3 and 4 to protect on-screen text information.

5.1 Overview of HideText

Let d be the user's viewing distance. Based on our design captured in Eq. (9) and the result of Eq. (7), we can calculate the grid size ℓ as:

$$\ell = (d + 12")/3333. \tag{10}$$

Since the grids are composed of pixels, the actual grid size, ℓ^*, is also determined by the pixel size, ℓ_p:

$$\ell^* = round[\ell/\ell_p] \times \ell_p. \tag{11}$$

From our preliminary testing of pure grid-based protection, we find it sometimes difficult for users to comprehend the protected texts. So, we need to enhance the readability of the protected texts without weakening the protection against shoulder surfers.

We propose the most intuitive way to enhance the intended user's readability — adding boundaries to the girds. However, directly adding boundaries to the grid-based texts will compromise protection, because adding boundaries is equivalent to adding a constant low-frequency component to the original grid-based texts, and hence a shoulder surfer can see this low-frequency component from far away. To solve this problem, we need to add not only the boundaries to the grid-based texts, but also its complementary components to neutralize the effect of low-frequency component. Fig. 8(b) shows an example of grid-based texts with boundary enhancements.[2]

5.2 Generating HideText

We take three basic steps to generate a protected text: (i) identify the text boundaries, (ii) replace the text boundaries, and (iii) fill in background.

[2]Note that readers may find it easier to comprehend Fig. 8(a) than Fig. 8(b) because the colors/images displayed vary with devices.

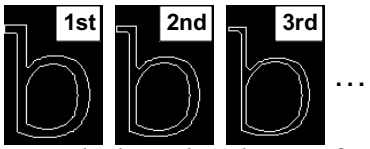


Figure 9: The boundary layers of text "b"

5.2.1 Identify Text Boundaries. The first step of generating a protected text is to identify the text boundaries. Since a text can be viewed as a binary image when displayed on the screen, the pixels that have direct contact with the background are considered as boundary pixels. HideText divides the text recursively by identifying the pixels that are next to the inner part of the previous boundary as shown in Fig. 9. We refer this set of boundaries as "boundary layers".

5.2.2 Replace Text Boundaries. After obtaining the boundary layers, we can use grids to display the texts as shown in Fig. 8(a). To create the highest contrast, H_{dark} and H_{bright} should be set to black (#000000) and white (#FFFFFF), respectively. The next step is to replace the first three boundary layers with curves of different colors. The rule of thumb is to replace the first and second boundary layers with darker and brighter curves, respectively, than the background color (H_{BG}), and the difference in color between the boundary layers and H_{BG} should decrease as L increases. We chose the colors for the first three boundary layers as:

$$H_L = round\left[H_{BG} + \frac{(3-L)(-1)^L}{10} H_{bright}\right], \tag{12}$$

where $L = 1, 2, 3$ are the indices of boundary layers. This way, HideText is able to compensate the low frequency components of each replaced layer. The boundary layers are used because they will increase low-frequency components and enhance the readability. However, adding more boundary layers will also make it easier to see the information from far distances. According to our preliminary experimental results, replacing three boundary layers can enhance readability without injecting low-frequency components too much.

5.2.3 Fill Background. The last step is to fill in the background with a certain color H_{BG}, so the protected text will blend into the background (*i.e.*, $H_{BG} \approx H_{bright} \oplus H_{dark}$) when viewed from the outside of the visible distance. In Section 7.2.1, we will discuss how to calibrate color to find H_{BG}.

5.3 Properties of HideText

The protected texts that HideText generates can be considered as a special type of font, which hides the texts from shoulder surfers. HideText can be implemented so as to store the previous computation results (as font files) and reuse them later. Since the grid patterns are composed of

Figure 10: Example of applying `HideImage` protection

pixels, the grid size can only be certain discrete values. Depending on the user's regular viewing distance and the size of an individual pixel, the grid size is usually composed of 3–5 pixels. So, it won't take up much storage space.

Since `HideText` protects information based on the average of adjacent pixels, it provides better protection if the original information has more lower frequency components. That is, `HideText` will provide better protection if the original texts are in bolder fonts than in thinner fonts.

`HideText` can also be applied to colored texts by changing the color of bright components. For example, if the desired text color is red, the bright components $H_{bright,R}$ can be set to (255,0,0) and we can use this colored grid with background $H_{BG} \approx H_{bright,R} \oplus H_{dark}$ to display texts with colors.

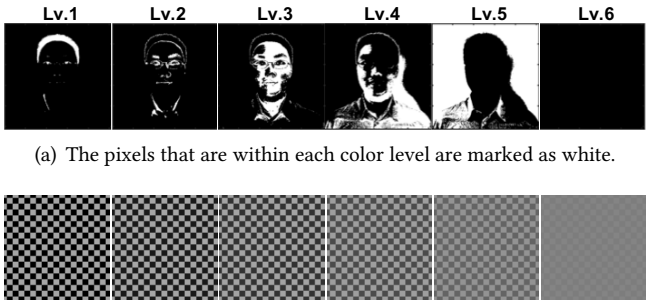
6 IMAGE PROTECTION

6.1 HideImage

`HideImage` — `HideScreen`'s direct image protection — is an enhancement of `HideText`. While `HideText` displays the full text without information loss, the protection of images will be achieved at some loss of information. The goal of `HideImage` is thus to provide a coarse-grained but comprehensible image to the user when he views the protected image.

Like `HideText`, `HideImage` utilizes the fact that a grid will look like a single-colored background when viewing from the outside of the designed range. Moreover, it utilizes different grids of the same "average" color (H_{avg}) to represent a different brightness scale. Fig. 10 shows an example of the original image and the corresponding protected version.

The first step of applying `HideImage` is to transform the image into grayscale and partition the greyscale image into layers of different color levels. We choose to use color levels that evenly partition the entire grayscale. Assuming we have a grayscale image (1-byte per pixel) as Fig. 10 and the desired number of levels (N_{level}) is 6, `HideImage` will identify the pixels within each level as shown in Fig. 11(a). The next step of applying `HideImage` is to replace the identified color levels with grids of different bright–dark component combinations



(a) The pixels that are within each color level are marked as white.

(b) The pixels marked as white in each level are replaced by the corresponding grids.

Figure 11: Example of applying color level partition

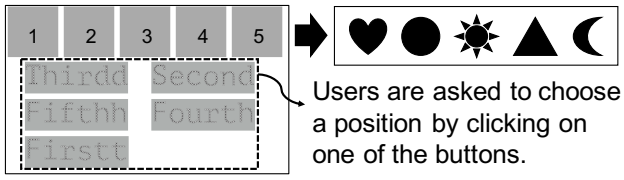


Figure 12: Image protection via user interaction

(Fig. 11(b)). Finally, the protected image can be obtained by adding the levels together.

The bright–dark component combinations are determined as follows. First, we identify the "average" color $H_{avg} = round[(H_{white} + H_{black})/2]$. Second, the color of dark component for layer i is identified by:

$$H_{dark,i} = round\left[\frac{(H_{avg} - H_{black})}{N_{level}} \times i + H_{black}\right]. \quad (13)$$

Finally, we identify $H_{bright,i}$ such that $H_{avg} \approx H_{bright,i} \oplus H_{dark,i}$ using the color calibration process (Section 7.2.2).

6.2 SelImage

Since `HideImage` protection will cause some visual information loss, we discuss an alternative image protection, called `SelImage`, which doesn't incur any information loss. This protection utilizes the concept of k-anonymity [14] to hide an image. That is, the real image is displayed along with $k-1$ other decoy images on the screen. Fig. 12 shows an example of `SelImage`. Before showing an image on the screen, the system will display blank image icons and ask the user to choose where to display the image. Of course, all the texts shown on the screen are protected by `HideText` and the order of choice options is randomized.

7 IMPLEMENTATION

We used Android smartphones and a tablet[3] for prototyping `HideScreen`. In particular, we have implemented 6 apps for

[3]Nexus 5X, Nexus 6P, Pixel XL, and Nvidia Shield K1 tablet.

the purpose of (i) color calibration, (ii) text protection evaluation, (iii) image protection evaluation, and (iv) use-case study (×3), respectively. Given below are the implementation details of these apps.

7.1 System Workflow

Since we use visual characteristics to hide the on-screen information, `HideScreen` first calibrates the color of the screen and creates a color profile. This is a one-time effort and every device needs color calibration once. After the calibration, `HideScreen`-supported applications will be able to load the information (*i.e.*, texts or images) and generate protected texts/images to be displayed on the screen according to the estimation user's viewing distance [15] and the color profile.

7.2 Color Calibration

The main goals of color calibration are to identify (i) H_{BG} for grids with different grid sizes and (ii) $H_{bright,i}$ for different color layers in `HideImage`. Fig. 13 shows the user interface we use during the development of `HideScreen`.

7.2.1 Identifying H_{BG}. First, a target grid is shown on the screen and the user should move the smartphone until s/he cannot see the details of the grid. The user can then adjust the background color by moving the seek bars. The next step is to adjust the brightness of background to match the target grid. The user may repeat the adjustments until s/he cannot distinguish the grid from the background (*i.e.*, $H_{BG} \approx H_{bright} \oplus H_{dark}$). This process should be repeated for each grid size.

7.2.2 Identifying $H_{bright,i}$. Opposite to the process of identifying H_{BG}, the background color will first be set to H_{avg}. The user should then adjust the seek bar of $H_{bright,i}$ until $H_{avg} \approx H_{bright,i} \oplus H_{dark,i}$ for a given $H_{dark,i}$. This process should be repeated for each combination of color levels and grid sizes.

During the development of `HideScreen`, we recruit participants to assess the calibration time using the interface shown in Fig. 13. The average calibration time per calibration is 16.0s. For a smartphone, it usually requires 3 grid sizes and 6 color levels to provide proper protection, i.e., a total of $3 \times (1 + 6) = 21$ cases. That is, users will be able to finish the entire calibration process within 6 minutes.

8 PROTECTION EFFECTIVENESS

8.1 General Evaluation Settings

Fig. 14 shows the basic evaluation settings. We recruited 20 volunteers of ages 18–40 on our campus for the evaluation of `HideScreen`.[4] There are at least 10 participants in each test.

[4]Institutional Review Board (IRB) No.HUM00140993.

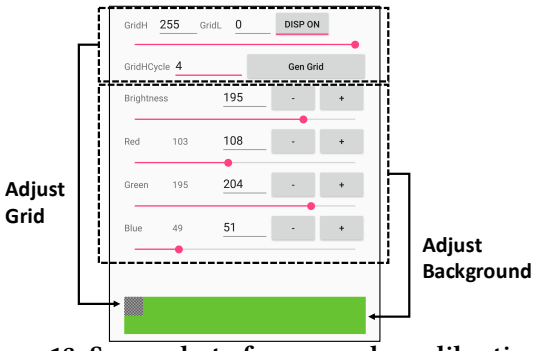


Figure 13: Screenshot of screen color calibration app

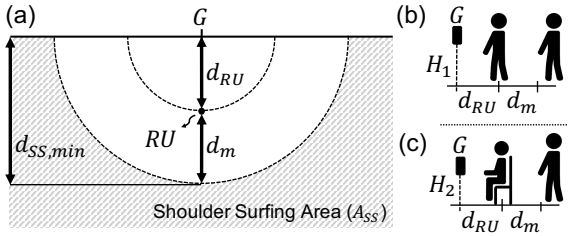


Figure 14: (a) Basic evaluation settings. (b) Settings when testing with a smartphone. (c) Settings when testing with a tablet or a laptop.

Before the evaluation starts, devices were placed on a table or a stand (position G in Fig. 14 (a)). Participants first act as the real user (RU) and view the content shown on the screen from certain distances (d_{RU}) right in front of the device. The information displayed on the screen is adjusted based on the estimation of the viewing distance. After acting as a RU, the participants act as a shoulder surfer and try to read/acquire the information shown on the screen. While acting as a SS, the participants are free to move their position as long as they are in the shoulder surfing area (A_{SS}). A_{SS} is defined to be the area in which the viewing distance (d_{SS}) is greater than $d_{SS,min} = d_{RU} + d_m$. In each case, we ask shoulder surfers to identify the information shown on the screen when it is viewed (a) from the outside of the designed visible range and (b) with special equipment. The participants were asked to try their best to identify as many targets as possible while acting as both a RU and a SS. Note that the participants did not know their roles in the experiment, since we did not tell them which roles they were to play and they were only asked to view the OSI within a certain range.

When testing with a smartphone (Fig. 14(b)), the participants were standing with the device placed on a phone stand ($H_1 = 50 - 55$") according to participants' heights. When testing with a tablet or a laptop (Fig. 14(c)), the device was placed on the desk ($H_2 = 30$") and the participants were sitting while acting as a user, and standing while acting as a shoulder surfer.

8.2 Evaluation Metrics

We use *shoulder surfer recognition rate* (SSRR) and *user recognition rate* (URR) as the metrics for the evaluation of `HideScreen`'s effectiveness. SSRR is defined as the probability that the surfer successfully reads the information on the screen:

$$SSRR = N_{SS,Success}/N_{Total} = 1 - R_{Protection},$$

where $R_{Protection}$ is the information protection rate. SSRR indicates the likelihood to fail to protect the on-screen information. Similarly, URR is defined as the probability that the user successfully reads the information on the screen:

$$URR = N_{RU,Success}/N_{Total}.$$

URR indicates whether or not the protection scheme maintains the comprehensibility of information. Ideally, SSRR (URR) should be close to 0 (1).

8.3 Evaluation of Text Protection

8.3.1 Experiment Settings. To evaluate the effectiveness of `HideText`, the protected texts are displayed in large size (0.3" character height on smartphone, 0.4" on tablet, 0.6" on laptop). In each test, one word is displayed on the screen with different levels of difficulty. The RU is asked to read out the displayed word. Similarly, the SS is also asked to read out the displayed word.

Even though this evaluation uses a single word at a time, it does not imply that HideText can only be applied to the short single word. On the contrary, we simulated the worst case for the users where the shoulder surfer knows exactly what s/he is looking for and directly targets that specific text on the screen. Furthermore, this design gives the shoulder surfer the advantage of using the length to identify the corresponding word. If the shoulder surfer cannot recognize a simple large-size word (e.g., boy), then it will be much harder for him to identify a specific, small-size (and possibly random) OSI mixed within a complete, protected message.

The test consists of 10 simple-level words ($\leq$ 5 characters), 10 medium-level words (6–10 characters), and 10 difficult words (11–15 characters). The words in the lists are chosen to be easily recognizable by the participants. That is, there are no difficult words in the lists that a college-level participant may not know. Some examples of "difficult" words (11–15 characters) are: congratulation, environment, and neighborhood. For each test-case, the participants were given 5 seconds to identify the word. In 5 seconds, the participants are able to guess or identify the word as many times as they can.

For the evaluation of smartphone, RUs are asked to read the words from 10–12" and 20–24", respectively. The former simulates the conditions that people hold/use the phone in normal posture while the latter simulates the conditions that the user stretches his arm forward to read the content. The SS is then asked to view the information from a distance greater than, or equal to $d_{SS,min} = d_{RU} + 12$ inches, simulating the case of the SS standing just behind the RU or sitting next to the RU. Similarly, we ask RUs to view the devices from their normal viewing distances, and ask SSs to view the device 12" away from the RU for the evaluation of tablet/laptop. Finally, we ask the SS to use binoculars from anywhere greater than 200" ($\approx$ 5m) to identify the information shown on the screen. This simulates the attacker's observation of users across the street.

8.3.2 Results. The results of protection effectiveness are summarized in Table 2, showing that `HideText` is able to achieve high URR ($\geq$ 96.4%) and low SSRR ($\leq$ 3.8%). In scenarios with normal viewing distances, URRs are 100%. That is, `HideText` can protect text information without degrading the RU's readability. To ensure that low SSRR values are not caused by texts being too small or participants' nearsightedness, we also ask SSs to identify the unprotected (plain) texts, which have the same size as the protected texts. The results show that all SSs are able to identify the plain texts (*i.e.*, SSRR = 100%), confirming `HideScreen`'s prevention of SSs from recognizing the texts.

When they are free to move while keeping their distance to the device greater than 200", all attackers with binoculars are able to identify the unprotected texts; none of the attackers using binoculars is able to identify the protected texts, which have the same size as the unprotected texts. This protection against use of binoculars is expected according to the calculation of resolving power. The binoculars we used in the experiments have 2.5cm (0.98") object lens diameters, hence requiring the SS to view the information within 193".

We further ask SSs with binoculars to move inside the 200" range to see if they can identify the texts. The recognition rate in this case is only 1.8%. This is due to the mismatch of focus range of binoculars and the visible range of the protected text. That is, depending on the attacker's vision, it usually requires a greater than 193" distance to correctly focus on the device screen.

Case (d_{RU})	URR (%)	SSRR (%)	SSRR w/ Bin. (%)
Phone (10-12")	100	3.8	0
Phone (20-24")	96.4	2.2	0
Tablet ($\approx$18")	100	0.0	0
Laptop ($\approx$24")	100	3.6	0

Table 2: Effectiveness of text protection

8.4 Evaluation of Image Protection

8.4.1 Experiment Settings. Similar to the testing scenarios of `HideText`, a protected image (0.5"×0.5") was displayed on the screen. In each test, RUs and SSs were asked to identify the protected image. We evaluated the protection effectiveness of `HideImage` with the following settings. Participants first

act as RUs and are asked to view the figure shown on the screen from the standard viewing distance (10–12"). The participants are asked to identify the protected images. Next, participants act as SSs and try to identify the information displayed on the device screen from 12" away from the RU and are also asked to identify the images.

Participants are asked to identify the protected images in two ways: with or without reference images. For the 20 test cases without reference images, participants are asked to identify the presented images, such as a human figure or a dog. These tests evaluate whether participants are able to identify the coarse-grained information.

For the 10 test cases with reference images, participants are asked to identify the original images from 5 given options. These tests evaluate whether participants can tell differences between multiple images with similar coarse-grained information. We made sure that all the original (unprotected) images were recognizable by shoulder surfers before applying `HideImage` to eliminate the possibility that the protection is the result of image size being too small.

Similar to the previous evaluation setting, participants are asked to first act as RUs and then SSs in the evaluation of `SelImage`. Five options are shown on the screen for the RU to pick which one is the real image (as the example shown in Fig. 12). For instance, if the RU chooses *Fifthh*,[5] the real object (the moon image) will be shown in the rightmost box. The locations of these object options are generated randomly and protected by `HideText`. After one of these options is chosen, the options disappear and the real object is then shown along with other four decoy images.

8.4.2 Results. Tables 3 summarizes the results of the effectiveness of `HideImage` and `SelImage`. Without reference images, `HideImage` is able to achieve 92.5% URR and 0.9% SSRR. Even though SSRR with reference images seems to be higher, 12% is still less than the probability of making a random guess (*i.e.*, randomly picking one of the five image options as the real information). That is, given the protected image, an attacker is not able to read information more accurately than making a random guess. For `SelImage`, URR is 100% and SSRR is 2%, indicating `HideScreen`'s protection of information from a SS who tries to read on-screen information. We also asked participants to use binoculars to read the OSI. Participants are then asked whether they have any clue in telling the real object. As expected, none of the participants was able to read the information.

[5]As one way to help the participants avoid using the length of the word as a hint to determine the information, we deliberately added an additional character at the end of shorter words.

	URR (%)	SSRR (%)	SSRR w/ Bin. (%)
`HideImage` (w/o ref.)	92.5	0.9	0
`HideImage` (w/ ref.)	92.7	12.0	0
`SelImage`	100.0	2.0	0

Table 3: Effectiveness of image protection

8.5 Protection of Applying Privacy Film

We compare the protections provided by `HideScreen` and a privacy film. We use a standard privacy film with a 60° viewing range [16], which will dim the brightness of the screen when the viewing angle (ϕ) exceeds 30°. We adopted the same settings as shown in Fig. 14(b) and asked the participants to view the OSI from various distances and angles. Participants were asked to first stand in front of the device and gradually increase the viewing angle while maintaining the same viewing distance ($d' = 24$"). The participants were shown unable to recognize the 0.3" texts at around 42°, indicating that the privacy film provides strong protection if the shoulder surfer is outside of a certain viewing angle. However, all 10 participants were able to recognize 0.3" unprotected texts when $d' = 36$" and $\phi < 60°$. Furthermore, all participants were also able to read regular size texts (14sp) when $d' = 24$" and $\phi < 60°$. That is, using a privacy film provides only little to no protection if the shoulder surfer is standing behind the user while `HideScreen` provides > 96% protection.

8.6 Protection Against Smartphone Cameras

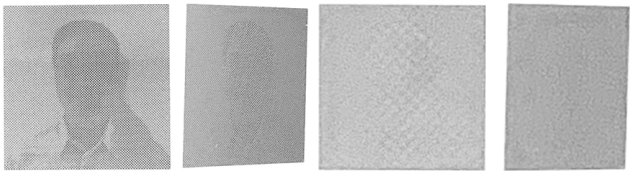
We now consider a case when a malicious party (MP) uses the camera on his smartphone to take snapshots[6] at the RU's screen. Fig. 15 shows an example of how a protected image will appear in the photos taken from different distances (d') and angles (ϕ). We use iPhone 6 and Pixel XL to take pictures of the RU's screen from 28" away. In this setting, the theoretical visible distance is around 34". That is, the MP is only 83% of the maximum visible distance away from the device. We took pictures of the 30-word test cases as in Section 8.3 and recruited 17 participants to view the pictures. The rate of correctly identifying at least one character in each photo is only 1.6% while the recognition rate of correctly identifying the entire word in each photo is only 0.8%.

We also use Google Cloud Vision API [17] to analyze the photos of protected information. The results show that only 3.3% of the photos are recognized as texts (within top 5 recognition results/tags). Furthermore, none of the texts can be correctly identified by the optical character recognition (OCR) function in Google Doc even if the recognition results

[6]Note that all the cameras are set to use their maximum zooming level to take photos in this part of experiments.

suggest that certain images contain texts. Similarly, we also ask participants to identify the objects shown in photos of protected images. The rate of correctly identifying the information in each photo is 0%, and that of using vision API is also 0%. When choosing the texts and images for this evaluation, we make sure that all of their unprotected versions can be identified by the vision API from the photos. In summary, HideScreen is able to provide good information protection even when the MP uses his smartphone camera to take a picture of the user's screen.

Albeit not included in our original threat model, we also conducted experiments with compact digital cameras and digital single-lens reflex (DSLR) cameras and took photos within 83% of their theoretical maximum visible distances. Using the same previous methodology, the results show that the recognition rate is 0% (4%) for texts (images) and that of vision API is 0% (0%).



(a) $d' = 10$", $\phi = 0°$ (b) $d' = 10$", $\phi = 45°$ (c) $d' = 28$", $\phi = 0°$ (d) $d' = 28$", $\phi = 45°$

Figure 15: Photos of protected images

8.7 Information Loss/Difference

We evaluate the information loss/difference of HideImage-protected images, especially the average information loss by computing the entropy differences [18] between the original and protected images. Note that the information loss of HideImage is due to converting the image into greyscale and partitioning the images into color layers. Since there is no information loss when switching color layers with grids, the information loss is independent of the viewing range of the protected image. We used 100 color images (512×512 size, 24-bit color-coded) for this evaluation. The average information loss is shown to be 4.66 bits with maximum (minimum) of 5.87 (1.79) bits.

We further evaluate the pixel-wise value differences between the original and the protected images as an indicator of how different the protected images appear from the original image. Table 4 shows the mean, maximum, and minimum of root-mean-square of the pixel-wise differences with various d_{max}. The pixel-wise differences are shown to be around 90 within normal viewing distances ($d_{max} = d + d_m = 24$") and gradually increase when d_{max} increases.

9 EVALUATION OF USER EXPERIENCE

We evaluate users' experience with HideScreen. Specifically, we evaluate (i) the readability of protected information, (ii) the amount of energy HideScreen consumes, and (iii) use-case study.

ℓ (pixel)	1	2	3	4	5	6	7
d_{max} (inch)	6.24	12.48	18.72	24.97	31.21	37.45	43.69
Mean	89.55	89.55	89.55	91.64	91.38	92.32	92.31
Max.	112.39	112.39	112.39	115.43	117.21	117.12	117.07
Min.	60.29	60.29	60.29	61.00	59.99	60.54	60.52

Table 4: This table shows the root-mean-square of the pixel-wise differences with various maximum viewing distance d_{max} based on the screen resolution of Pixel XL (534 ppi) .

	Simple Word T_C (s)	Medium Word T_C (s)	Difficult Word T_C (s)
Plain	1.29	1.22	1.40
Protected	1.88	2.07	2.75
Overhead	0.59 (46%)	0.85 (70%)	1.35 (96%)

Table 5: Comprehension time, T_C, for a single word

9.1 Evaluation of HideText Readability

We evaluate the time needed for users to comprehend (called *comprehension time* T_C) the protected texts; it is an indicator of how HideText affects user experience. Under an ideal condition, HideText should not increase the comprehension time much to provide good user experience. We use the same settings in Section 8.3 and record the comprehension time of both protected and (unprotected) plain texts. Table 5 summarizes the results of average comprehension time.

T_C for protected texts increases as word length increases. The overhead per word is reasonable for our target application with up to 96% for difficult words. Since HideText is targeting privacy-sensitive applications, such as messaging, and the lengths of sensitive parts are usually short (a maximum of 160 characters for SMS), users will not have any problem in reading the entire protected message for less than 1min.[7] The use of a single word can also be considered as the worst case in terms of reading time. Since no other context can be used to help the user predict the word, s/he can only identify the word by visual perception. Therefore, it will take the user longer to identify the word individually than in a sentence.

9.2 Energy Consumption and Latency

Since energy consumption is an important concern to smartphone users, we evaluate the energy consumption of HideScreen. We use Nexus 5X (Android 6.0) as the main device for this evaluation.

We consider a typical scenario where a user uses instant messaging apps implemented with HideScreen's API. So,

[7] Calculated based on the average reading speed of 987 characters/min [19] and the researchers have shown no significant effect of font size on reading speed [20–22].

HideScreen need not capture the original screenshot and process it before generating the protected information. In such a case, the main energy consumption stems from the estimation of viewing distance, which depends highly on the design choice of how frequently HideScreen performs the estimation. Table 6 summarizes the energy consumption incurred by the estimation of viewing distance. It is calculated when the brightness of the screen is adjusted to 50% and all communication functions (*i.e.*, WiFi, Bluetooth, and cellular) are turned off. Since users tend to maintain the same viewing distance, the estimation is done every 5min and the corresponding energy overhead is 3.39%.

Update Period (min)	1	3	5	10	30
OH. of Dist. Est. (%)	16.95	5.65	3.39	1.69	0.57
OH. w/ Processing (%)	18.09	6.03	3.62	1.81	0.60

Table 6: HideScreen's energy overhead

The other energy consumption comes from generating protected information. Since HideScreen can be implemented in a way that it stores the generated HideText characters and reuses them later, there will be no energy/latency overhead after their first use for text protection. Therefore, we consider the instant message displaying images of size 256×256. The second row of Table 6 lists the total energy overhead when images are received at the same rate of updating viewing distance. According to Facebook's statistics [23, 24], a Messenger user sends an average of 0.5 photo per day. Even if we consider the case where the user receives a photo every 5min, the energy overhead is only 3.62%. Since this overhead is compared to the condition with communication turned off, the energy overhead will be even lower in the normal usage scenario with cellular connection on. We thus conclude that HideScreen's energy overhead is very low while providing good user experience.

During this evaluation, we also measured the latency of generating protected images. We use different-size images and repeat the HideImage generation 20 times. The average latency of generating a 128×128 image is 131ms, and that of a 256×256 and a 512×512 image are 533 and 1684ms, respectively. This latency is reasonable for apps like instant messages to provide good user experience since the main delay for these applications is the time for delivering the information from one device to another, and all 10 participants who provided feedback on the latency agree that it is short enough for good user experience.

9.3 Use-Case Study

9.3.1 Preliminary User Feedback. After participants tried the basic functions of HideScreen, we asked their opinions about the protection of sensitive OSI and their feedback on HideScreen's user experience. While 16 (80%) of participants indicate they want to protect their OSI, only 5 of them stated that they *may* buy/use a privacy film for their mobile devices and the rest clearly stated they will not use a privacy film. After trying HideScreen, 90% (45%) of the participants reported HideText (HideImage) is moderately easy, or easy to use for comprehension and protection of OSI. All participants are willing to use HideScreen for at least one type of applications for information protection, and password entering (80%), texting/messaging (50%), and unlocking phones (50%) are the top three apps of HideScreen. Based on this preliminary feedback, we implemented three prototypes with HideScreen's protection (*i.e.,* account login, messaging, and PIN entering) for the evaluation of practical use-cases.

9.3.2 Evaluation Settings. Since System Usability Scale (SUS) [25, 26] has proven to be a robust tool for evaluating user experience, we chose it as the evaluation metric. Our average SUS scores are compared against the average SUS score (=68/100) in [27] and are transformed to adjective user experience (UX) ratings [26] to see if those applications can provide good user experience. The larger the SUS, the better the UX.

We recruited 25 participants most of whom also participated in the previous evaluation. For each application, we conducted a usability study by taking the following steps. First, we introduce the functions of the application to the participants. Second, we ask the participants to perform certain tasks as RU. Third, one of the authors acts as RU and the participant acts as a SS who tried to acquire the sensitive OSI. We use the same settings as in Fig. 14, where $d_m = 12"$. Finally, we ask participants to fill out SUS evaluation form.

We evaluate how HideScreen can be integrated into (bank) account login process by using the UI shown in Figs. 16(a) and (b). During the evaluation, participants were asked to perform the following tasks: (i) input their school ID in the account field (region A) using the HideScreen-protected and randomly shuffled keyboard (region F), (ii) choose HideImage (Case I) or SelImage (Case II) using the buttons in region B to show the security image in region C, and (iii) input password (region D) using the same HideScreen-protected keyboard. After completing the tasks, participants were asked to fill out two SUS forms, where one is for the case of using HideImage and the other is for the case of using SelImage. When acting as a shoulder surfer, the participants were asked to identify the characters on the keyboard and the account.

Fig. 16(c) shows the UI for PIN entering, where the numbers are randomly shuffled and HideText-protected. Participants were asked to enter PIN for at least two times and act as a SS to identify the numbers on the keypad.

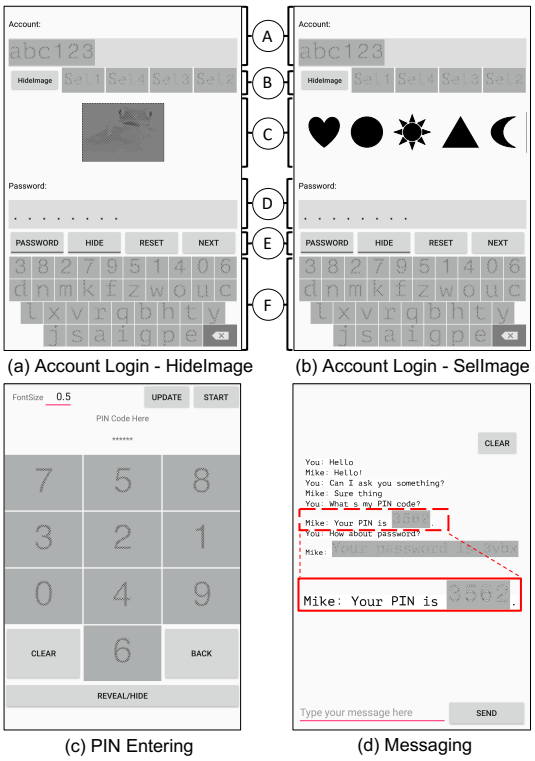


Figure 16: Testing UI used in use-case study

Fig. 16(d) shows the testing UI of a HideScreen-supported messenger, where the participants are able to send protected messages by enclosing the target texts with '{' and '}'. In this use-case, we adopt a smaller text size (≈14sp) for protected texts. The participants were asked to interact with a simple chatbot, which simulates the condition of exchanging some sensitive information. During the evaluation, participants will receive some protected information and are asked to send messages with protected contents, including both partially and completely protected messages.

9.3.3 Results. Table 7 summarizes the results of our use-case study, showing that all use-cases have SUS scores greater than the average SUS score (=68). Furthermore, when transformed to adjective UX ratings, 3 out of 4 use-cases are able to provide good (71.4) to excellent (85.5) user experience [26]. It is worth noting that the participants' feedback on applying HideScreen to the messaging app is very positive and they all agree that it is a very useful function.

Even though the SUS of account login seems to be lower than others, the participants indicate that it is mainly due to the shuffling of the keyboard that affects the user experience, but not HideScreen itself. However, the keyboard must be shuffled to provide protection in this use-case, else shoulder surfers will be able to obtain the user's password by observing the position where the user clicked. This is a tradeoff between convenience and privacy protection.

We also asked participants whether they would use these apps frequently in future after trying the prototypes. Their willingness of using HideScreen is similar to, or higher than their preliminary feedback, meaning that the effect of HideScreen and its user experience is similar to, or better than their expectation when applied to real usage scenarios. Furthermore, for PIN entering and messaging, participants' interest in using HideScreen increases noticeably after trying the prototypes, and only one participant indicates that he will not use the protection frequently. Since no participants are able to retrieve the protected OSI during the use-case study, we conclude that HideScreen is able to provide moderate or excellent user experience while providing the protection of information.

Application	SUS	Adjective UX Rating	May/Will Use Frequently (%)
Account Login (I)	71.0	OK - Good	72.0
Account Login (II)	77.6	Good - Excellent	92.0
PIN Entering	79.3	Good - Excellent	96.0
Messaging	84.7	Good - Excellent	96.0

Table 7: Results of use-case study

10 DISCUSSION

We discuss some relevant topics that have not been covered in this paper.

10.1 Deployment of HideScreen

There are several ways to deploy HideScreen in devices:

HideScreen **API**: HideScreen can be deployed as a library/API that provides a special "view" to display sensitive information. Developers can use this API to exercise fine-grained control over whether or not to protect certain information. This method has the most flexibility for developers to provide the best user experience.

Built-in Function in OS: OS vendors can deploy HideScreen as a built-in function in their OS. Once HideScreen is enabled, every piece of information that a user requests will be processed and protected before it is displayed on the screen. This deployment can provide system-level protection without the support from app developers, but requires OS modifications.

Stand-Alone App: Without requiring any modification to the mobile device OS, HideScreen can be deployed as a stand-alone app and acts as a software filter applied to on-screen information. After the user grants necessary permissions, this stand-alone app will first block all the information shown on the screen, capture the screenshot, process the contents, and finally generate the protected contents for display.

10.2 Limitations and Future Work

Discussed below are the limitations of current version of HideScreen. First, the protected OSI will be harder for legit users to read than the original OSI, so HideScreen is not suitable for reading a long text/article. Also, since HideText utilizes edge replacement to boost readability, the minimum size of the resulting protected information is bounded by this mechanism. That is, the stroke width of a protected text must be greater than 6 pixels for proper protection and readability enhancement. If the original OSI is too small, HideScreen may scale the OSI to the minimum applicable size to provide its full protection capability. For the image protection, HideImage will sacrifice the color information and some high-frequency components of images as a tradeoff for privacy protection, which is not suitable for apps that require users to examine details of images.

Protection of color images and videos is a natural extension of HideScreen which is part of our future work. To protect color images, we can utilize the bright components of the grids to represent the original color and utilize its complementary color as the dark components to make the grids look like another color when viewing from far away. Since human eyes have different sensitivities to different colors, how to adjust the colors so that the protected images will look more natural is an important topic to be explored.

Theoretically, HideImage can be applied to create a protected video since each video is composed of image frames, but there are some special video properties that may make the protection difficult. For example, device screens may have different display mechanisms while playing consecutive frames, incurring unexpected visual effects. Last but not the least, how to efficiently process live stream or chat videos while incurring minimum delay and energy overheads is a practical challenge that needs to be investigated.

11 RELATED WORK

11.1 Authentication Protection

Authentication protection focuses on the protection of authentication secrets (*e.g.*, PIN). Its goal is to prevent shoulder surfers from obtaining the authentication secrets from the information displayed on the device screen so that they cannot access the legitimate user's device in his absence. It usually adopts some special UI by (i) displaying decoy information on the screen [28–31], (ii) asking users to authenticate devices with the information derived from the authentication secret [32–34], or (iii) adopting multiple channels to display/perform the authentication process [35–45].

Since the authentication protection schemes usually involve dedicated UI to protect authentication secrets, most of them cannot be used for general information, such as texts or images. One exception is IllusionPIN [7], which preserves the high-frequency components (HFCs) of OSI and eliminates the low-frequency components (LFCs) for privacy protection. This approach is more suitable than HideScreen for apps that require inspection of HFCs. However, its protection mainly depends on the frequency components of the OSI itself and, therefore, the OSI needs to be properly formatted to meet the app requirements to ensure that the required information will not be removed along with LFCs. In contrast, HideScreen's protection is achieved by replacing the LFCs that are easily seen by others with HFCs, instead of their elimination, thus making it more flexible to handle different types of information.

11.2 General Information Protection

We introduce three most commonly used schemes in this category.

Interpretation barrier [46, 47] slows down the information leak by displaying the OSI in specific formats that only the user can understand easily, but shoulder surfers cannot. It aims at slowing down, not preventing, the OSI leakage.

Shoulder surfer alert [5, 48] utilizes sensors, such as cameras, to detect whether there are any people around the user trying to peek at the user's device screen.

Information blocking [6, 49–51] hides the information on the screen by dimming the screens or removing the information on the screen. The latter two schemes usually require additional hardware, or directly block either partial or entire on-screen information, which limits their protection or prevents users from performing certain tasks.

12 CONCLUSION

We have proposed HideScreen to protect the information displayed on device screens from shoulder surfers. Specifically, HideScreen is comprised of HideText, HideImage, and SelImage. These protection schemes are grounded on optical system properties and human vision characteristics, which provide fundamental protection guarantees. Grids are used to replace low-frequency components so that the information may not be distinguished from the background when viewed from the outside of the designed viewing range. Shoulder surfers or malicious parties will not be able to read the information on the screen while the user can read it without difficulty. Our extensive evaluation has shown HideScreen to be able to provide a high rate of on-screen information protection ($> 96\%$ for texts and $> 99\%$ for images) while incurring low overhead. Furthermore, our use-case study shows that HideScreen achieves good user experience while providing privacy protection.

ACKNOWLEDGEMENT

The work reported in this paper was supported in part by the NSF under grants CNS-1505785 and CNS-1646130.

REFERENCES

[1] Malin Eiband, Mohamed Khamis, Emanuel von Zezschwitz, Heinrich Hussmann, and Florian Alt. Understanding Shoulder Surfing in the Wild. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*, pages 4254–4265, New York, New York, USA, 2017. ACM Press.

[2] Adam J. Aviv, John T. Davin, Flynn Wolf, and Ravi Kuber. Towards Baselines for Shoulder Surfing on Mobile Authentication. In *Proceedings of the 33rd Annual Computer Security Applications Conference on - ACSAC 2017*, pages 486–498, New York, New York, USA, 2017. ACM Press.

[3] Polotiko. Aguirre furious at photo leak of private text message. http://politics.com.ph/aguirre-furious-photo-leak-private-text-message/, 2017. Accessed: 2018-01-31.

[4] HP. HP Introduces World's Only Notebooks with Integrated Privacy Screens. https://press.ext.hp.com/us/en/press-releases/2016/hp-introduces-worlds-only-notebooks-with-integrated-privacy-scre.html, 2016. Accessed: 2018-07-28.

[5] Hee Jung Ryu and Florian Schroff. Electronic Screen Protector with Efficient and Robust Mobile Vision Demo Video. https://nips.cc/Conferences/2017/Schedule?showEvent=9757, 2017. Accessed: 2018-02-01.

[6] BlackBerry Limited. BlackBerry Privacy Shade - Android Apps on Google Play. https://goo.gl/MFGeAX, 2016. Accessed: 2018-02-01.

[7] Athanasios Papadopoulos, Toan Nguyen, Emre Durmus, and Nasir Memon. IllusionPIN: Shoulder-Surfing Resistant Authentication Using Hybrid Images. *IEEE Transactions on Information Forensics and Security*, 12(12):2875–2889, Dec 2017.

[8] Aude Oliva, Antonio Torralba, and Philippe G. Schyns. Hybrid images. In *ACM SIGGRAPH 2006 Papers on - SIGGRAPH '06*, volume 25, pages 527–532, New York, New York, USA, 2006. ACM Press.

[9] NASA. Anthropometry and Biomechanics. https://msis.jsc.nasa.gov/sections/section03.htm, 2000. Accessed: 2017-11-27.

[10] Michitaka Yoshimura, Momoko Kitazawa, Yasuhiro Maeda, Masaru Mimura, Kazuo Tsubota, and Taishiro Kishimoto. Smartphone viewing distance and sleep: an experimental study utilizing motion capture technology. *Nature and science of sleep*, 9:59–65, 2017.

[11] Wikipedia. Airline seat, 2017. Accessed: 2017-12-29.

[12] Devraj Singh. *Fundamentals of optics.* Prentice-Hall Of India, 2015.

[13] J. Mannos and D. Sakrison. The effects of a visual fidelity criterion of the encoding of images. *IEEE Transactions on Information Theory*, 20(4):525–536, Jul 1974.

[14] Latanya Sweeney. k-Anonymity: A Model For Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, Oct 2002.

[15] Jakub Dostal, Ola Kristensson, and Aaron Quigley. Estimating and using absolute and relative viewing distance in interactive systems. *Pervasive and Mobile Computing*, 10:173–186, 2014.

[16] Tech Armor. Tech Armor Website. https://techarmor.com/, 2018.

[17] Google. Vision API – Image Content Analysis. https://cloud.google.com/vision/, 2018. Accessed: 2018-01-29.

[18] Ze-Nian Li, Mark S. Drew, and Jiangchuan Liu. *Introduction to Multimedia.* Springer, 2014.

[19] Susanne Trauzettel-Klosinski and Klaus Dietz. Standardized Assessment of Reading Performance: The New International Reading Speed Texts IReST. *Investigative Opthalmology & Visual Science*, 53(9):5452, Aug 2012.

[20] David Beymer, Daniel Russell, and Peter Orton. An Eye Tracking Study of How Font Size and Type Influence Online Reading. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction - Volume 2*, pages 15–18. British Computer Society, 2008.

[21] Iain Darroch, Joy Goodman, Stephen Brewster, and Phil Gray. The Effect of Age and Font Size on Reading Text on Handheld Computers. pages 253–266. Springer, Berlin, Heidelberg, 2005.

[22] Mark C Russell and Barbara S Chaparro. Exploring Effects Of Speed And Font Size With RSVP. In *Proceedings of the Human Factors And Ergonimics Society 45th Annual Meeting*, 2001.

[23] Facebook. Making Visual Messaging Even Better – Introducing High Resolution Photos in Messenger. https://newsroom.fb.com/news/2017/11/making-visual-messaging-even-better-introducing-high-resolution-photos-in-messenger/, 2017. Accessed: 2018-01-29.

[24] Facebook. We now have over 1.2 billion people actively using Messenger every month. https://goo.gl/diJ4t4, 2017. Accessed: 2018-01-29.

[25] John Brooke. SUS - A quick and dirty usability scale Usability and context. *Usability evaluation in industry*, 1986.

[26] Aaron Bangor, Philip Kortum, and James Miller. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *Journal of Usability Studies*, 4(3):114–123, 2009.

[27] Jeff Sauro. MeasuringU: Measuring Usability with the System Usability Scale (SUS), 2011.

[28] W; Eekelen, J; Van Den Elst, J V Khan, Wouter Van Eekelen, John Van Den Elst, and Vassilis-Javed Khan. Dynamic layering graphical elements for graphical password schemes. *Proceedings of the Chi Sparks 2014 Conference*, pages 65–73, 2014.

[29] Haichang Gao, Zhongjie Ren, Xiuling Chang, Xiyang Liu, and Uwe Aickelin. A New Graphical Password Scheme Resistant to Shoulder-Surfing. In *2010 International Conference on Cyberworlds*, pages 194–199. IEEE, Oct 2010.

[30] Jan Gugenheimer, Alexander De Luca, Hayato Hess, Stefan Karg, Dennis Wolf, and Enrico Rukzio. ColorSnakes: Using Colored Decoys to Secure Authentication in Sensitive Contexts. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI '15*, pages 274–283, New York, New York, USA, 2015. ACM Press.

[31] Nur Haryani Zakaria, David Griffiths, Sacha Brostoff, and Jeff Yan. Shoulder surfing defense for recall-based graphical passwords. In *Proceedings of the Seventh Symposium on Usable Privacy and Security - SOUPS '11*, New York, New York, USA, 2011. ACM Press.

[32] Alexander De Luca, Katja Hertzschuch, and Heinrich Hussmann. ColorPIN — Securing PIN Entry through Indirect Input. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*, pages 1103–1106, New York, New York, USA, 2010. ACM Press.

[33] Volker Roth, Kai Richter, and Rene Freidinger. A PIN-entry method resilient against shoulder surfing. In *Proceedings of the 11th ACM conference on Computer and communications security - CCS '04*, pages 236–245, New York, New York, USA, 2004. ACM Press.

[34] Susan Wiedenbeck, Jim Waters, Leonardo Sobrado, and Jean-Camille Birget. Design and evaluation of a shoulder-surfing resistant graphical password scheme. In *Proceedings of the working conference on Advanced visual interfaces - AVI '06*, pages 177–184, New York, New York, USA, 2006. ACM Press.

[35] Abdullah Ali, Ravi Kuber, and Adam J Aviv. Developing and evaluating a gestural and tactile mobile interface to support user authentication. In *IConference 2016 Proceedings*, 2016.

[36] 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 *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction - TEI '11*, pages 197–200, New York, New York, USA, 2011. ACM Press.

[37] Alexander De Luca, Martin Denzel, and Heinrich Hussmann. Look into my eyes!: can you guess my password? In *Proceedings of the 5th*

Symposium on Usable Privacy and Security - SOUPS '09, New York, New York, USA, 2009. ACM Press.

[38] Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. Touch me once and I know it's you!: implicit authentication based on touch screen patterns. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*, pages 987–996, New York, New York, USA, 2012. ACM Press.

[39] Alexander De Luca, Marian Harbach, Emanuel von Zezschwitz, Max-Emanuel Maurer, Bernhard Ewald Slawik, Heinrich Hussmann, and Matthew Smith. Now you see me, now you don't: protecting smartphone authentication from shoulder surfers. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*, pages 2937–2946, New York, New York, USA, 2014. ACM Press.

[40] Alexander De Luca, Emanuel Von Zezschwitz, and Heinrich Hußmann. VibraPass - Secure Authentication Based on Shared Lies. In *CHI*, pages 913–916, 2009.

[41] Alain Forget, Sonia Chiasson, and Robert Biddle. Shoulder-surfing resistance with eye-gaze entry in cued-recall graphical passwords. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*, pages 1107–1110, New York, New York, USA, 2010. ACM Press.

[42] Mohamed Khamis, Florian Alt, Mariam Hassib, Emanuel von Zezschwitz, Regina Hasholzner, and Andreas Bulling. GazeTouchPass. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '16*, pages 2156–2164, New York, New York, USA, 2016. ACM Press.

[43] Behzad Malek, Mauricio Orozco, and Abdulmotaleb El Saddik. Novel Shoulder-Surfing Resistant Haptic-based Graphical Password. In *Proc. EuroHaptics, Vol. 6*, 2006.

[44] Toan Van Nguyen, Napa Sae-Bae, and Nasir Memon. DRAW-A-PIN: Authentication using finger-drawn PIN on touch devices. *Computers & Security*, 66:115–128, May 2017.

[45] Christian Winkler, Jan Gugenheimer, Alexander De Luca, Gabriel Haas, Philipp Speidel, David Dobbelstein, and Enrico Rukzio. Glass Unlock: Enhancing Security of Smartphone Unlocking through Leveraging a Private Near-eye Display. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, pages 1407–1410, New York, New York, USA, 2015. ACM Press.

[46] Malin Eiband, Emanuel von Zezschwitz, Daniel Buschek, and Heinrich Hußmann. My Scrawl Hides It All. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '16*, pages 2041–2048, New York, New York, USA, 2016. ACM Press.

[47] Emanuel von Zezschwitz, Sigrid Ebbinghaus, Heinrich Hussmann, and Alexander De Luca. You Can't Watch This!: Privacy-Respectful Photo Browsing on Smartphones. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*, pages 4320–4324, New York, New York, USA, 2016. ACM Press.

[48] Mohammed Eunus Ali, Tanzima Hashem, Anika Anwar, Lars Kulik, Ishrat Ahmed, and Egemen Tanin. Protecting Mobile Users from Visual Privacy Attacks. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct Publication - UbiComp '14 Adjunct*, pages 1–4, New York, New York, USA, 2014. ACM Press.

[49] Frederik Brudy, David Ledo, Saul Greenberg, and Andreas Butz. Is Anyone Looking? Mitigating Shoulder Surfing on Public Displays through Awareness and Protection. In *Proceedings of The International Symposium on Pervasive Displays - PerDis '14*, pages 1–6, New York, New York, USA, 2014. ACM Press.

[50] Shiguo Lian, Wei Hu, Xingguang Song, and Zhaoxiang Liu. Smart privacy-preserving screen based on multiple sensor fusion. *IEEE Transactions on Consumer Electronics*, 59(1):136–143, Feb 2013.

[51] Peter Tarasewich, Richard Conlan, and Jun Gong. Protecting Private Data in Public. In *Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems*, 2006.