

Shoulder-surfing resistant graphical passwords — Draft

Leonardo Sobrado, Jean-Camille Birget *

April 11, 2005

Abstract

We describe two graphical password schemes that are resistant to shoulder-surfing, i.e., to attacks by direct observation during login. These schemes are based on challenge-response interactions that can be carried out by humans without computer aid. Moreover, we include results of a human feasibility study for the first scheme.

Keywords: Graphical passwords, shoulder surfing, computer security.

1 Introduction

1.1 Background on graphical passwords

Graphical passwords were first introduced by Greg Blonder [2]. In his scheme, when a user selects a password, an image is presented in which the user chooses a subset from a predefined set of click regions in the image. In order to log in later, the user must click in the same regions as he or she selected earlier. Implementations of this idea have been given by Passlogix Co. [3]. A version of click regions that includes movement was introduced by Brad Isaacson [6]. A different type of graphical password schemes, based on recognition of images from a sequence, can be found in [12], [4], and [11]. An enlightening analysis of graphical passwords appears in [7]. In [1], Blonder's graphical passwords are generalized so as to let users choose arbitrary points as click regions (as opposed to predefined click regions in a pre-processed image).

1.2 Shoulder-surfing

Passwords, alphanumeric and graphical alike, are vulnerable to shoulder-surfing. *Shoulder-surfing* consists of watching over people's shoulders as they enter or view information. More generally, shoulder-surfing can also include recording a user's login session with a camera (paradoxically, security cameras can add to the insecurity of passwords), or using key-logging software and trojan software, or remote electro-magnetic sensors to capture user actions. Moreover, "sniffing" tools that obtain information from a user's network communications can also be viewed as a form of shoulder-surfing. We make the very liberal assumption that the adversary

*Both authors were supported by NSF grant CCR-0310793.

can film a very large number of login sessions and record them in detail. None of the graphical or alphanumeric password schemes mentioned earlier can resist to this kind of attack, and it seems surprising that shoulder-surfing resistant passwords can exist at all.

Earlier work of ours on shoulder-surfing [13] outlines the ideas of a few shoulder-surfing resistant graphical password schemes, and proves that it is indeed possible to have shoulder-surfing resistant password schemes that can be used by humans. Our schemes are based on a form of challenge-response authentication that humans are able to use without computer help; see [9] for general background on challenge-response authentication. In the present paper we work out the details of some of the ideas introduced in [13] and add improvements and variants. Our main goal is to create authentication protocols that prevent an adversary from doing any significant learning from the recordings of several login sessions. In addition, we consider possible ways to reduce login time.

In the next sections we develop two shoulder-surfing resistant password schemes, one based on clicking in the convex hull of secret icons (Section 2), and another one based on relations between an image and a number grid (Section 3).

2 Convex-hull click scheme

Our first scheme is a more detailed version of the main scheme outlined in [13].

Password selection: The system displays a large number (say N) of *icons* on the screen. To create a password, the user selects k icons among these N . The user and the system must remember these k icons and keep them secret. We call those k icons “the user’s *pass-icons*”. The selection of pass-icons must be done in an environment free of shoulder-surfing adversaries.

Login: (a) *System’s actions:* When a user wants to log in, the system challenges the user as follows. The system randomly chooses j in the range $3 \leq j \leq k$. It then chooses j icons among the k pass-icons, uniformly at random, and “randomly” places them on the screen, along with many more non-pass icons; let n ($\leq N$) be the total number of icons displayed on the screen. We will discuss the random placement of icons below.

Note that in a challenge, the system does not display all the icons nor all the pass-icons; in each challenge (usually) different subsets of the icons and the pass-icons are displayed. Practical values of the parameters might be $N = 500$, $n = 250$, $k = 10$. For the random selection of j , we pick j in the range $[3, k]$, giving higher probabilities to the lower values, so that the average value j_{ave} of j is around 5. This way, $\frac{j_{ave}}{k} = \frac{n}{N}$, i.e., the frequencies $\frac{j}{k}$ and $\frac{n}{N}$ are equal on average; so, pass-icons would appear as frequently on the screen as non-pass-icons, on average. More generally, we could let n fluctuate too, but in such a way that $\frac{j_{ave}}{k} = \frac{n_{ave}}{N}$.

(b) *User’s actions:* The user is expected to respond to this challenge by clicking (with a mouse or a stylus) anywhere inside the *convex hull* of the j displayed pass-icons. The convex hull of a set of pass-icons is, by definition, the smallest convex surface on the screen that contains these pass-icons. Note that clicking in the convex hull of the displayed pass-icons is equivalent to clicking inside the (roughly triangular) region spanned by any three of these pass-icons.

We will always talk about the locations of icons as if they were points. Although our icons are not points, we can pick some fixed point in each icon and anchor the icon on the screen at that point (e.g., pick the barycenter of each icon).

(c) *Repeat*: The above challenge-response interaction is repeated h times to avoid accidental or random logins; a typical value is $h = 10$. The user has to pass all h challenges (and miss none) in order to log in. In each challenge, the system chooses (most likely) a different value for j , and a different set of j pass-icons among the k pass-icons. Moreover, the system should limit the number of failed logins that a user can make in a day (to e.g., 4).

In summary, here are the parameters:

- N is the total number of icons available; e.g., $N = 1000$ or $N = 500$.
- n is the total number of icons displayed on the screen in a particular challenge; e.g., $n = 500$ or $n = 250$.
- k is the number of secret pass-icons chosen at password selection; e.g., $k = 10$.
- j (with $3 \leq j \leq k$) is the number of pass-icons included in a particular challenge (it changes randomly). We require that $j/k = n/N$ on average.
- h is the number of consecutive successful challenges that the user must pass in order to log in; e.g., $h = 10$ or 15 .

Design and security issues

(1) Random placement of icons on the screen: We have to solve the problem that if the j displayed pass-icons are placed independently and uniformly at random, chances are rather high that the center of the image is in the convex hull of the pass-icons (especially if j is not very small). This would greatly increase the chance of **accidental login**. In order to avoid this problem we want to make sure that all locations on the screen have roughly similar probabilities of being in the convex hull of the pass-icons. A precise formulation of the problem is as follows.

Placement problem: Find a probabilistic algorithm for placing j points (called “pass-points”) in a rectangle of size $a \times b$ according to a probability distribution such that: (1) Every location on the screen, except for locations close to the borders of the screen, has approximately the same probability of being in the convex hull of the j pass-points. (By “close” to the borders we mean within distance $\leq \frac{a}{10}$ of the border of length b , and within distance $\leq \frac{b}{10}$ of the border of length a . By “approximately the same probability” we mean that the probability of one point is no more than double the probability of any other point.) (2) A non-negligible portion of the j -tuples of locations on the screen have approximately the same probability of being chosen for the placement of the j pass-points to be displayed. More precisely, there is a set of j -tuples S with probability $P(S) > \alpha$ (for some fixed $\alpha > 0$) such that for any two j -tuples T_1 and T_2 in S we have the following; let p_i be the probability that T_i consists of the j pass-points, for $i = 1, 2$; then $\frac{1}{4}p_1 \leq p_2 \leq 4p_1$. (The choice of “4” and “ $\frac{1}{4}$ ” is somewhat arbitrary; we just want to make sure that the probabilities p_1 and p_2 are not too different.)

We give two probabilistic placement algorithms.

1. *Out-of-the-shadow placement:* The system first places two pass-points anywhere on the screen, independently and uniformly; let P_1 and P_2 be the chosen positions of these two pass-points. Let O be the center of the screen (assumed to have the shape of a rectangle). The system places the 3rd pass-point with a probability distribution that gives low probability to the half-cone with vertex O and boundary lines P_1O , P_2O , away from P_1P_2 (we call this half-cone “the *shadow* of O relative to P_1P_2 ”); see Fig. 1. More precisely, the probability of any location in the shadow region is s_1 times the probability of any location in the non-shadow region (for some parameter $0 < s_1 < 1$; e.g., $s_1 = 0.25$). Similarly, the system places the other

$j - 3$ pass-points with a probability distribution that gives low probability to the shadow of O relative to the positions where pass-points have already been placed; moreover, the shadow of three points has a lower probability than the shadow of two points; and the shadows of more points get yet lower probabilities. To be more precise, let's say that every location in a double shadow has s_2 times the probability of any location in a single shadow region (for some parameter $0 < s_2 < 1$; e.g., $s_2 = 0.25$); similarly, we parameterize the probabilities of placement in triple, quadruple, etc. shadows by s_3, s_4 , etc.

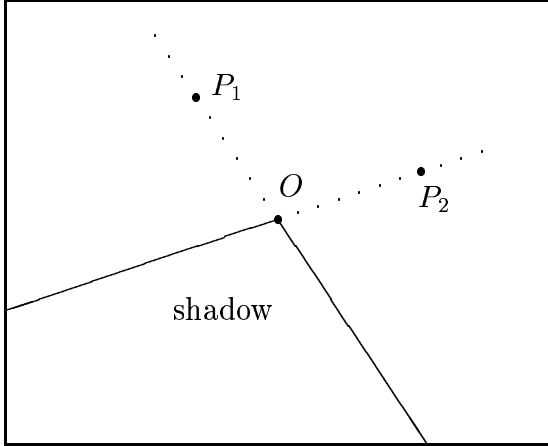


Figure 1: The shadow of O relative to P_1P_2

Another probabilistic placement algorithm is as follows.

2. *Grid-based adaptive placement:* The screen is divided into g grid squares (e.g., $g = 300$ squares). For each square the system keeps track of how often the square has overlapped with the convex hull of the j displayed pass-icons over the past t placements (e.g., $t = 200$); this defines an empirical tagging probability for each square. To place the j pass-icons, the system picks a fixed number p of placements at random (e.g., $p = 10$ or 20), and then among those it picks a placement that has a maximal “uniformization effect” on the grid of tagging probabilities. (We measure the proximity of a distribution $(p(x, y) : (x, y) \in G)$ to the uniform distribution $(p_{\text{unif}}(x, y) : (x, y) \in G)$ by taking $\max\{|p(x, y) - p_{\text{unif}}(x, y)| : (x, y) \in G\}$, where G denotes the square grid.)

Moreover, before deciding on one of the p generated placements, the system discards placements that produce too large an area for the convex hull of the j pass-icons (say more than 40% of the screen area), or too small an area (e.g., less than 5%).

The mathematical analysis of the properties of these probabilistic placement algorithms, as well as the task of developing better ones, remains a problem for future research. An important consideration is that the parameters used in the placement algorithms can be tuned so that the expected area of the convex hull of the j pass-points is between 0.1 and 0.3 as a fraction of the total area of the screen. Moreover, the placement should be such that the area of the convex hull is never too small, otherwise it is too hard for the user to click in it.

(2) Accidental login: In any challenge-response authentication system there is a possibility that an attacker successfully responds to the challenges without actually knowing the secret.

The difference between accidental login and knowing the secret is that an attacker cannot repeat an accidental login at will, and hence is not in a position to change the password. In our system (as in pretty much all challenge-response systems), accidental login is much more likely than guessing the secret, and must therefore be addressed.

The probability that a random location on the screen is in the convex hull of the j displayed (but secret) pass-icons can be assumed to be at most 0.3 (by tuning the parameters as we just saw in (1) above). To make accidental login unlikely, we require a fairly large number h of challenge-response rounds between the system and the user for each login (e.g., $h = 10$ or more). If the probability of an accidental login in one interaction is < 0.3 then for h interactions it is $< (0.3)^h$. When $h = 10$ this is $< 6 \times 10^6$; when $h = 20$, it is $< 3.5 \times 10^{11}$.

Moreover, the system should not allow more than 4 false logins in a day for each user. This way, with $h = 10$ it would take $\frac{1}{2} \times 6 \times 10^6 / 4 \approx 750\,000$ days (≈ 2053 years) on average for an accidental login. For $h = 20$ it would take 4.38×10^{10} days on average (> 120 million years). However, although this shows that accidental logins in a particular user's account is unlikely, an attacker can still try a collective attack against all accounts in a system. E.g., if a system has 1000 users, and an attacker can do 4 false logins in a day per user, it will now take 1000 times less time for the collective attack to succeed; for $h = 10$ it would take ≈ 2.05 years on average for an accidental login in some account, and for $h = 20$ it would take $> 120\,000$ years. So, $h = 10$ may be too low for security.

(3) The password space of this scheme is large: The total number of possible k -element sets is the binomial coefficient $\binom{N}{k} = \frac{N!}{k!(N-k)!}$. We estimate that by using graphic composition software we can place 500 icons of size 40×40 pixels on a 17 inch monitor, set to a screen resolution of 1024×768 . At these settings each icon retains enough detail to be distinguishable from all other icons. Naturally, bigger monitors set to higher resolutions would be able to display both more detailed and more numerous icons. For example, if $k = 10$ and $N = 500$, the total number of possible “passwords” is $\binom{500}{10} \approx 2.5 \times 10^{20}$; if $k = 10$ and $N = 1000$ the number of possible “passwords” is $\binom{1000}{10} \approx 2.6 \times 10^{23}$. Thus, these password spaces are too large for an attacker to be able to guess the correct password.

(4) Brute-force attacks: Here the attacker tries to discover the password (i.e., the k pass-icons) by searching exhaustively through certain sets of collections of icons. We assume that the attacker knows how the system was designed, and that the attacker can observe and record several login sessions of the legitimate user. We can also assume that the attacker knows the total set of N icons.

We assume that the attacker has made a guess of the total number of pass-icons k and of the number j of currently displayed pass-icons. Since k cannot be very large $k \leq 100$ (it would be very inconvenient for a user to remember even 100 random icons), and since $3 \leq j \leq k$, the attacker could exhaustively try all pairs (j, k) ; let us now consider a fixed guess of (j, k) . Recall that for a given password, k is fixed, but j varies with every login challenge.

When the attacker observes a click point, this determines the set of all j -tuples of icons that do or do not contain the click point; conversely, the set of triples that contain (or do not contain) the click point determines the click point with as much information as we could wish (indeed, the only information that matters for logging in is the set of convex hulls that contain the click point). In a brute-force attack the attacker tries to record a list of all j -tuples of icons that contain the click point, as well as a list of all the j -tuples that do not contain the click

point. However, since j varies, $\sum_{j=3}^k \binom{N}{j}$ sets of icons would have to be recorded, which is too large to be feasible when $k \geq 10$ and $N \geq 500$.

Note however that if j were kept fixed and small, the brute-force attack would be possible (e.g., with $j = 4$ and $N = 1000$ we have $\binom{N}{j} \approx 4.1 \times 10^{10}$). Therefore it is necessary to let j vary from challenge to challenge, and hide the current value of j . It will be acceptable to have $3 \leq j \leq 5$ most of the time, provided that in every few screens j is larger, up to k .

Remarks:

(1) A great inconvenience of the graphical password scheme described above is that in order to be able to make a valid click, the user must find at least 3 icons among hundreds on the screen. One may think that this could end up resembling a “Where is Waldo?” game, and could be tedious. However, the “Where is Waldo?” games have actually thousands of little figures; our user only has to search through hundreds, which is significantly easier – but nevertheless tedious.

(2) Another great inconvenience is that the system gives the user h challenges (e.g., $h = 10$ or more). This succession of challenges is tedious. In order to make the switch-over from one screen to the next more agreeable to the user, the system can do the following: 1. The icons that are removed from the screen leave at the upper left corner, and new icons enter at the lower right corner. 2. The icons that remain on the screen are moved to their new positions first horizontally then vertically, at a speed that is slow enough so that the user can follow the movement. The main advantage of this procedure is that the user can follow where previously seen pass-icons move; this will facilitate the next challenge for the user.

(3) A drawback from the system’s point of view is that the password is not cryptographically hashed. In fact, it seems to be very difficult to find a password scheme that combines challenge-response (for shoulder surfing resistance) and password hashing. So at this time we have to assume that the system knows the user’s pass-icons explicitly and that it is able to keep them secret.

(4) Extension of the above password system: One can require that the user click in the convex hull, or outside the convex hull, depending on the presence or absence of certain special secret pass-icons.

A variation: Dragging the screen

This scheme uses the same parameters as the scheme above. Again, the user is expected to indicate the convex hull of the j displayed pass-icons. But this time, instead of explicitly clicking in the convex hull, the user “drags the screen” so as to move a secret *pointer icon* into the convex hull of the j displayed pass-icons. To *drag the screen* means that a second copy of the screen (drawn in a shadowy or semi-transparent manner) is shifted or translated, while the original copy of the screen remains in place; in its movement, the shadow copy of the screen is wrapped around like a torus (top-bottom, left-right). In this scheme we have two kinds of secret icons that the user and the system remember: the pass-icons, and the pointer icons. In each challenge, the system displays j (≥ 3) pass-icons and at least one pointer icon. The user is expected to drag the screen so that a pointer icons (in the shifted copy of the screen) has been transported into the convex hull of the j displayed pass-icons (shown in the fixed first copy of the screen).

The advantage of this method is that the attacker doesn’t see any click point, and hence, receives much less information about the convex hull.

3 Image with numbers

In this scheme, the system displays an image on the screen and a square grid, superposed (unobtrusively) on the image. Each square in the grid could have an area of $2 \times 2 \text{ mm}^2$, for example.

In *password selection*, the user chooses k grid squares (e.g., $k = 10$) and keeps them secret. Let us call those k squares “the pass-squares”; the corresponding places in the image are called “pass-places”. The user should choose squares that correspond to places in the image that are easy to remember. The role of the image is only to help the user remember the k chosen grid-squares. During password selection, the user could also decide to choose a different image, from a data base of images, or from the user’s own collection.

At *login*, the system gives the user a sequence of challenges of the following form: In every grid-square, the system writes an integer, randomly chosen in the range $[0, N]$ (where N is a small positive integer, typically $N = 10$). The user should now do the following:

(1) Compute the sum modulo N of the integers contained in the k pass-squares; let S ($0 \leq S < N$) be this sum. This is supposed to be done by mental calculation; if $N = 10$, the calculation is easy since all the numbers are just decimal digits, and in the sum all digits except the least significant digit are discarded.

(2) Drag the number grid (with the mouse or with a stylus) so that some grid-square labeled by the number S contains the first pass-place. More precisely, the user clicks-and-holds on an intersection of grid-lines and then drags; the whole grid and the numbers (but not the underlying image) are pulled along; the screen behaves like a torus during this transformation (i.e., it is left-right wrapped around and top-bottom wrapped around).

In order to prevent accidental login, the system gives the user h (e.g., $h = 10$) independent challenges of the above type. If the user passes all of them, login is successful.

Security: An observer does not see which grid-squares are pass-squares or, equivalently, which places on the image are pass-places. The sum $S \pmod{N}$, computed mentally by the user also remains unknown to the observer. The dragging of the grid and the displacement vector (let us denote it by (d_x, d_y)) will be seen; here, d_x is an integer mod a (where a is the number of grid squares in the horizontal direction of the screen), and d_y is an integer mod b (where b is the number of grid squares in the vertical direction of the screen). The observer concludes that the first pass-place (whose location is still unknown) is (d_x, d_y) away from some grid-square that contained the number S . But S is unknown; moreover, there usually are many grid-squares containing the same number S , and in each challenge, the numbers are different. A brute-force attack consists of guessing any grid square for the first pass-place and recording all k -tuples of possible pass-places that contain numbers such that their sum $S' \pmod{N}$ is contained in the square at displacement (d_x, d_y) from the guessed first pass-place. Many k -tuples will produce the same sum S' , and many values of S' (in the interval $[0, N]$) are possible. On average a fraction $\frac{1}{N}$ ($= \frac{1}{10}$ when $N = 10$) of all possible k -tuples of places can be expected to be ruled out in one observation. For example, let $N = 10$, $k = 10$, assume the screen is 40 by 30 cm^2 , and assume that each grid square is 2 by 2 mm^2 ; so, the screen has $200 \times 150 = 30,000$ grid-squares. Then there are $30,000^{10} \approx 5.9 \times 10^{44}$ possible k -tuples of pass-places; even if only a tenth of the grid squares (i.e., 3000 of them) are suitable for memorization by the user, there are 5.9×10^{44} possible k -tuples of pass-places. Thus, the password space is far too large for a brute-force attack.

References

- [1] J.C. Birget, Dawei Hong, Nasir Memon, “Robust discretization, with an application to graphical passwords”, Cryptology ePrint Archive <http://eprint.iacr.org/paper2003/168>
- [2] G. Blonder, “Graphical Password”, US Patent 5559961 (1996).
- [3] M. Boroditsky, “Passlogix password schemes”. <http://www.passlogix.com>
- [4] R. Dhamija, A. Perrig, “Déjà Vu: User study using images for authentication”, 9th *Usenix Security Symposium* (2000).
- [5] D.C. Feldmeier, P.R. Karn, “UNIX Password security – ten years later”, *Advances in Cryptology – CRYPTO’89*, Lecture Notes in Computer Science 435, Springer-Verlag (1990) 44-63.
- [6] Brad Isaacson, “The password problem”, Honors Project, Rutgers University at Camden (June 2001).
- [7] I. Jermyn, A. Mayer, F. Monroe, M. Reiter, A. Rubin, “The design and analysis of graphical passwords”, 8th *Usenix Security Symposium* (1999).
- [8] D. Klein, “A survey of, and improvements to, password security”, *UNIX Security Workshop II*, Berkeley, Calif., Usenix Association (1990).
- [9] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press (1997).
- [10] R. Morris, K. Thompson, “Password security: a case history”, *Communications of the ACM* 22 (1979) 594-597.
- [11] “The science behind Passfaces”, Real User Corporation (Sept. 2001).
<http://www.realuser.com>
- [12] A. Perrig, D. Song, “Hash visualization: A new technique to improve real-world security”, *Proceedings of the 1999 Workshop on Cryptographic Techniques and E-Commerce (CryTEC99)*.
- [13] L. Sobrado, J.C. Birget, “Graphical passwords”, *The Rutgers Scholar*, vol. 4 (2002).
<http://RutgersScholar.rutgers.edu/volume04/contents.htm>

Leonardo Sobrado

`lsobrado@camden.rutgers.edu`

Jean-Camille Birget

Dept. of Computer Science

Rutgers University at Camden

Camden, NJ 08102, USA

`birget@camden.rutgers.edu`