Robust discretization, with an application to graphical passwords

Jean-Camille Birget, Dawei Hong, and Nasir Memon *

Abstract

When data or the processing on the data have some uncertainty, discretization of those data can lead to significantly different output. For example, in certain graphical password schemes, a slight uncertainty in the clicking places can produce a different password. We present a discretization method, called *robust discretization*, which gives stable outputs in the presence of uncertainties.

Robust discretization enables us to implement graphical password schemes that are much more flexible and versatile than previously know ones.

1 Introduction

Discretization (also called quantization) of data consists of approximating a continuum, or a very large discrete set, by a discrete set of limited size. The limited size allows digital storage and information processing.

To fix the terminology, let us describe a simple example of a discretization of a two-dimensional rectangular grey image. The image is given by a function $g : [0, a] \times [0, b] \to [0, 1]$, where $[0, a], [0, b]$ are intervals in the reals $\mathbb{R}$, or in the integers $\mathbb{Z}$. In this paper we are only interested in the discretization of the domain of the image, namely the rectangle $R_2 = [0, a] \times [0, b]$.

The simplest way to discretize the rectangle R_2 is to choose a positive number q (called the **quantum**) and an **offset** (φ, ψ) (where $|\varphi|, |\psi| < q$), and to superimpose a square grid on the rectangle. The grid has $\lfloor a/q \rfloor + 1$ vertical lines

$(V_m) \qquad x = qm + \varphi \ \ (\text{where} \ \ m = 0, \ldots, \lfloor a/q \rfloor),$

and $\lfloor b/q \rfloor + 1$ horizontal lines

$(H_n) \qquad y = qn + \psi \ \ (\text{where} \ \ n = 0, \ldots, \lfloor b/q \rfloor).$

This subdivides R_2 into little grid squares of side-length q; near the edges of R_2 the grid squares are truncated. The discretization can also be described by a **grid map**, which tells us which points of the rectangle R_2 are mapped to which grid vertices.

$$g : \quad (x, y) \in [0, a] \times [0, b] \quad \longmapsto \quad \left(\left\lfloor \frac{x - \varphi}{q} \right\rfloor, \left\lfloor \frac{y - \psi}{q} \right\rfloor \right).$$

*JCB supported by NSF grant DMS-9970471. JCB and DH supported by an ISATC pilot grant from Rutgers University. NM supported by NSF

The set of points of R_2 that are mapped to a given grid point (m, n) is

$$g^{-1}(m,n) = \{(x,y) \in R_2 : \quad qm + \varphi \le x < q(m+1) + \varphi, \quad qn + \psi \le y < q(n+1) + \psi\}.$$

The set $g^{-1}(m,n)$ is called a **grid square**; the grid map g maps this entire grid square to the **grid point** (m, n).

Usually, the goal in the design of a good discretization is to minimize the loss of precision, or "quantization error", i.e., the distance between data points (x, y) and their discretizations. In our simple method above, the quantization error is the maximum value of $\|(x,y) - g(x,y) \cdot q\|$; see e.g. [3].

However, our goal is quite different. We are concerned with the stability of the discretization, and not with the loss of precision. Indeed, in our applications one of the main problems with discretization is **the edge problem**: If important features of the image are near a grid line V_m or H_n then slight changes or uncertainties in the image or in the processing can lead to significant changes in the discretization. For example, a person might repeatedly "point" to the same feature in an image (with a mouse or a stylus), but usually a person will not be able to point repeatedly to *exactly* the same place. If the place pointed to is near a grid edge, the discretization will lead to unintended changes in the output. In the graphical password schemes described later, this would often prevent the legitimate user from logging in.

In general we cannot expect the images to be such that all important features fit safely into grid squares, at a safe distance from the edges. What we need is a *robust discretization* in which there is no edge problem.

Previous work on robust discretization: Ideas similar to our robust discretization appear in Wu's work [15]. However, the exact definitions are not given there, nor are the properties of robust discretization (namely, our definitions, lemmas, and Theorem 2.4) precisely stated or proved.

Multi-grid methods have been used in Numerical Analysis for a long time, mainly in the area of partial differential equations. These methods, and their goals, are not closely related to the present paper.

Our robust discretization method is described in Section 2. The main motivation and the main originality of our paper is the application of robust discretization to graphical passwords, as described in Section 3.

2 Multigrid discretization

In order to make sure that all features of an image are at a safe distance from grid edges we use several grids at the same time. It is fairly intuitive that in 2-dimensional images, 3 grids are necessary and sufficient; then we can lay out the grids so that every point in the image is at a safe distance from the edges in at least one of the 3 grids. In a d-dimensional "image", $d+1$ grids are necessary and sufficient. We will first describe the multigrid discretization in the 2-dimensional case.

The "safe distance from the edges" is a parameter $r > 0$. The closed r-**disk** around a point (x_0, y_0) is $D_r(x_0, y_0) = \{(x,y) : \|(x_0, y_0) - (x,y)\| \le r\}$, where $\|.\|$ denotes Euclidean norm. The following definition makes the phrase "a point is at a safe distance from the edges" precise.

Definition 2.1 *A point (x, y) is r-**safe** in a grid G iff the closed r-disk around (x, y) is entirely contained in one grid square of G.*

The following characterization is immediate from the definition of a grid square. Just as for the integers, we define the **mod** operation for reals t and q (with $q \neq 0$) as follows:

$$t \ \mathbf{mod}\ q \ =_{\text{def}} \ t - \lfloor t/q \rfloor \cdot q$$

The next Lemma is obvious from the picture:

Lemma 2.2 *A point (x, y) is r-safe in a grid G with quantum q and offset (φ, ψ) iff*

$$\begin{cases} r \ \leq \ (x - \varphi) \ \mathbf{mod}\ q \ < \ q - r, & \textit{and} \\ r \ \leq \ (y - \psi) \ \mathbf{mod}\ q \ < \ q - r. \end{cases}$$

We introduce three grids, G_0, G_1, G_2. All three have quantum $q = 6r$, and they are "staggered": G_k has offset $(-2rk, -2rk)$, for $k = 0, 1, 2$. Then it turns out (as a consequence of the Theorem below), that every point is r-safe in at least one of the three grids.

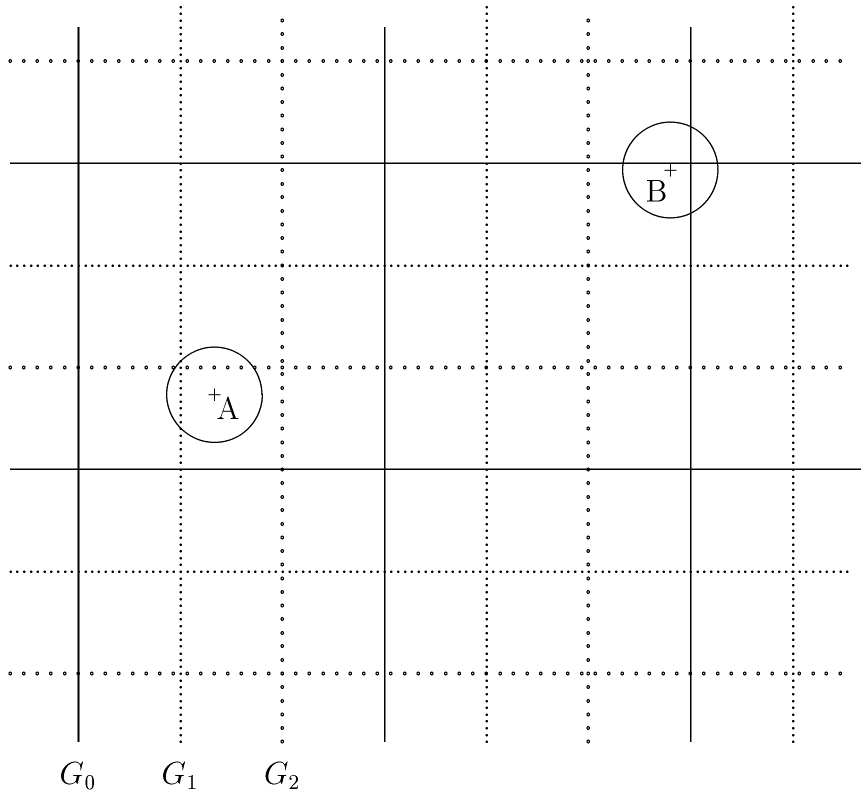

Fig. 1: The three grids G_0, G_1, and G_2 (A is safe in G_0, B is safe in G_1 and in G_2)

More generally, in a d-dimensional space we consider a rectangle $R_d = [0, a_1] \times \ldots \times [0, a_d]$, and we generalize all the definitions above in an obvious way. We introduce $d + 1$ grids G_k (with $k = 0, 1, \ldots, d$), all with quantum $q = 2r(d + 1)$, that are staggered: G_k has offset $(-2rk, \ldots, -2rk)$. By Lemma 2.2 above we have immediately:

Lemma 2.3 *A point $(x_1, \ldots, x_d)$ is r-safe in grid G_k $(k = 0, 1, \ldots, d)$ iff for all $i = 1, \ldots, d$,*

$$r \;\leq\; (x_i + 2rk) \textbf{ mod } (2r(d+1)) \;<\; r(2d+1).$$

The following theorem shows that robust discretization is possible. When the dimension d is 1 or 2, the proof is obvious from the picture of the grids.

Theorem 2.4 *For every point $(x_1, \ldots, x_d)$ of the d-dimensional rectangle R_d there is at least one grid G_k $(k = 0, 1, \ldots, d)$ such that $(x_1, \ldots, x_d)$ is r-safe in that grid.*

Proof. Consider any point $(x_1, \ldots, x_d) \in R_d$. Since r-safety only depends on the value of the coordinates **mod** $(2r(d+1))$ we can assume $0 \leq x_i < 2r(d+1)$ for each $i \in \{1, \ldots, d\}$. Also, by renaming the coordinates, if necessary, we can assume that $0 \leq x_1 \leq x_2 \leq \ldots \leq x_d < 2r(d+1)$.
Claim. For some $i_o \in \{1, \ldots, d\}$ and some $h_o \in \{0, 1, \ldots, d\}$:

$[2rh_o + r, 2r(h_o+1) + r[\quad \subset \quad]x_{i_o}, x_{i_o+1}[$, with $i_o < d$, $h_o < d$, or

$[2rd + r, 2r(d+1)[\;\cup\; [0, r[\quad \subset \quad]x_d, 2r(d+1)[\;\cup\; [0, x_1[$ (here, $i_o = h_o = d$).

Proof of the Claim. There are d numbers x_j $(j = 1, \ldots, d)$, and $d+1$ disjoint sets

$S_h = \; [2rh + r, 2r(h+1) + r[\quad (h = 0, 1, \ldots, d-1)$ and

$S_d = \; [2rd + r, 2r(d+1)[\;\cup\; [0, r[$.

Hence, at least one of these sets, say S_{h_o}, does not contain any x_j. We take x_{i_o} to be the largest x_j that is less than all the elements of the set S_{h_o}. This proves the Claim.

Proof of the Theorem.
Case A: $[2rh_o + r, 2r(h_o+1) + r[\quad \subset \quad]x_{i_o}, x_{i_o+1}[$, with $h_o < d, i_o < d$.

We claim that in this case, the point $(x_1, \ldots, x_d)$ is r-safe in grid G_k with $k = d - h_o$. Indeed, for all x_j with $j \leq i_o$ we have

$0 \leq x_j \leq x_{i_o} < 2rh_o + r;$

hence by adding $2r(d - h_o)$ $(= 2rk)$,

$2r(d - h_o) \leq x_j + 2rk < 2rd + r;$

hence, since $r \leq 2r(d - h_o)$ when $h_o \leq d - 1$,

$r \leq x_j + 2rk < 2rd + r.$

So, for all x_j with $j \leq i_o$, the condition of Lemma 2.3 is satisfied.
For all x_j with $j \geq i_o + 1$ we have

$2r(h_o + 1) + r < x_{i_o} \leq x_j < 2r(d+1);$

hence by adding $2r(d - h_o)$ $(= 2rk)$ and then subtracting $2r(d+1)$,

$r \leq (x_j + 2rk) \textbf{ mod } (2r(d+1)) < 2r(d - h_o)$ $(< 2rd + r)$,

hence the condition of Lemma 2.3 is satisfied for all x_j with $j \geq i_o + 1$.

Case B: $[2rd + r, 2r(d+1)[\;\cup\; [0, r[\quad \subset \quad]x_d, 2r(d+1)[\;\cup\; [0, x_1[$.

We claim that in this case, the point $(x_1, \ldots, x_d)$ is r-safe in grid G_0. Indeed, in this case, $x_d < 2r(d+1)$ and $r < x_1$, hence $r \leq x_1 \leq \ldots \leq x_j \leq \ldots \leq x_d < 2r(d+1)$. So the condition of Lemma 2.3 is satisfied (with $k = 0$) for all x_j. $\square$

It is easy to see that $d + 1$ *is the minimum number of grids that gives us r-safety.* Indeed, for d grids G_k ($k = 1, \ldots, d$), let $x_i = c_{i,k}$ be the grid hyperplane of G_k perpendicularly to coordinate axis x_i. Then the point $(x_i = c_{i,i})_{i=1,\ldots,d}$ belongs to a grid hyperplane for each grid. Hence, this point cannot be r-safe, no matter how small a positive number r is. In this example we did not assume anything about the discretization quantum or the offset; so d grids won't be sufficient, no matter what the quantum and the offset might be.

It is also easy to see that the quantum $2r(d+1)$ is the minimum required in order to make the theorem true with $d + 1$ grids.

Robust discretization

Since now we know that every point $(x_1, \ldots, x_d)$ in R_d is r-safe in at least one of the $d + 1$ grids G_k ($k = 0, 1, \ldots, d$), we simply map the point into one of the grids in which it is r-safe. We have to make a choice here, since usually there will be more than one grid in which $(x_1, \ldots, x_d)$ is r-safe. Let $\gamma : R_d \longmapsto \{0, 1, \ldots, d\}$ be any map such that $(x_1, \ldots, x_d)$ is r-safe in grid $G_{\gamma(x_1,\ldots,x_d)}$, for all $(x_1, \ldots, x_d) \in R_d$.

E.g., $\gamma(x_1, \ldots, x_d)$ could be defined to be the smallest k such that $(x_1, \ldots, x_d)$ is r-safe in grid G_k; or $\gamma(x_1, \ldots, x_d)$ could be a k that the distance of $(x_1, \ldots, x_d)$ to the grid hyperplanes of G_k is maximized; or $\gamma(x_1, \ldots, x_d)$ could be picked randomly, always subject to the r-safety condition (the latter will be described in the section on graphical passwords).

The **robust grid map** is then defined by

$$g : (x_1, \ldots, x_d) \in R_d \quad \longmapsto \quad (\gamma(x_1, \ldots, x_d), \quad g_{\gamma(x_1,\ldots,x_d)}(x_1, \ldots, x_d)).$$

So the grid map yields a grid identifier $k = \gamma(x_1, \ldots, x_d) \in \{0, 1, \ldots, d\}$ and a grid-point $g_{\gamma(x_1,\ldots,x_d)}(x_1, \ldots, x_d)$ in the corresponding grid G_k.

Note that we are not considering the intersection of grids; for each point we pick one of the original grids. (Using intersections of lattices is a different research idea, see e.g. [13], [1].)

3 A graphical password scheme

Graphical passwords were first proposed by G. Blonder [2]; in that scheme, a password uses an image in which many small regions have been preselected. The user has to choose some of these regions as a password, and in order to login later, the user must click on each one of the chosen regions (with a mouse or a stylus). Several implementations of this idea were given by [11]. Another version of click regions, this time with movement, appears in [6]. Somewhat different graphical password schemes (based on drawings, similar to a manuscript signature) were introduced and analyzed in [7]. Yet other graphical password schemes exist, based on image recognition [10], [12], [4].

The click region passwords of Blonder have a limitation, namely the fact that the click regions are predefined; they are part of the design of the image. This implies that the users cannot provide images of their own for making passwords, since a user's images will not have any predefined click regions. It also implies that users cannot choose click places that are not among the preselected ones. It would be desirable to let users introduce their own images, which they are familiar and in which they recognize and remember many details. Moreover, users would like to choose any places that attract them as click regions; such places are easier to remember.

On the other hand, allowing arbitrary click regions leads to the edge problem of discretization, as we mentioned at the end of Section 1: Users are unable to click repeatedly at the *exact* place that they chose when they made up their password; therefore a discretization has to be used. But then it will often happen that a click region overlaps different grid-squares, which means that the password clicked by the user is "a little" different from the password that was originally chosen.

Allowing approximately correct passwords, however, prevents us from using *secure password hashing* (a. k. a. "password encryption") since passwords that are approximately (but not exactly) the same will usually have very different hashes. Secure password hashing is important because it enables secure storage of passwords in an insecure storage (and back-up) environment.

Thus, *robust discretization* is the method of choice for implementing graphical password schemes that let the user bring in their own images and let the user choose arbitrary places as click regions, and that permits us to use password hashing.

Our graphical password scheme consists of three components: image handling, password selection, login.

1. The **image handling** component enables users to introduce their own images; the images are stored, together with a collection of images provided by the system. For this password system to work well, it is important that the images be fairly intricate, with lots of interesting details that could be chosen as click regions (e.g., maps, architectural images, cityscapes, certain landscapes, renaissance paintings).

2. The **password selection** component allows the user to select a new password. Assuming the user has already logged in (by using either a graphical or a conventional password), the user types the password command (or clicks the 'new password' button). The system then prompts the user for a user name and current password. If the system accepts the current password, it asks the user to specify a new image to the image handling component, or to keep the current image. The safety parameter r (for robust discretization) can be set by the user; a default of $r = 2.5$ mm would be fine.

Next, the image is displayed, and the user has to click on a few places (of the user's choice); for security's sake, at least 5 places should be clicked. Let c be the number of clicks, and let $(x_1, y_1), \ldots, (x_c, y_c)$ be the sequence of click places. The system takes the pixel coordinates of the click places, and for each click place (x_i, y_i) it computes a grid identifier $k_i \in \{0, 1, 2\}$ such that (x_i, y_i) is r-safe in grid G_{k_i} (for $i = 1, \ldots, c$). If a click place (x_i, y_i) is r-safe in more than one of the 3 grids, the system chooses one of the safe ones at random (e.g., take k_i to be $x_i + y_i \bmod 2$ if there are two choices, or $x_i + y_i \bmod 3$ if there are three choices). Random choice is better than 'best-fit' because it hides as much information as possible from attackers. This defines the function γ of the robust discretization.

For each click place (x_i, y_i), the system remembers the grid identifier $k_i = \gamma(x_i, y_i)$; in our scheme, k_i does not need to be secret, so k_i is stored in the clear (not hashed).

The system also computes the grid point $g_{k_i}(x_i, y_i)$ of the click place (x_i, y_i) with respect to the grid G_{k_i}, and it remembers the hashed value of the sequence of grid points of the click places.

In summary, the user provides a sequence of click places $((x_1, y_1), \ldots, (x_c, y_c))$, terminated by a 'return'. The system remembers

$$(\gamma(x_1, y_1), \ldots, \gamma(x_c, y_c)) \quad \text{and} \quad \text{HASH}(g_{\gamma(x_1,y_1)}(x_1, y_1), \ldots,\ g_{\gamma(x_c,y_c)}(x_c, y_c))$$

in the user's password record. The secret consists of the sequence of grid points.

As usual for password systems, before putting the new password in operation, the system should ask the user to confirm the password (by repeating the choice of clicks, but this time it tolerates errors within the safety parameter r).

Note that this system is based on the *sequence* (not the set) of click points; i.e., the order of the clicks matters. (See below for the unordered version, or 'set version', of the system.)

3. The **login** component presents the user with a window into which the user types the user name. The system then retrieves the user's password record (which contains the sequence of grids to be used), and displays the user's password image. (If the user is not valid, the system will display an arbitrarily chosen image, and will eventually reject the user.)

The user then makes a sequence of clicks on the image. For the ith click the system uses the ith grid ($1 \leq i \leq c$) in the stored sequence of grid identifiers, and computes the grid point of that click place. (The system will *not* check whether the clicked point is r-safe in this grid, because we want to tolerate errors up to r.) When the user types the 'return' the system computes the hash value of the sequence of grid points and compares this with the hash value stored in the user's password record. If the two are identical the user is accepted, otherwise the user is rejected.

Improved implementation

In the graphical password scheme described above, the c clicks were implemented as c 2-dimensional points. Instead, we could represent the click points $(x_1, y_1), \ldots, (x_c, y_c)$ as one $2c$-dimensional point $(x_1, y_1, \ldots, x_c, y_c)$.

This does not change anything from the user's point of view, but it improves the security of the scheme. Indeed, when the c clicks are viewed as c 2-dimensional points, a sequence of c discretization grids is stored in the system in the clear (and possibly revealed to a potential attacker). At each click, there are 3 grids that are a priori possible, so for c clicks, 3^c grid sequences are priori possible. One out of 3^c possible grid sequences is revealed. On the other hand, for a $2c$-dimensional point, there are $2c+1$ grids. One grid out of $2c+1$ possible grids is revealed. Of course, a source of information in which each event has probability $\frac{1}{2c+1}$ has much less entropy than a source in which each event has probability $\frac{1}{3^c}$ (when $c > 1$). E.g., for $c = 5$, the grid information is $\log_2(2c+1) = \log_2 11 \approx 3.459$ for the improved method; i.e., approximately 3.5 bits are lost. For the first implementation, the grid information is $\log_2 3^c = c \log_2 3 \approx 5 \times 1.58 \approx 7.9$; so approximately 8 bits are lost. If there are $c = 10$ clicks, the improved method loses approximately 4.4 bits, whereas the first method loses approximately 15.8 bits.

In summary, in the improved implementation, much less information is revealed by the discretization grids.

Note however that the revealed bits "do not accumulate", i.e., if an attacker strikes several times, the same grid information ($\log_2(2c+1)$ bits, respectively $\log_2 3^c$ bits) is learned again and again by the attacker.

The zoom

Zooming-in magnifies the area of the screen close to the cursor. The magnification allows the user to choose finer features of the image as click places. This may be a convenience for the user; it also increases the password space significantly.

One can imagine several options: (1) The user could click with the second mouse button to activate or deactivate the zoom-in. (2) There is automatic zoom-in in the vicinity of the cursor. (3) The automatic zoom-in around the cursor depends on the speed of movement of the cursor; as the cursor slows down (near a possible click target), the magnification increases.

Set version of the system (instead of a sequence):

One can design the system (both in its first implementation as well as in the improved implementation) so that the order of the click places does not matter. For this, the system *sorts* the grid point sequence $(g_{\gamma(x_1,y_1)}(x_1, y_1), \ldots, \; g_{\gamma(x_c,y_c)}(x_c, y_c))$ before applying the hash function. The grid sequence $(\gamma(x_1, y_1), \ldots, \gamma(x_c, y_c))$ will also be reordered accordingly. The order-relation used for this sorting can be any total order (e.g., column dominant order, or row dominant order, or lexicographic order of the grid points with coordinates treated as strings of digits).

The set version has the drawback that it offers a smaller password space. However, it is possible that in some situations, the set of click places is more memorable than the sequence; this would need some human testing.

4 Security of graphical passwords

The security of a password scheme depends, roughly, on the size of the password space and also on the way humans tend to use the password scheme.

Let us look at the **size of the password space** of our graphical passwords. For an image of size 330×250 mm^2 (for example), with safety parameter $r = 1$ mm and quantum $q = 6r = 6$ mm, each grid has at least 2214 ($= 54 \times 41 \approx 330/6 \times 250/6$) grid points. (We round 330/6 and 250/6 down to the integer below). For graphical passwords with 5 clicks the number of possible passwords is therefore $> 2214^5 \approx 5 \times 10^{16}$. With 6 clicks the number of possible passwords is $2214^6 \approx 10^{20}$, and with 7 clicks it is $2214^8 \approx 2.61 \times 10^{23}$. The latter number is comparable to the Avogadro number $\mathrm{N}_A \approx 6.02 \times 10^{23}$ (mol g)$^{-1}$, which is, e.g., the number of hydrogen molecules in 2 grams.

The value $r = 1$ mm will be inconveniently small for most users, and for most computer screens (due to poor resolution). However, when the zoom feature is used, $r = 1$ mm should not be a problem.

If $r = 2$ mm, there will be about 540 grid points; with 6 clicks the number of possible passwords is then $540^6 \approx 2.48 \times 10^{16}$. For 9 clicks the number of possible passwords is then $540^9 \approx 3.9 \times 10^{24}$ (which is larger than the Avogadro number). Note that $\log_2 3.9 \times 10^{24} \approx 81.7$, so such passwords correspond to bit-strings of length 81; thus, these graphical passwords have a password space close to the key spaces used for cryptography.

Human aspects of the security of graphical passwords

The security features and advantages of graphical passwords, in general, and especially from a human point of view, are quite well analyzed in [7]; the graphical passwords considered there were different than ours, but the same analysis of human capabilities applies here. The conclusion of this analysis is that for a given password space size, graphical passwords are much more memorable than alpha-numeric passwords; human memory is greater for images than for text.

For our graphical password system to be effective, we have to use images that are intricate enough to offer many attractive click places to a user (e.g., maps, architectural graphics, renaissance paintings, city scapes, complicated landscapes). On the other hand, featureless images, repetitive patterns, or abstract paintings are less likely to be useful in this context.

However, the claimed advantages of graphical passwords have not been sufficiently studied by human-factors experiments; the whole field of passwords, and most cryptographic tools for that matter, have received very little attention from experimental human-factors studies. Nor is there any body of practical experience for graphical passwords. The arguments in favor of graphical passwords are plausible, but indirect.

Let us look in more detail at attacks on and weaknesses of passwords.

• *Dictionary attack, and weak passwords:* When using alpha-numerical passwords, very few humans will pick random character strings, since such strings are very hard to remember. Instead, human users almost always choose their passwords to be words or names that appear in dictionaries or directories. Quite often, some add-ons are applied to the passwords (e.g., capitalization, insertion of a few digits, and perhaps some permutations). In most systems there are some users with passwords that are explicitly in a dictionary or a directory; therefore, most multi-user computer systems are vulnerable to dictionary attacks. Even with these add-ons, the password space of the actually used alpha-numerical passwords is much smaller than the a priori space (e.g., 10^9 instead of $62^8 \approx 2 \times 10^{14}$). Hence, the actually used alpha-numerical passwords are weak (see some classical studies [9], [5], [8]).

For graphical passwords the situation is much better. First of all, there are no "dictionaries" of click places in existence. Moreover, sequences of click places don't have meanings in the same way character sequences do, so we can expect a much greater variety of graphical passwords that people will actually use. Hence, making "dictionaries" for click places in graphical passwords seems unfeasible.

It would be hard for a computerized attack to guess what regions in an image will attract a user; all one can say is that a user is likely to chose a part of an image that has features, edges, etc. But an intricate image (see our discussion above) will have hundreds of such features. A human attacker could often narrow down the number of candidate features of an image; but this will probably still leave hundreds of choices for each click. The same can be said about ad hominem attacks (i.e., attacks directed against a particular user).

• *Grid attack:* Can a computerized or a human attacker make use of the grid sequence to learn something about the graphical password? The grid sequence of numbers 0, 1, 2, denoted $(\gamma(x_1, y_1), \ldots, \gamma(x_c, y_c))$ in the previous Section, is not kept secret. In a $6r$-by-$6r$ grid square, only a $4r$-by-$4r$ subsquare is r-safe (i.e., at distance $\geq r$ from the edges of the grid). So each grid G_k ($k = 0, 1, 2$) has a proportion $\frac{16r^2}{36r^2} = \frac{4}{9} \approx 0.444$ of the area that is r-safe. Hence, if the attacker knows that a certain grid (e.g., G_0) is used in a certain click, the unsafe regions of this grid are ruled out. So, the attacker only has to consider 44.4% of the image in order to search for the click point. However, the only thing that matters for the graphical password is which grid square the click is in, not where in the grid square the click is. In other words, although only 44.4% of the screen need to be considered by the attacker, these 44.4% cover all the grid points of the grid (100%). So, the attacker learns nothing about which grid square was clicked; hence, as far as the graphical password is concerned, the attacker learns nothing about the password by knowing the grids. In particular, the knowledge of the grid sequence does not reduce the size of the password space.

A human attacker could find the grid information useful, since it restricts the number of "humanly interesting" features that could have been chosen for this click. But again, these 44.4% of the screen cover all the grid squares, and in a suitably intricate image, most grid square can be expected to have interesting features within their $4r$-by-$4r$ subsquare.

Other security issues

Graphical passwords cannot easily be written down. This is a security advantage, but it can also be considered an inconvenience. Actually, a user could make notes about the password. In the best case, those notes may be clear enough for the user but rather unclear for an intruder. Human-factors experiments, and long-term experience with graphical passwords would be needed to clarify this issue.

A general drawback of all graphical password schemes is *shoulder surfing*, which happens when an attacker watches of films a person during login. Alpha-numerical passwords are also vulnerable to this, but a little less. Also, passwords could be "sniffed", i.e., captured during login communications; sniffing1 can be viewed as a form of shoulder surfing. Password sniffing can be made impossible by encryption. Direct shoulder surfing is a difficult problem; see [14] for a graphical password scheme that is resistant to shoulder surfing, at the expense of a more elaborate login procedure.

Acknowledgements: This paper benefited from the work with the first author's students Brad Isaacson and Leonardo Sobrado.

References

[1] B. Beferull-Lozano, A. Ortega, "Efficient quantization for overcomplete expansions in $\mathbb{R}^N$", *IEEE Transactions on Information Theory* 49(1) (Jan. 2003) 129-150.

[2] G. Blonder, "Graphical Password", US Patent 5559961 (1996).

[3] J. H. Conway, N.J.A. Sloane, *Sphere packings, lattices and groups*, Springer-Verlag (1998).

[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. Monrose, 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] R. Morris, K. Thompson, "Password security: a case history", *Communications of the ACM* 22 (1979) 594-597.

[10] "The science behind Passfaces", Real User Corporation (Sept. 2001). http://www.realuser.com

[11] M. Boroditsky, "Passlogix password schemes". http://www.passlogix.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] N.J.A. Sloane, B. Beferull-Lozano, "Quantizing Using Lattice Intersections", *Discrete Comput. Geometry* 25 (2003) 868-881.

[14] L. Sobrado, J.C. Birget, "Graphical passwords", *The Rutgers Scholar*, vol. 4 (2002). http://RutgersScholar.rutgers.edu/volume04/contents.htm

[15] Chai-Wah Wu, "On the design of content-based multimedia authentication systems", to appear in *IEEE Transactions on Multimedia.* (Conference version: "Limitations and requirements of content-based multimedia authentication systems", *IS&T/SPIE's International Symposium on Electronic Imaging: Science and Technology*, 2001.) http://www.research.ibm.com/people/c/chaiwah/

Jean-Camille Birget and **Dawei Hong**
Dept. of Computer Science
Rutgers University at Camden
Camden, NJ 08102, USA
{birget, dhong}@camden.rutgers.edu

Nasir Memon
Dept. of Computer and Information Science
Polytechnic University
Brooklyn, NY 11201, USA
memon@poly.edu