

# Robust discretization, with an application to graphical passwords

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

18 March 2005

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

Keywords: Robust discretization, images, passwords.

## 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.

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  $g : [0, a] \times [0, b] \rightarrow [0, 1]$ , 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  $R_2$  is to choose a positive number  $q$  (called the **quantum**) and an **offset**  $(\varphi, \psi)$  (where  $|\varphi|, |\psi| < q$ ), and to superimpose a square grid on the rectangle. The grid has  $\lfloor a/q \rfloor + 1$  vertical lines

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

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

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

This subdivides  $R_2$  into grid squares of side-length  $q$ ; near the borders 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.

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

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

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

The set  $g^{-1}(m, n)$  is called a **grid square**; the grid map  $g$  maps this entire grid square, namely  $[qm, q(m + 1)) \times [qn, q(n + 1))$ , to the **grid point**  $(m, n)$ .

---

\*JCB supported by NSF grant DMS-9970471. JCB and DH supported by an ISATC pilot grant from Rutgers University, and by NSF grant CCR-0310793. NM supported by NSF grants. An earlier version of this paper appears in the *Cryptology ePrint Archive*, report 2003/168 (Aug. 2003). Preliminary work also appeared in [11].

**Notation for intervals:** In order to avoid confusion with the *point*  $(a, b)$ , the open interval  $\{x : a < x < b\}$  will be denoted by  $\langle a, b \rangle$ . The closed interval is denoted by  $[a, b]$ , and the half-open intervals are denoted by  $[a, b)$  and  $\langle a, b]$ .

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. 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 a 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. More precisely, in this paper a discretization is called *robust* if and only if we have the following: (1) when a location pointed to is close (within a distance  $< r_1$ ) to an originally chosen location, the output is the same as when one clicks on the originally chosen location; (2) if a location pointed to is at distance greater than  $r_2$  from the originally chosen location (for some specified distance  $r_2$  with  $r_2 > r_1$ ), the output is guaranteed to be different than for the originally chosen location. (Here, “pointing to” a location means, e.g., pointing with a stylus, or clicking with a mouse.)

*Previous work on robust discretization:* Ideas similar to our robust discretization appear in Wu’s work [14]. However, the edge problem is not stated there, nor are the properties of robust discretization (namely, the definition or  $r$ -safe and Theorem 2.4) precisely stated or proved.

*Overview:* 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. Security is discussed in Section 4.

## 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”, we will show that  $d + 1$  grids are necessary and sufficient.

The “safe distance from the edges” is a parameter  $r > 0$ . The open  $r$ -**disk** around a point  $(x_1, \dots, x_d)$  is  $D_r(x_1, \dots, x_d) = \{(z_1, \dots, z_d) : \|(x_1, \dots, x_d) - (z_1, \dots, z_d)\| < r\}$ , where  $\|\cdot\|$  denotes euclidean norm in  $d$ -dimensional space. The following definition makes the phrase “a point is at a safe distance from the edges” precise.

**Definition 2.1** *A point  $(x_1, \dots, x_d)$  is  $r$ -safe in a  $d$ -dimensional square grid  $G$  iff the open  $d$ -dimensional  $r$ -disk around  $(x_1, \dots, x_d)$  is entirely contained in one grid hypercube of  $G$ .*

Just as for the integers, we define the **mod** operation for reals  $t$  and  $q$  (with  $q \neq 0$ ) as follows:

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

We have the following easy characterization:

**Lemma 2.2** *A point  $(x_1, \dots, x_d)$  is  $r$ -safe in a  $d$ -dimensional grid  $G$  with quantum  $q$  and offset  $(\psi_1, \dots, \psi_d)$  iff for all  $i = 1, \dots, d$ :*

$$r < (x_i - \psi_i) \bmod q < q - r.$$

**Proof.** Suppose that the grid spacing (i.e., the discretization quantum) is  $q$ , and the offset of the grid hyperplanes in dimension  $i$  is  $\psi_i$  (for  $i = 1, \dots, d$ ). Let  $H_{m_1}^{(1)}, \dots, H_{m_d}^{(d)}$  be the grid hyperplanes that border the grid hypercube containing the point  $(x_1, \dots, x_d)$ . The point is  $r$ -safe iff it is at distance  $> r$  from each one of these hyperplanes, i.e.,  $(x_1, \dots, x_d)$  is  $r$ -safe iff for some integers  $m_1, \dots, m_d$  we have (for  $i = 1, \dots, d$ ):  $\psi_i + qm_i < x_i < \psi_i + q(m_i + 1)$ . By the definition of “mod”, this is equivalent to the statement of the Lemma.  $\square$

In two dimensions we now 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  $k = 0, 1, 2$ . We will see (as a consequence of the Theorem below), that every point is  $r$ -safe in at least one of these three grids.

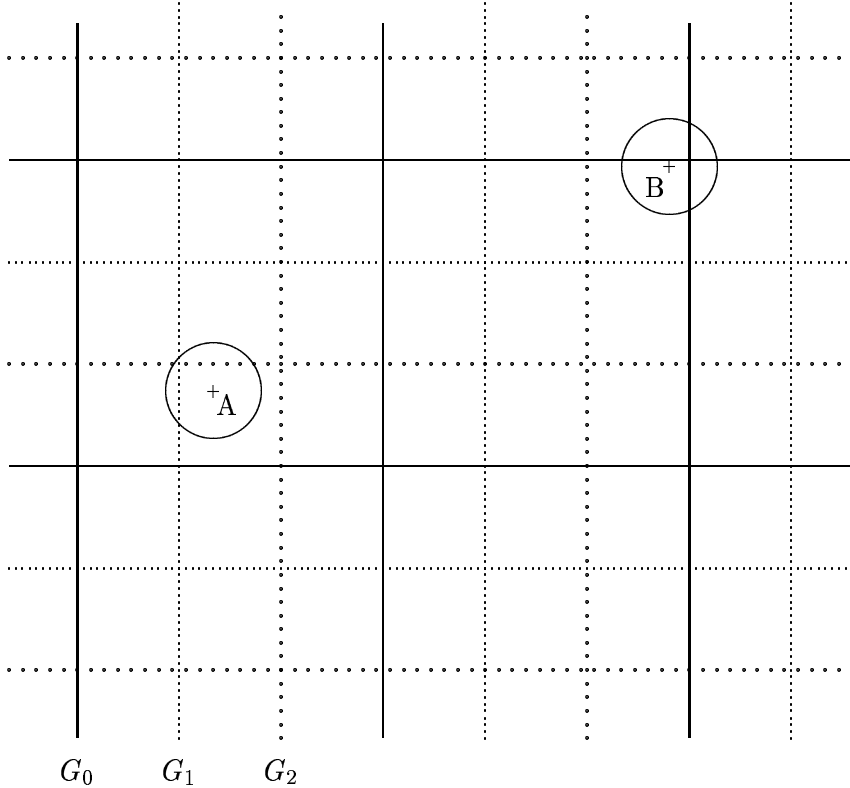


Fig. 1: The three grids  $G_0, G_1$ , and  $G_2$  (A is safe in  $G_0$ , B is safe in  $G_1$  and in  $G_2$ )

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

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

$$r < (x_i + 2rk) \bmod (2r(d+1)) < r(2d+1).$$

**Proof.** In Lemma 2.2, just plug in  $6r$  for the quantum  $q$ , and plug in  $2rk$  for each offset  $\psi_i$ .  $\square$

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

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

**Proof.** Consider any point  $(x_1, \dots, x_d)$ . 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  $i = 1, \dots, d$ . Also, by renaming the coordinates if necessary, we can assume that  $0 \leq x_1 \leq x_2 \leq \dots \leq x_d < 2r(d+1)$ .

**Claim.** *For some  $i_o \in \{1, \dots, d\}$  and some  $h_o \in \{0, 1, \dots, d\}$ :*

$$\begin{aligned} [2rh_o + r, 2r(h_o + 1) + r) &\subset \langle x_{i_o}, x_{i_o+1} \rangle \quad \text{with } i_o < d, h_o < d, \quad \text{or} \\ [2rd + r, 2r(d+1)) \cup [0, r) &\subset \langle x_d, 2r(d+1) \rangle \cup [0, x_1) \quad (\text{when } i_o = h_o = d). \end{aligned}$$

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

$$\begin{aligned} S_h &= [2rh + r, 2r(h+1) + r) \quad (\text{for } h = 0, 1, \dots, d-1) \quad \text{and} \\ S_d &= [2rd + r, 2r(d+1)) \cup [0, r). \end{aligned}$$

Hence by the pigeonhole principle, 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.* By the Claim we only need to consider the two cases *A* and *B* below.

*Case A:*  $[2rh_o + r, 2r(h_o + 1) + r) \subset \langle x_{i_o}, x_{i_o+1} \rangle$  with  $h_o < d, i_o < d$ .

We claim that in this case, the point  $(x_1, \dots, x_d)$  is  $r$ -safe in grid  $G_k$  with  $k = d - h_o (> 0)$ . 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  $r < 2r(d - h_o)$  when  $h_o \leq d - 1$ ,

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

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  $2r(d+1)$ ,

$$r < (x_j + 2rk) \bmod (2r(d+1)) < 2r(d - h_o) (< 2rd + r),$$

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

*Case B:*  $[2rd + r, 2r(d+1)) \cup [0, r) \subset \langle x_d, 2r(d+1) \rangle \cup [0, x_1)$ .

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

*The number  $d+1$  is the minimum number of grids that gives us  $r$ -safety.* Indeed, for  $d$  grids  $G_k$  ( $k = 1, \dots, d$ ), let  $x_i = c_{i,k}$  be a grid hyperplane of  $G_k$  perpendicular to coordinate axis  $x_i$ . Then the point  $(x_i = c_{i,i})_{i=1, \dots, d}$  belongs to a grid hyperplane for each grid. Hence, this point cannot be  $r$ -safe, no matter how small a positive number  $r$  is. So  $d$  grids won't be sufficient, no matter what the quantum and the offsets may be.

## Robust discretization

Since now we know that every point  $(x_1, \dots, x_d)$  is  $r$ -safe in at least one of the  $d + 1$  grids  $G_k$  ( $k = 0, 1, \dots, d$ ), we simply map the point into one of the grids in which it is  $r$ -safe. We have to make a choice here, since often there will be more than one grid in which  $(x_1, \dots, x_d)$  is  $r$ -safe. A safe grid choice map  $\gamma : \mathbb{R}^d \mapsto \{0, 1, \dots, d\}$  is any map such that  $(x_1, \dots, x_d)$  is  $r$ -safe in grid  $G_{\gamma(x_1, \dots, x_d)}$ , for all  $(x_1, \dots, x_d)$ . E.g.,  $\gamma(x_1, \dots, x_d)$  could be defined to be the smallest  $k$  such that  $(x_1, \dots, x_d)$  is  $r$ -safe in grid  $G_k$ ; or  $\gamma(x_1, \dots, x_d)$  could be a  $k$  for which the distance of  $(x_1, \dots, x_d)$  to the grid hyperplanes of  $G_k$  is maximized; or  $\gamma(x_1, \dots, x_d)$  could be chosen randomly, always subject to the  $r$ -safety condition (but kept fixed once chosen). The **robust grid map** is then defined by

$$g : (x_1, \dots, x_d) \in \mathbb{R}^d \mapsto (\gamma(x_1, \dots, x_d), g_{\gamma(x_1, \dots, x_d)}(x_1, \dots, x_d)).$$

So the grid map provides a grid identifier  $k = \gamma(x_1, \dots, x_d) \in \{0, 1, \dots, d\}$  and a grid-point  $g_{\gamma(x_1, \dots, x_d)}(x_1, \dots, x_d) \in \mathbb{Z}^d$  of the corresponding grid  $G_k$ .

The definition of  $r$ -safe says that if a chosen point  $p$  is  $r$ -safe in the grid  $G_k$ , and *if a point  $x$  is at euclidean distance  $< r$  from  $p$ , then  $x$  is in the same grid square as  $p$* . In other words, once the system has chosen a grid  $G_k$  in which the click point  $p$  is  $r$ -safe (i.e.,  $k = \gamma(p)$ ), then any click within distance  $< r$  from  $p$  will be in the same grid square as  $p$ , and hence will be recognized by the system as the same as  $p$ . So the user has a guaranteed tolerance  $r$  for click errors.

On the other hand, *if the user clicks on a point  $x$  at a distance  $> r(2d + 1)\sqrt{d}$  from the chosen point  $p$ , then the click is guaranteed to be treated as incorrect*. Indeed, each grid square has side-length  $q = 2r(d + 1)$ , and a safe point is at distance  $> r$  from the edges; therefore, if  $x$  is in the same grid square as  $p$ , then  $|p_i - x_i| \leq 2r(d + 1) - r = r(2d + 1)$  for all the coordinates  $(x_1, \dots, x_d) = x$ ,  $(p_1, \dots, p_d) = p$ . In other words, max-distance between  $p$  and  $x$  is  $\leq r(2d + 1)$ ; hence the euclidean distance (in  $d$ -dimensional space) between  $p$  and  $x$  is  $\leq r(2d + 1)\sqrt{d}$ . In 2-dimensional space this means that a click at distance  $> r5\sqrt{2}$  ( $\approx 7.07r$ ) is guaranteed to be treated as incorrect.

If a point  $x$  is at distance between  $r$  and  $r(2d + 1)\sqrt{d}$  from a chosen point  $p$ , it can happen that  $x$  is in the same grid square as  $p$ , and it can happen that they are in different grid squares (depending on the exact locations of  $x$  and  $p$ ). In any case, for each grid the number of grid squares in an  $a_1 \times \dots \times a_d$  hyper-rectangle is at least  $\lfloor a_1/q \rfloor \cdot \dots \cdot \lfloor a_d/q \rfloor$ , and all of these grid squares will be treated as different by the system.

## 3 A graphical password scheme

Graphical passwords were first proposed by G. Blonder [1]; 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 log in later, the user must click in each one of the chosen regions (with a mouse or a stylus). Several implementations of this idea were given by [9]. Another version of click regions, this time with movement, appears in [4]. Somewhat different graphical password schemes (based on drawings) were introduced and analyzed in [5]. Yet other graphical password schemes exist, based on image recognition [8], [10], [2].

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, and that users cannot choose click places that are not among the preselected ones. Moreover, complex “natural” images (landscapes, cityscapes, art, photos of individual people or groups, etc.) are not easy to subdivide into fixed and recognizable click regions.

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 hash values. Secure password hashing is important because it enables secure storage of passwords in an insecure storage (and back-up) environment. In this paper, “hashing” always means secure, cryptographic hashing.

Thus, *robust discretization* gives us graphical password schemes with a great variety of images and click places, and with secure password hashing.

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

**1. The image handling** component enables users to choose images or to introduce their own; 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., topographic 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 enters the “password” command. The system then prompts the user for a user name and current password. If the system accepts the current password, it lets the user specify a new image, or keep the current image. The safety parameter  $r$  (for robust discretization) can be set by the user, and a default value is also available.

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  $(x_1, y_1), \dots, (x_c, y_c)$  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  $i = 1, \dots, c$ ). For each click place  $(x_i, y_i)$ , the system remembers the grid identifier  $k_i = \gamma(x_i, y_i)$ ; in our scheme,  $k_i$  is stored in the clear (not cryptographically 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 secure hash value of the sequence of grid points of the click places. In summary, the user provides a sequence of click places  $((x_1, y_1), \dots, (x_c, y_c))$ , terminated by a ‘return’. The system remembers

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

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  $r$ ).

**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 a default image, and will eventually reject the user.)

The user then makes a sequence of clicks on the image. For the  $i$ th click the system uses the  $i$ th 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 secure 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  $c$  two-dimensional points. Instead, we could represent the click points  $(x_1, y_1), \dots, (x_c, y_c)$  as one  $2c$ -dimensional point  $(x_1, y_1, \dots, x_c, y_c)$ . This does not change anything from the user's point of view, but it decreases the amount of information contained in the chosen grids. Indeed, when the  $c$  clicks are viewed as  $c$  two-dimensional points, a sequence of  $c$  grids is stored in the system (in the clear). At each click, there are 3 grids that are possible, so for  $c$  clicks,  $3^c$  grid sequences are possible. One out of  $3^c$  possible grid sequences is stored in the system files. On the other hand, for a  $2c$ -dimensional point, there are  $2c+1$  grids. One grid out of  $2c+1$  possible grids is stored. 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  $\log_2(2c+1) = \log_2 11 \approx 3.5$  for the improved method. For the first implementation, the grid information is  $\log_2 3^c \approx 7.9$ . If there are  $c = 10$  clicks, the improved method requires approximately 4.4 bits, whereas the first method needs approximately 15.8 bits.

### 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 by, in effect, decreasing the parameter  $r$ .

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.

## 4 Security of graphical passwords

### 4.1 Generalities

The security of a password scheme depends, roughly, on the size of the password space (i.e., the total number of possible passwords for any given setting of the password parameters), and also on the way humans tend to use the password scheme.

The **size of the password space** of our graphical passwords is parameterized by the safety parameter  $r$  (or, equivalently, the quantum  $q = 6r$ ) and the number of click points  $c$ . Moreover, the image size  $a \times b$  (or the screen size), and the resolution (number of pixels per square centimeter) are parameters, but we usually have little control over them. In general, the number of possible passwords is  $(\lfloor a/q \rfloor \cdot \lfloor b/q \rfloor)^c$ . For example, for an image of size  $330 \times 260$  mm<sup>2</sup> (for example), with safety parameter  $r = 1$  mm (so,  $q = 6r = 6$  mm), each grid has at least  $2365$  ( $\approx 330/6 \times 260/6$ ) grid points. For graphical passwords with  $c = 5$  clicks the number of possible passwords is therefore  $2365^5 \approx 7 \times 10^{16}$ . With 6 clicks the number of possible passwords is  $2365^6 \approx 1.7 \times 10^{20}$ , and with 7 clicks it is  $2365^7 \approx 4 \times 10^{23}$ . (Compare with the Avogadro number  $N_A \approx 6.02 \times 10^{23}$  (mol g)<sup>-1</sup>.)

For the value  $r = 1$  mm the grid squares have side-length  $q = 6$  mm; this could be inconveniently small for many users, and for present-day computer screens (due to poor resolution). However, when the zoom feature is used,  $r = 1$  mm will not be small.

When  $r = 2$  mm, there will be about 560 grid points; with 6 clicks the number of possible passwords is then  $560^6 \approx 3 \times 10^{16}$ . For 9 clicks the number of possible passwords is then  $560^9 \approx 5.4 \times 10^{24}$ .

The human use of a password scheme determines the **effective password space** which, intuitively, consists of the passwords that users are likely to use. Although it is difficult to define the effective password space rigorously in general, for our graphical password system we can find an operational definition as follows: The chosen image itself is an additional parameter now, and instead of the whole image size  $a \times b$  we now count the area of the image that is occupied by “memorable” features. We cannot define rigorously what a memorable feature is, but by human experiments we can find out which grid squares contain features that humans will accept as being possible to remember; we can also double check by determining the non-memorable regions – intuitively those are large “empty” areas that do not contain edges or contrasts. Small empty regions, surrounded by features, could however be used for clicking (e.g., the center of such a small region could be chosen). Thus, for our graphical password system to be effective, we have to use images that are intricate enough to offer many attractive click places (e.g., maps, architectural graphics, renaissance paintings, city scapes, complicated landscapes). Based on our experience from [12] and [13] we estimate that in a “good” image as much as *half* of the area will be occupied by possible click points; however, human factors experiments will be needed to explore this. According to this estimate, the size of the total password space is divided by  $2^c$  when we go to the effective password space. E.g., for  $c = 6$  click points, we divide by 64, which in the case of  $r = 1$  mm, leaves us with an effective password space of size  $2.6 \times 10^{18}$ .

Some human aspects of graphical passwords are quite well analyzed in [5] and in [8]. But the graphical passwords considered in those papers are very different from ours. Some of our own results on human aspects will appear in [12] and [13], and further human factors testing is in progress.

## 4.2 Attacks on graphical passwords

We consider two attacks (attack models) against graphical passwords, (1) a dictionary attack against the collection of all the users of a system, (2) an individual attack against a particular user. The discussion in this subsection is preliminary, and will develop as on-going human factors experiments progress.

- *Collective dictionary attack*

Dictionary attacks against the *collection of all the users* of a system (which we will call “collective attacks”) are well known for alphanumeric passwords [7], [3], [6]. In this attack it is assumed that an attacker has managed to obtain a copy of the password file; thus, the attacker can work off-line and do extensive searches. For every user the password file contains the user-id and the graphical password; recall that the stored graphical password consists of the cryptographically hashed value of a  $2c$ -dimensional grid point, together with the (non-encrypted) number of the grid to which this grid point belongs (one grid out of  $2c + 1$ ); here we follow the “improved implementation” of Section 3. It is assumed, that the attacker knows the algorithms of the graphical password system, including the cryptographic hash function. The goal of the attacker is to find the user-id and explicit password (in the clear) of some user, no matter which one. This attack is carried out by a computer, with limited human intervention.

For graphical passwords, as for alpha-numeric passwords, we assume that the collective attack attempts to make use of a known set of password candidates (the “dictionary”); this set should be small enough to be exhaustively searchable, and it should have a non-negligible probability of intersecting with the set of passwords that are actually used by users.

For graphical passwords there are no “dictionaries” of click point sequences in existence. In order to construct such a dictionary, the attacker needs to discover regularities in human click patterns. The attacker could find a good approximation of the effective password space, but this space is still too large to be searched exhaustively, as we saw in the previous subsection. Moreover, recent studies [12],

[13] (whose main focus was usability, not security), suggest that human users use a large collection of click points, and that there is no obvious subset of the memorable click points that are chosen with very high probability; however, on-going studies, focused on security, will give us better estimates.

If it is not possible to obtain a small enough dictionary by just restricting the set of click points, the only possible regularity that the attacker is left with is the possibility that humans might follow patterns in the way they combine and sequence their click points; e.g., the click points could perhaps be lined up along an imaginary curve. Further experiments will be needed to find out whether some humans follow such a strategy, and whether such patterns have enough regularity and predictability to be candidates for an attack dictionary. Further studies and experience will reveal whether dictionary attacks can be dangerous, or alternatively, how many click points are needed to thwart dictionary attacks.

- *Attack against individuals*

In this attack the goal is to find the password of a particular user; it is assumed that the attacker is in possession of the password record of this user (from the password file). It is also assumed, rather vaguely, that the attacker has some personal information about this user but has no direct information about the password itself. This attack is carried out by a computer-aided human. Experience and human testing will show whether it is feasible to guess what click points (and combinations of click points) a user may or may not choose, if one knows this user well.

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

**New Developments:** Since the submission of our paper, new publications have appeared. However, these are related to different types of password schemes than the one in our paper: [15], [16], [17].

## References

- [1] G. Blonder, "Graphical Password", US Patent 5559961 (1996).
- [2] R. Dhamija, A. Perrig, "Déjà Vu: User study using images for authentication", 9th *Usenix Security Symposium* (2000).
- [3] 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.
- [4] Brad Isaacson, "The password problem", Honors Project, Rutgers University at Camden (June 2001), under the direction of J.C. Birget.
- [5] I. Jermyn, A. Mayer, F. Monroe, M. Reiter, A. Rubin, "The design and analysis of graphical passwords", 8th *Usenix Security Symposium* (1999).
- [6] D. Klein, "A survey of, and improvements to, password security", *UNIX Security Workshop II*, Berkeley, Calif., Usenix Association (1990).
- [7] R. Morris, K. Thompson, "Password security: a case history", *Communications of the ACM* 22 (1979) 594-597.
- [8] "The science behind Passfaces", Real User Corporation (Sept. 2001).  
<http://www.realuser.com>

- [9] M. Boroditsky, “Passlogix password schemes”. <http://www.passlogix.com>
- [10] 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)*.
- [11] L. Sobrado, J.C. Birget, “Graphical passwords”, *The Rutgers Scholar*, vol. 4 (2002). <http://RutgersScholar.rutgers.edu/volume04/contents.htm>
- [12] S. Wiedenbeck, J. Waters, J.C. Birget, A. Brodskiy, N. Memon, “PassPoints: Design and evaluation of a graphical password system”, to be submitted. See <http://clam.rutgers.edu/birget/grPssw/index.html>
- [13] S. Wiedenbeck, J. Waters, J.C. Birget, A. Brodskiy, N. Memon, “Authentication using graphical passwords: Effects of tolerance and image choice”, to be submitted. See <http://clam.rutgers.edu/birget/grPssw/index.html>
- [14] Chai-Wah Wu, “On the design of content-based multimedia authentication systems”, *IEEE Transactions on Multimedia* 4(3) (Sept. 2002) 385- 393.
- [15] D. Davis, F. Monroe, M. Reiter, “On user choice in graphical password schemes”, *13th Usenix Security Symposium* (2004) 151-164. [Appeared after submission of our paper.]
- [16] V. Roth, K. Richter, R. Freidinger, “A PIN-entry method resilient against shoulder surfing”, *11th ACM Conf. on Computer and Communication Security* (2004) 236-245. [Appeared after submission of our paper.]
- [17] J. Thorpe, P. van Oorschot, “Graphical dictionaries and the memory space of graphical passwords”, *13th Usenix Security Symposium* (2004) 135-150. [Appeared after submission of our paper.]

**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