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 has 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; another example is digital watermarking, where a slight change in the features can produce a different watermark.

We present a discretization method which is tolerant to uncertainties (described by an error distance $r$, which can be adjusted to the applications). This enables us to implement very flexible graphical password schemes, and digital watermarking.

(Abstract still to be completed.)

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 $g : [0,a] \times [0,b] \rightarrow [0,1]$, 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 the rectangle $R_2$ 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

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

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

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

This subdivides $R_2$ into little grid squares of side-length $q$; near the edges of $R_2$ the grid squares are truncated.

*JCB supported by NSF grant DMS-9970471. JCB and DH supported by ISATC pilot grant from Rutgers University. NM supported by ? ? ?
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 grid square to the grid point $(m, n)$.

Usually, the goal in the design of a good discretization is to minimize the “quantization error”, i.e., some 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]. Our goal is quite different; we are more concerned with the stability of the discretization than with the error. 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 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 differences 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 is needed is a robust discretization in which there is no edge problem.

## 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 turns out 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 $D_r(x_0, y_0) = \{(x, y) : \|(x_0, y_0) - (x, y)\| \leq r\}$, where $\|\cdot\|$ 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 like for the integers, we define the mod operation for reals $t$ and $q$ (assuming $q \neq 0$) as follows:

$$t \mod q =_{\text{def}} t - \lfloor t/q \rfloor \cdot q$$
Lemma 2.2  A point \((x, y)\) is \(r\)-safe in a grid \(G\) with quantum \(q\) and offset \((\varphi, \psi)\) iff

\[
\begin{align*}
\begin{cases}
    r \leq (x - \varphi) \mod q < q - r \\
    r \leq (y - \psi) \mod q < q - r
\end{cases}
\end{align*}
\]

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.

![The three grids \(G_0, G_1, G_2\) (A is safe in \(G_0\), B is safe in \(G_1\) and in \(G_2\))](image)

Fig. 1: The three grids \(G_0, G_1, G_2\) (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:

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 \(i = 1, \ldots, d\),

\[
r \leq (x_i + 2rk) \mod (2r(d + 1)) < r(2d + 1).
\]

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 the \(r\)-safety condition takes 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 \(h_o \in \{0, 1, \ldots, d\}\):
\[
\begin{align*}
[2rh_o + r, 2r(h_o + 1) + r[ &\subset ]x_{i_o}, x_{i_o+1}[ , \text{ with } i_o < d, h_o < d, \text{ or} \\
[2rd + r, 2r(d+1)[ \cup [0, r[ &\subset ]x_d, 2r(d+1)[ \cup [0, x_1[ \text{ (here, } i_o = h_o = d).}
\end{align*}
\]

Proof of the Claim. There are \(d\) numbers \(x_j (j = 1, \ldots, d)\), and \(d+1\) disjoint sets
\[
S_h = [2rh + r, 2r(h+1) + r[ (h = 0, 1, \ldots, d-1) \text{ and}
\]
\[
S_d = [2rd + r, 2r(d+1)[ \cup [0, r[ .
\]
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 the set \(S_{h_o}\).

Proof of the Theorem.
Case A: \([2rh_o + r, 2r(h_o + 1) + r[ \subset ]x_{i_o}, x_{i_o+1}[ , \text{ 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 \(2r(d+1)\),
\[
r \leq (x_j + 2rk) \mod (2r(d+1)) < 2r(d-h_o) (< 2rd + r),
\]
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[ \subset ]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 \((x_i = c_{i,k})\) \((i=1,\ldots,d)\) 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.

**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 $\gamma : R_d \rightarrow \{0, 1, \ldots, d\}$ be a 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

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

So the grid map yields a grid identifier $k = \gamma(x_1, \ldots, x_d) \in \{0, 1, \ldots, d\}$ and a grid-point 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. [10], [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 [8]. Another version of click regions, this time with movement, appears in [5]. Somewhat different graphical password schemes (based on drawings, similar to a manuscript signature) were introduced and analyzed in [6]. Yet other graphical password schemes exist, based on image recognition [7], [9], [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. Of course, 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. 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 \( r = 2 \) mm would be fine.

Next, an 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 \( x_i + y_i \mod 2 \), or \( x_i + y_i \mod 3 \)). Random choice is better than ‘best-fit’ because it hides as much information as possible from attackers. This defines the function \( \gamma \) 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 the sequence of grid identifiers

\[
\gamma(x_1, y_1), \ldots, \gamma(x_c, y_c)
\]

and

\[
\text{HASH}(g_{\gamma(x_1,y_1)}(x_1, y_1), \ldots, g_{\gamma(x_c,y_c)}(x_c, y_c))
\]

in the user’s password record. The secret consists of the sequence of grid points (not in the sequence of grid identifiers).

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 an arbitrarily chosen image, and will eventually reject the user.)

The user then makes a sequence of clicks on the image. For click number $i$ 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 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.

Security analysis

The security features and advantages of graphical passwords, in general, are quite well analyzed in [6]. Let’s look at our numbers. For an image of size $330 \times 250$ mm$^2$ (for example), with safety margin $r = 1$ mm and quantum $q = 6r = 6$ mm, each grid has about 2290 grid points. For graphical passwords with 5 clicks the number of possible passwords is therefore $2290^5 \approx 6 \times 10^{16}$. If $r = 2$ mm, each grid has about 570 grid points; with 6 clicks the number of possible passwords is then $570^6 \approx 3 \times 10^{16}$. Compare with alpha-numeric passwords (digits and upper-case and lower-case letters) of length 9; here the number of passwords is $62^9 = 1.3 \times 10^{16}$.

However, regarding the size of the password space, the main advantage of graphical passwords is that they will be chosen much more randomly than alpha-numeric passwords; [6] gives a good discussion of this point (however, systematic human factors analyses of graphical passwords have not been done so far).

Compared to graphical passwords of Blonder’s type (with predefined click regions in pre-designed images), our graphical passwords have the advantage of giving the users more flexibility and convenience; this makes our passwords easier to remember too; at the same time, we have much larger password spaces.

Another difference between our graphical passwords and Blonder type passwords is the grid sequence, of numbers 0, 1, 2 (denoted $(\gamma(x_1, y_1), \ldots, \gamma(x_c, y_c))$ above), that arizes from robust discretization, and that is not kept secret. What can an attacker learn from the grid sequence?

In a $6r$-by-$6r$ grid square, only a $4r$-by-$4r$ sub-square is $r$-safe (i.e., at distance $\geq r$ from the edges); 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 (say $G_0$) is used in a certain click, the unsafe region of this grid are ruled out; so this restricts the possibilities to a little over 44.4% of the screen. However, the attacker learns nothing about where in these 44.4% of the screen the click occurs (recall also that if two or three grids are safe at the same time, the system chooses one of them at random). Moreover, the only thing that matters for the password is which grid square the click is in, not where in the grid square the click is; the attacker learns nothing about which grid square was clicked. Of course, a human could try to apply human psychology to guess what places in these 44.4% of the screen might attract the user; but the chances seem slim that this will succeed for 5 or more clicks since usually there are many grid squares that contain interesting features. For a computerized attack, any place in these 44.4% of the screen are as likely as any other.

In summary, the only weakness created by robust discretization is with regard to human
psychological attacks. However, the password space is much too large for such attacks to succeed.

**Improved password 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 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 a 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 $\log_2(2c + 1) = \log_2 11 = 3.4594316$ for the improved method; i.e., approximately 3.5 bits are lost. For the first implementation, the grid information is $\log_2 3^c = c \log_2 3 = 5 \times 1.5849625 = 7.9248125$; 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.

A general drawback of all graphical password schemes is **shoulder surfing**, which happens when an attacker watches a person log in. See [11] for a graphical password scheme that is resistant to shoulder surfing, at the expense of a slower login procedure.

## 4 Other applications

**THIS SECTION IS INCOMPLETE**

Robust discretization can be useful whenever we need a stable discretization, in the presence of uncertainty in the data or in the data processing.

### 4.1 Robust Authentication of Multimedia

Authentication techniques provide a means of ensuring the integrity of a message. The recent proliferation of digital multimedia content has raised concerns about authentication mechanisms for multimedia data. For example, consider digital photography, which is fast replacing conventional analog techniques. In the analog world, an image (a photograph) had generally been accepted as a "proof of occurrence" of the depicted event. The proliferation of digital images and the relative ease, by which they can be manipulated, has changed this situation dramatically. Given an image, in digital or analog form, one can no longer be assured of its authenticity. This has led to the need for image authentication techniques.

Authentication techniques that provide a means of ensuring the integrity of a message have been studied in cryptography for a few decades. However, given the large amount of redundancy present in image data, and consequently the large number of different representations of
perceptually identical content, image authentication presents some unique problems. Specifically, the data representing the image content can be changed without effecting a change that is actually perceptible; further, changes to the data can also perceptible, but may not affect the content. The problem here of course is that "image content" is itself an extremely ill defined quantity despite numerous attempts by vision and compression researchers. However, clearly certain image processing operations do not affect the underlying content. For example, one can brighten an image, lossy compress it, or change contrast settings. The changes caused by these operations could well be perceptible, even desirable, but the image content is not considered changed - people, if any in the image, are in the same positions; the clothes they are wearing as well as the geographical setting are recognizable. This leads to the notion of robust authentication mechanisms for digital images which tolerate small changes made to the image representation. In this section we show how robust quantization can be used to develop robust authentication techniques.

There have been a number of recent attempts at authentication which address authentication of "image content" and not of only image data. Such techniques can be broadly classified as taking one of two different approaches towards the development of content authentication techniques. The first approach involves computing a "robust hash" of an image. This hash can then be encrypted by a symmetric key to yield a message authentication code or by an asymmetric key to yield a digital signature. The MAC or signature itself can then be attached to the image as a tag or imbedded into the image using a suitable digital watermarking technique. Irrespective of this process, the basic idea behind this approach is to construct a robust hash function such that similar images will result in similar hashes (see figure ??).

Perhaps the best known technique based on this approach was proposed by Fridrich [?]. In this technique, N random matrices are generated with entries uniformly distributed in [0,1], using a secret key. Then, a low-pass filter is applied to each of these random matrices to obtain N random smooth patterns as shown in Figure ?? . These smooth patterns are then made DC free by subtracting their respective means to obtain $P_i$ where $i = 1, \ldots, N$. Then an image block, B, is projected on to each of these random smooth patterns. If a projection is greater than zero then the hash bit generated is a '1' otherwise a '0' is generated. In this way N hash bits are generated. These hash bits are then encrypted with a private key and embedded into the image using a robust watermarking technique.

Since the patterns $P_i$ have zero mean, the projections do not depend on the mean gray value of the block and only depend on the variations within the block itself. The robustness of this bit extraction technique was tested on real imagery and it was shown that it can reliably extract over 48 correct bits (out of 50 bits) from a small 64×64 image for the following image processing operations: 15% quality JPEG compression (as in PaintShop Pro), additive uniform noise with amplitude of 30 gray levels, 50% contrast adjustment, 25% brightness adjustment, dithering to 8 colors, multiple applications of sharpening, blurring, median, and mosaic filtering, histogram equalization and stretching, edge enhancement, and gamma correction in the range 0.7-1.5. However, operations like embossing and geometrical modifications, such as rotation, shift, and change of scale, lead to a failure to extract the correct bits.

The second approach for developing a robust authentication technique, first given in [?] is as follows: compute a set of features and quantize the features using a suitable quantization
step size that does not affect the visual quality of the image\textsuperscript{1}. The image resulting from this quantized set of features is then treated as the ”original” image which is then authenticated by conventional cryptographic techniques like hash functions and digital signatures. In order to check the authenticity of a potentially altered image, the same quantization step is first carried out and then the cryptographic hash function computed. As long as the quantized features are the same as the original, the image will be considered authentic, otherwise not. Clearly, the amount of ”change” tolerated by the authentication technique is directly related to the quantization step size. Figure ?? illustrates the basic framework of this approach. The framework itself is quite general and can be used with a variety of feature sets and ”allowable operations”.

The key problem in the approach presented in [?] is that the authentication process incurs distortion. That is, features from the ”original” image are first quantized and this new quantized version is then treated as the original image for the purposes of authentication. This is clearly undesirable. The authentication process should not distort the original. It should be noted that quantization is needed prior to computing an authenticator (like a hash value) due to the same stability or edge problem described in section 1. That is, if a feature point is close to the quantization boundary then slight changes to the image will lead to significant changes in the discretization and would defeat the goal of robust authentication. However, the robust discretization procedure we have presented in the previous section can alleviate this problem to a large degree. In figure ?? we show a robust authentication process that now incorporates robust discretization.

Perhaps we need experimental results now.

2. Uncertainty in the processing: Marking up images, adding references or text to features in an image (when different people refer to the same feature, they will not point to exactly the same pixel; so, when they create their reference, the system robustly discretizes the pointing).

Acknowledgements. This paper benefited from our previous work with Brad Isaacson, and from discussions with Leonardo Sobrado.

References

http://biron.usc.edu/~beferull/research.html


\textsuperscript{1}a dithered quantization scheme may need to be performed here


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