Click Passwords

Darko Kirovski, Nebojša Jojić, and Paul Roberts

Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
{darkok,jojic,proberts}@microsoft.com

Abstract. We present a set of algorithms and tools that enable entering passwords on devices with graphical input (touch-pad, stylus, mouse) by clicking on specific pixels of a custom image. As one of the most important features, when entering a password, the user is given limited tolerance for inaccuracy in the selection of pixels. The goal of the proposed click password system is to maximize the password space, while facilitating memorization of entered secrets. Besides enabling personalization of the login procedure through selection of the background image, the proposed system provides superior password space compared to traditional 8-character textual passwords.

1 Introduction

Traditional textual passwords leave a lot to be desired with respect to the two most important characteristics of any password system: security and usability. Whereas a truly random 8-character password potentially may produce a reliable $95^8 > 10^{15}$ password space, memorizing and entering such a password on a keyboard is an uncomfortable task that only very few practice. The fact that many users chose textual passwords that are relatively easy to guess has been analyzed and emphasized in several studies [9], [1], [13], [2]. By analyzing 14,000 UNIX passwords, Klein has concluded that nearly one quarter of them belonged to a relatively small dictionary of $3 \cdot 10^6$ words [9]. Thus, in the past decade hackers have focused on utilizing "the dictionary attack" along with buffer overflow and packet sniffing more than any other type of attack [11]. Textual passwords have been used as a de facto standard login mechanism, primarily because of the dominance of the keyboard as an input device. With the recent proliferation of mobile devices which typically have some form of graphical input (touch-pad, stylus) and graphics display, many routines, so as the login process, require adaptation to point-and-click mediated human-computer interaction.

The idea of using visual content to create password systems is not new. Several other techniques have been developed to date with such a goal: (*i*) icon-based passwords [3], (*ii*) passfaces [10], and (*iii*) draw-a-secret [8]. The first two systems use self-contained icons and facial photos as symbols that the user selects while creating a password. In a sense, such systems mimic a keyboard with letters replaced by distinct self-contained images. Draw-a-secret passwords are a step forward as they are conceived by creating a secret drawing on a grid of rectangles. The motion of the stylus across rectangles is encoded into a password.

The difficulty in adopting visual password systems stems from several facts. First, in case of icon-based systems (*i-ii*), large area of the display is reserved to clearly distinguish the tiled icons. In order to have easily recognizable content, individual icons usually contain entire objects, whereas the granularity of visual detail that can be semantically selected by the user is commonly far smaller. Icon-based systems cannot enable users to select an arbitrary region of an image as a password construct. As a consequence, such systems rarely explore the real potential of visual content to build password systems.

Second, due to the nature of point-and-click user interfaces, in case of both image-based (e.g., *i-ii*) and in particular draw-a-secret (*iii*) systems, users should be given certain tolerance to an input error while entering a password. Tolerance to misaligned input is difficult to achieve because password systems do not memorize password's plain-text but its cryptographically secure hash. Thus, a slight change in the entered password results in a hash value which is at a large distance from the hash of the set up password. As a consequence, graphics-based password systems are typically either not secure due to a small number of symbols just as textual passwords or have a high likelihood of erroneous logins.

Fig. 1. An image and the result of image analysis. The darker regions are estimated to be more likely to be used as symbols in a click password. Cardinality of the password alphabet for this high-resolution image exceeds 500 symbols.

2 Click Passwords

In order to address the main concerns of visual password systems, we introduce *click passwords* – a login mechanism that uses an arbitrary image as a domain for creating a password. The user creates a password by clicking on several arbitrary pixels of the image. At login, the user is required to click in a certain neighborhood of the originally selected pixels in the same order as during the password setup. A desired tolerance to input error can be adjusted by the user before the password is set up. Since the image is not visibly partitioned into regions, the user is free to select and memorize arbitrary visual features of the image. The task of the underlying mechanisms is to maximize system security for a given image while providing robustness to limited erroneous input.

A click password system uses the domain image to build an alphabet of distinct symbols. The alphabet is built by tiling the image with Voronoi polygons

where each polygon corresponds to a unique symbol. The tiling is not displayed to the user; it is used within the click password system to convert user clicks into an string of symbols that is hashed and stored during password setup. The user sets a password by selecting an ordered set of pixels. Each selected pixel is translated into a symbol that corresponds to the identifier of the polygon which contains the pixel. For each click password, the system stores three objects: (a) the **username**, (b) the **hash** value of the arithmetic concatenation of the selected symbols, and (c) for each selected pixel, an alignment **offset** of the polygon grid that moves the selected pixel to a random pixel in the neighborhood of the center of the polygon that contains the selected pixel. The offset is stored as plain-text and it guarantees limited tolerance to pixel selection inaccuracy at logon. Radius of the neighborhood central to a polygon equals the maximum tolerance to error that can be adjusted by the user to a desired value. Since the offsets are stored as plain-text, their values reveal information that can reduce the complexity of a brute force attack (e.g., see example in Figure 2). Thus, a click password system is tailored to a given image in two steps: first, a cognitive system extracts a map of pixels likely to be selected as part of a password and then, an optimization system creates a Voronoi polygon tiling of the image that maximizes the entropy of a selected password under the assumption that the adversary conducting a brute force attack may have access to the password offsets.

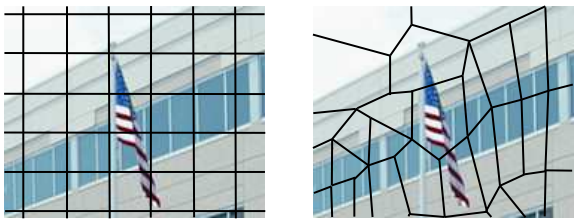

Fig. 2. In this image, one of the pixels likely to be clicked is the top of the flag. In the left subfigure, this pixel is in the upper left corner of a rectangular cell. An offset that moves this pixel to its polygon center is most likely pointing to this cell, as most of the other cells have clickable pixels in other regions. An optimized Voronoi tiling in the right subfigure rectifies this problem by creating a grid in which pixels likely to be clicked are placed near the centers of each polygon.

The click password system has two goals: **improved security** and **usability**. The expectation is that an image with visually diverse content can create a symbol alphabet of significantly higher cardinality than the alphabet that consists of printable characters of a standard US keyboard. Similarly, understanding cross-symbol dependencies in text is commonly attributed to the fact that most passwords have semantic value, whereas clicks on an image, at least intuitively, may not have strong and well-defined interactions. Using images with content tailored for a particular user, click passwords are supposed to be significantly easier to memorize than textual secrets of comparable level of security.

We have developed a set of algorithms that initiate the click password system as follows. During password setup, the user selects an image as a domain for

pixel selection. Next, a component of the platform, the *image analyzer*, evaluates for each pixel the likelihood that it can be selected as part of a password. For brevity, in this version of the manuscript we do not overview this component. The image analyzer also quantifies, as feedback to the user, whether the selected image contains sufficient graphical diversity to enable the desired level of security. Finally, based on the information provided by the image analyzer, the *grid designer* builds a grid of enumerated Voronoi polygons that tile the image such that the cardinality of the password space is heuristically maximized. An example of an image that we have used in our user study as well as the corresponding output of our image analyzer is presented in Figure 1.

3 Related Work

Traditional approach to logging onto a system is to type in a password associated with the username. When a password is entered for the first time, the system computes a secure hash of the entered password and stores it into one of the system files (e.g., on UNIX systems `/etc/passwd`). Upon login, the system computes again the hash of the entered password and compares this value with the stored one. In UNIX, the password hash is computed by encrypting 64 zero bits with the password using 25 rounds of DES. If they match, the user is granted access. In order to prevent one user from identifying another with the same password, before hashing, passwords are usually salted with a user-specific random variable (two characters in UNIX) also stored in the `passwd` file.

Although such a login mechanism has provable reliability [6], one of the most common related attacks is the dictionary attack [9]. The extent of dictionary attacks can be grasped from hacker web sites which commonly provide 100+ password cracking utilities [11]. The dictionary attack can be launched on- and off-line. The success rate of an on-line attack can be made low by prohibiting access to the system after a certain number of unsuccessful logins. It is difficult to prevent the off-line variant of the dictionary attack, where the adversary obtains the `passwd` file through a Trojan horse or as a system user.

4 Security of Click Passwords

Initially, the user selects the image $\mathcal{I}$ used as a domain for password selection. Next, the system builds the passwords space by tiling the image using a grid Π of L non-overlapping polygons. The polygons can be of arbitrary sizes. The grid is not displayed to the user. It is constructed with polygons of different size such that each polygon is approximately equally likely to be clicked. Given the map that quantifies the likelihood that a pixel is selected as part of a password, this approach aims at maximizing the password space. Such a map can be constructed empirically or using image processing for feature extraction [5].

Definition 1. Image Grid. *Given an image $\mathcal{I}$ of $m \times n$ pixels with a weight map $W = \{\mathbb{R}\}^{m \times n}$, where each weight $w(x, y) \in W$ corresponds[1] to pixel $P(x, y)$*

[1] For $P(x, y)$, we denote its weight as $w(x, y)$ or $w(P)$.

with x and y as coordinates, an image grid $\Pi(\mathcal{I}) = \{\pi_i, i = 1..L\}$ is defined as tiling of $\mathcal{I}$ into L non-overlapping polygons.

The weight $w(x, y)$ associated with pixel $P(x, y)$ equals the probability that this pixel can be selected while setting up a password. Thus, matrix W quantifies the visual effect of the image on the selectivity of certain pixels. The user sets up a p-long click password by selecting an ordered set C_p of p pixels.

Definition 2. Password Setup. *Given an image $\mathcal{I}$ of $m \times n$ pixels, a click password C_p of length p is set up as an ordered set C_p of p user-selected pixels.*

At logon time, the user selects an ordered set D_p of p pixels which are required to be in the ε_2-neighborhood of the pixels selected at password setup time C_p. Pixel P is in the ε_2-neighborhood of pixel Q if the Euclidean distance between them is $||P - Q|| \leq \varepsilon_2$ [2]. We enable such bounded tolerance with a requirement that the pixels in D_p map to the same polygons to which pixels from C_p map.

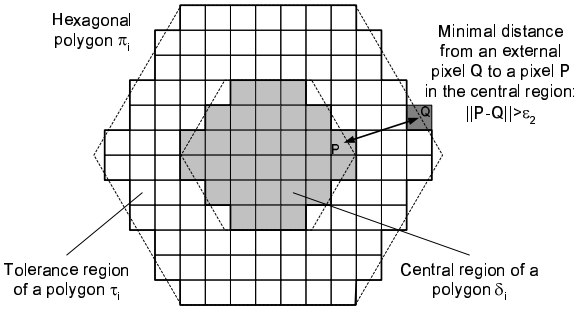

Fig. 3. An example of a hexagonal polygon π_i, its central δ_i (grey pixels) and tolerance τ_i region (white pixels). Minimal distance ε_2 between an external pixel Q and an internal pixel $P \in \delta_i$ must be greater than ε_2.

Definition 3. Central Region of a Polygon. *Given a polygon π_i, its central region δ_i is defined as a non-empty subset of pixels $\delta_i \subset \pi_i$ for which the minimal Euclidean distance with respect to any pixel outside the polygon is greater than a constant real number ε_2.*

Definition 4. Tolerance Region of a Polygon. *Given a polygon π_i and its central region δ_i, the corresponding tolerance region is defined as: $\tau_i = \pi_i - \delta_i$.*

An example of a hexagonal polygon and its central and tolerance region is presented in Figure 3. For the example polygon in the figure, its central region is defined using $3 \leq \varepsilon_2 < \sqrt{10}$ assuming pixel dimensions are 1×1.

We define the two regions of a polygon because of the following problem. During password setup, the user may select a pixel P which is in the tolerance region of its containing polygon π_i. This means that at logon, the user may click a pixel Q which is at distance $||P - Q|| \leq \varepsilon_2$ from P but does not belong to π_i.

[2] $||P(x_p, y_p) - Q(x_q, y_q)|| \equiv \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$.

To avoid this situation, at setup time, the system records an offset associated with P which realigns the grid such that P is in the central region of π_i. Thus, the central region of a polygon must contain at least one pixel.

Definition 5. Click Password Storage. *For a click password* $C_p = \{P_i \in \mathcal{I}, i = 1..p\}$*, the data* $\{\psi_0, \psi_1, \psi_2\}$ *is stored on the host computer, where* ψ_0 *is a pointer to a user,* ψ_1 *equals:*

$$\psi_1 = \mathrm{CSH}\left(\sum_{i=1}^{p} L^{i-1} \cdot (j | P_i \in \pi_j \wedge P_i \in C_p)\right) \tag{1}$$

where CSH *is a cryptographically secure hash function such as SHA-256, and* ψ_2 *is an ordered set of p pairs of integer numbers* $\psi_2 = \{\mathbb{Z}, \mathbb{Z}\}^p$*, where i-th pair* $\psi_2(P_i) = \{a_i, b_i\} \in \psi_2$ *is denoted as the offset of a password pixel* $P_i(x_i, y_i) \in \pi_j$ *and equals:*

$$\psi_2(P_i) = \mathrm{rand}\left[\left(\bigcup_{\forall P_k \in \delta_j} \{(x_k - x_i), (y_k - y_i)\}\right) \cap \Psi\right], \tag{2}$$

where x_k *and* y_k *are the coordinates of* P_k*,* $\mathrm{rand}(t)$ *returns a random element from the set t, and* Ψ *is a set of all possible offsets that can occur in* Π*:*

$$\Psi = \{0, 0\} \cup \left(\bigcup_{\forall \pi_i \in \Pi} \bigcup_{\forall P_j \in \tau_i} \{(x_q - x_j), (y_q - y_j)\}\right), \tag{3}$$

$$Q = (x_q, y_q) = \arg\min_{Q \in \delta_i}(||P_j - Q||). \tag{4}$$

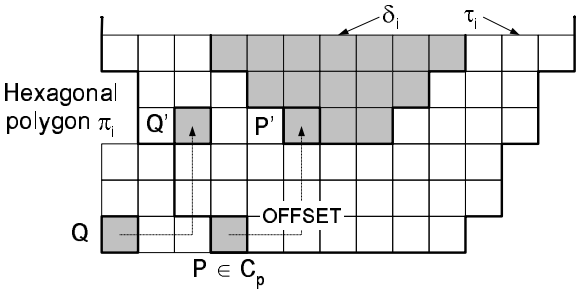

Fig. 4. An example of using an offset to realign a polygon grid such that each selected pixel of a click password is located in the center region of the containing polygon. Pixel P is originally selected in the tolerance region τ_i of π_i. Offset $\psi_2(P) = \{2, 3\}$ realigns P to P' which belongs to δ_i. Pixel Q entered at logon is at a distance of $||Q - P|| \leq \varepsilon_2$, hence this error should be tolerated. However, Q does not belong to π_i which results in an incorrect selection. Using the offset, the translated pixel $Q' = Q + \psi_2(P)$ results in a correct selection as it belongs to π_i.

For a given ε_2, the set of all possible offsets Ψ in Π is limited to:

$$\Psi \subseteq \{\{a, b\} | a, b \in \mathbb{Z}, |a| \leqslant \varepsilon_2, |b| \leqslant \varepsilon_2\}. \tag{5}$$

A password is stored as a 3-tuple of the username, the hash of the encoded selection of polygons, and the grid offsets used to realign the image grid prior to selecting each pixel at logon time. The data hashed in Eqn.1 can be salted and the salt can be stored as a fourth component of the stored password entry. The alignment offsets ψ_2 are randomized to disable analysis which may exclude certain polygons from the password space. An example of how offsets are used to enable correct password entry with ε_2-tolerance is presented in Figure 4.

A pixel (polygon) is correctly selected at logon if it is within a distance smaller or equal to ε_2 from the pixel selected at setup. Logon is granted, iff the ψ_1 function of the entered pixels D_p realigned for ψ_2, equals the stored $\psi_1(C_p)$.

Definition 6. Click Password Logon. *Let C_p be a selected p-long click password and let D_p be an ordered set $D_p = \{Q_i(x_i, y_i), i = 1..p\}$ of p pixels selected at logon time. System access is granted only if:*

$$\psi_1(C_p) \equiv \mathrm{CSH}\left(\sum_{i=1}^{p}[L^{i-1} \cdot (j|(Q_i + \{a_i, b_i\}) \in \pi_j)]\right) \tag{6}$$

where $\{a_i, b_i\}$ corresponds to the i-th offset pair in ψ_2 and $P(x, y) + \{a, b\}$ equals $P(x + a, y + b)$.

4.1 Security Metrics

Once constructed, the image grid creates a certain password space with respect to the content of the image and the potential offsets. It is important to quantify this metric as it directly impacts the security of the system against brute-force attacks. Naïve computation of the password space would raise the number of polygons to the power of the number of pixels in a password. However, it is not likely that an image can provide sufficient visual diversity, such that each polygon in a grid of regular polygons is selected with equal probability. In addition, the security of the system must be accounted for the fact that the grid structure and grid offsets ψ_2 may be obtained by the adversary. In order to provide better insight on the security of click passwords, we measure the entropy of polygon selection. A grid of L polygons Π which results in maximum entropy, is the objective of the grid designer.

Definition 7. Entropy of Polygon Selection. *Given an image $\mathcal{I}$, its weight map W, and a grid of L polygons $\Pi(\mathcal{I}) = \{\pi_i | i = 1..L\}$, the entropy of selecting a polygon equals:*

$$H(\Pi) = -\sum_{\forall\{a,b\}\in\Psi}\sum_{i=1}^{L}\vartheta(\{a, b\}, \pi_i) \cdot \log_2(\vartheta(\{a, b\}, \pi_i)), \tag{7}$$

where Ψ is the set of all possible offset combinations in the grid structure (see Eqn.3) and $\vartheta(\{a, b\}, \pi_i)$ denotes the probability that a pixel from polygon π_i is selected with an offset $\{a, b\}$ and equals:

$$\vartheta(\{a, b\}, \pi_i) = \sum_{\forall P_j\in\pi_i | P_j+\{a,b\}\in\delta_i} w(P_j). \tag{8}$$

Brute Force Attack. The general assumption for any attack on a click password system is that the system file with the corresponding entries $\{\psi_0, \psi_1, \psi_2\}$ is available to the adversary as well as the domain image $\mathcal{I}$, its weight map W, and resulting grid structure Π. In order to find the list of polygons that constitutes a given password C_p, the adversary can launch the following brute force attack:

- In the first step, for each pixel $P_i \in C_p$, the adversary computes the subset of polygons $\Omega(P_i) \subseteq \Pi$ of minimal cardinality such that for the corresponding offset $\psi_2(P_i)$,

$$\sum_{\forall \pi_j \in \Omega(P_i)} \vartheta(\psi_2(P_i), \pi_j) < \varepsilon_3 \cdot \sum_{\forall \pi_j \in \Pi} \vartheta(\psi_2(P_i), \pi_j). \tag{9}$$

  In this case, ε_3 is a parameter that balances computational complexity and likelihood of success. Typically, $\varepsilon_3 > 0.9$.
- In the second step, the adversary generates the set of attack vectors as all possible p-long combinations of polygons $C_A = \{\pi_{A1}..\pi_{Ap}\}$, where each variable π_{Ai} takes the values of all polygons in the corresponding $\Omega(P_i)$. For each attack vector $C_a \in C_A$, the adversary computes $\psi_1(C_a)$ until it matches $\psi_1(C_p)$ retrieved from the password file. The cardinality of the set of attack vectors C_A equals: $|C_A| = \prod_{\forall P_i \in C_p} |\Omega(P_i)|$. In order to minimize the expected length of the search, the adversary tests test vectors from C_A with decreasing probability of occurrence.

Under the assumption that the weight map W used by the adversary is accurate, the likelihood of success for this attack is strongly governed by the cut-off threshold ε_3 and equals $\Pr[C_p \in C_A] = 1 - \varepsilon_3^p$.

4.2 Grid Designer

In order to maximize the difficulty of a brute force attack, the Voronoi grid Π has to be such that the entropy $H(\Pi)$ of polygon selection is maximized for a given tolerance ε_2 of pixel selection. From Eqn.7, we conclude that maximal entropy is achieved for a set Π of L polygons if the likelihood of selecting a particular offset within a polygon is equivalent across all polygons and for all offsets. This case can occur only if the weight map W is constant across all pixels. It is not likely that any image can provide visual diversity such that human eye can equiprobably select any pixel as a password symbol. Thus, certain variance in the probability that a polygon π has been selected with an offset $\{a, b\}$ must exist with respect to distinct polygons and offset values. For simplicity and brevity, we adopt the following model of the image weight map W. For each pixel P, its weight takes randomly one of the two values:

$$w(P) = \begin{cases} 0, \Pr[w(P) = 0] = 1 - \frac{1}{mn\mu} \\ \mu, \quad \Pr[w(P) = \mu] = \frac{1}{mn\mu} \end{cases} \tag{10}$$

where $0 < \mu \leq 1$ and m and n are image dimensions in pixels. In addition, we denote pixels with $w(P) = \mu$ as "clickable."

We demonstrate the trade-offs related to the size of the Voronoi polygons as well as their central regions using the following two examples. Consider the polygons π_1 and π_2 in Figure 5. Note that only a single pixel represents the central region δ_1 of polygon π_1. Hence, one of 25 distinct offset values $\{a, b\} \in \Psi$ is used to realign every pixel of π_1 to the pixel in δ_1. This means that a single recorded offset points to one pixel in each polygon. One advantage of tiling an image with polygons similar to π_1 is that there are more polygons due to their size. However, this comes at the expense of having a smaller ratio of polygons with a "clickable" pixel for a given offset value.

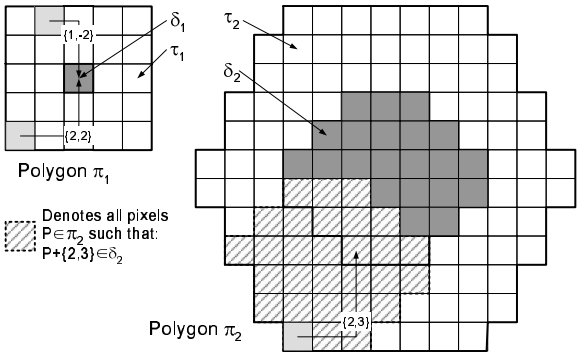

Fig. 5. An example of two polygons with central regions that consist of one and several pixels. An example of how offset is used to realign pixels selected within the tolerance region into pixels in the central region of the corresponding polygon.

Figure 5 also depicts a subset of pixels (shaded region) that belongs to polygon π_2, where each pixel in the subset can be moved for offset $\{2, 3\}$. Roughly, for any offset in Ψ, the cardinality of this subset equals approximately $|\delta_2|$. The likelihood that this subset contains a "clickable" point is substantially greater than in the case of π_1. This characteristic comes at the expense of increased polygon size which results in fewer polygons used in order to tile an image.

Preliminaries. In this subsection, we introduce several entities that facilitate the description of the tiling algorithm. First, we define an offset-$\{a, b\}$ region of a polygon as a subset of pixels within a region π which are translated with the offset $\{a, b\}$ into the central region of the polygon.

Definition 8. Offset-$\{a, b\}$ Region of a Polygon. *For a given polygon π with an associated central region δ, its offset-$\{a, b\}$ region, denoted as $\sigma_\pi(\{a, b\})$, is defined as a subset of pixels in π, such that for each pixel $P \in \sigma_\pi(\{a, b\})$, $P + \{a, b\} \in \delta$.*

Note that in general $|\sigma(\{a, b\})| \leq |\delta|$. For example, we have $|\sigma_2(\{2, 3\})| = |\delta_2| - 1$ in Figure 5. Next, we define how a Voronoi grid is constructed and represented for a digital image.

Definition 9. Pixel-based Voronoi Grid. *For an image $\mathcal{I}$, a Voronoi grid of L polygons, $\Pi(\mathcal{I}) = \{\pi_i, i = 1..L\}$, is defined with a set of L corresponding*

pixels $\Gamma = \{P_i, i = 1..L\} \subset \mathcal{I}$, $P_i \rightarrow \pi_i$, such that a given pixel $Q \in \mathcal{I}$ belongs to the polygon $\pi_j \in \Pi$, iff polygon's defining pixel, P_j, has the shortest Euclidean distance from Q with respect to all other pixels in Γ. If there are several pixels in Γ that share the same shortest distance from Q, they are sorted in the decreasing order of their x-ordinates and y-abscissas respectively, and the top-sorted pixel is selected.

According to Def.9, the grid is defined using the subset of pixels Γ. For a given click tolerance ε_2, the central region of each polygon is defined using Def.3.

Voronoi Polygon Tiling The goal of the tiling algorithm is to create a Voronoi grid as defined in Def.9, such that the entropy of polygon selection as defined in Def.7, is maximized. We heuristically solve this problem using a constructive algorithm which tiles the image using an 1-lookahead greedy strategy.

First, we revisit the optimization goal of the grid designer. Intuitively, polygon selection entropy is maximized if the cardinality L of the polygon set Π is maximized while within each polygon the minimal likelihood of occurrence of any offset from Ψ (see Eqns.3,1) is non-zero. For such a polygon grid and for a given offset, the adversary needs to consider all polygons in Π in its brute force attack. Such an approximation of the original optimization goal is effective because of two facts: "clickable" islands of pixels have relatively large mutual distances as the nature of human perception requires isolated graphical features to select them, and there are not more than a few "clickable" pixels per polygon, as we intend to keep the polygon size as small as possible.

This results in relatively small variance in the likelihood that a certain polygon is selected given a certain offset across all polygons in the grid. The quality of the final solution Π can always be verified via the computation of its true security metric, $H(\Pi)$, according to Eqn.7. Finally, the optimization objective can be generalized such that L is maximized under the condition that within each polygon $\varrho \cdot |\Psi|$ of offsets from Ψ have a non-zero likelihood of selection. For brevity and simplicity, we constrain $\varrho = 1$.

Pivotal part of the tiling algorithm are two maps: a binary coverage map $M_C = \{0, 1\}^{m \times n}$ where each element $M_C(x, y)$ denotes that pixel $P(x, y) \in \mathcal{I}$ has been covered during polygon tiling, and an integer polygon-size map $M_P = \{\mathbb{Z}\}^{m \times n}$, where each element $M_P(x, y)$ equals the minimal radius of the pixel-rasterized circle centered at $P(x, y)$ which has a non-zero likelihood occurrence of any offset in Ψ. The radius of the circle is at least $\varepsilon_2 + 1$ pixels.

For a given click tolerance ε_2 and a given pixel $P(x, y)$, its value $M_P(x, y)$ is computed in the following way. In the starting iteration, a polygon π of circular shape centered at $P(x, y)$ with radius $\varepsilon_2 + 1$ is created. If for all possible offsets in Ψ, their offset-regions contain at least one "clickable" pixel, then π is accepted as the resulting polygon and $M_P(x, y)$ is set to the value of polygon's radius. In the subsequent iterations, the radius of polygon π is increased until $2\varepsilon_2$. If a polygon with satisfactory characteristics is not found, then $M_P(x, y) = \infty$. Polygons larger than this maximal size are never selected explicitly during tiling, because a polygon with radius $2\varepsilon_2$ and a "clickable" pixel at $P(x, y)$, is guaranteed to

contain at least one "clickable" pixel for each possible offset-region. Once all polygons are selected, their borders are recomputed according to Def.9.

The goal of the Voronoi tiling algorithm is to find a max-cardinality subset Γ of pixels in $\mathcal{I}$ such that all polygons defined with Γ have non-zero likelihood of occurrence for any offset in Ψ. This problem is NP-complete as it can be mapped to the SET PACKING problem [7] (problem SP3, pp.221 in [7]) as follows. For each pixel $P(x, y)$, we create a set which encompasses all neighboring pixels covered by the polygon π centered at P and with radius $M_P(P)$. The domain of the problem is the collection of such sets for all pixels P with a finite value of their corresponding polygon-size map $M_P(P) \neq \infty$. The goal is to find the selection of mutually disjoint sets from this collection with maximal cardinality.

Voronoi tiling is performed in a sequence of steps as follows. In the first step, the coverage map M_C is initialized to zero, the polygon-size map M_P is computed as described, and finally the resulting set Γ is initiated to an empty set. The solution is built in a sequence of constructive iterations. In each iteration, we first compute the set Λ of all points which are not covered and with finite polygon-size map values. Per iteration, a single pixel $P(x, y)$ is added to the final solution – the added pixel has the following properties:

- it belongs to the set $\lambda_1 \subset \Lambda$, where each pixel $Q \in \lambda_1$ has a smaller or equal polygon-size value with respect to all other pixels in Λ and
- it has the largest value among all pixels in λ_1 for the following objective:

$$g(P) = \begin{cases} \infty & , M_C(P) = 1 \\ \frac{|\theta_2(P)|}{\eta(P)} - |\theta_1(P)| & , M_C(P) = 0 \end{cases} . \tag{11}$$

Sets $\theta_1(P)$ and $\theta_2(P)$ are created as a collection of "clickable" pixels in the circular polygon centered at P with radius $M_P(P) + \varepsilon_2 + 1$ and a collection of yet uncovered "clickable" pixels in the ring of pixels centered at P and with outer radius $M_P(P) + 2\varepsilon_2$ and inner radius $M_P(P) + \varepsilon_2 + 1$ respectively. More formally, sets $\theta_1(P)$ and $\theta_2(P)$ are defined as follows:

$$\theta_1(P) = \{Q \in \mathcal{I} \mid w(Q) > 0 \ \wedge \tag{12}$$

$$||Q - P|| \leq M_P(P) + \varepsilon_2 + 1\}, \tag{13}$$

$$\theta_2(P) = \{Q \in \mathcal{I} \mid M_C(Q) = 0 \ \wedge w(Q) > 0 \wedge \tag{14}$$

$$M_P(P) + \varepsilon_2 + 1 < ||Q - P|| \leq M_P(P) + 2\varepsilon_2\}. \tag{15}$$

The scalar ratio $\eta(P)$ quantifies the ratio of covered vs. total pixels in the above mentioned ring of pixels. Function $g(P)$ in Eqn.11 heuristically directs the search by enforcing intermediate solutions that have the following properties:

Least constraining – small number of "clickable" pixels in the circular polygon centered at P with radius $M_P(P) + \varepsilon_2 + 1$. By covering as few as possible "clickable" pixels with the selection of each Voronoi polygon, consequently the part of the image not yet covered with polygons, has as many as possible "clickable" pixels. This improves the likelihood for obtaining a better solution.

1-lookahead most constrained – the neighborhood of each selected polygon, i.e. the ring of pixels Q at distance $M_P(P)+\varepsilon_2+1 < ||Q-P|| \leq M_P(P)+2\varepsilon_2$ from P, proportional to its cardinality, should have as many as possible "clickable" pixels. Consequently, the likelihood that we find a better solution in the neighborhood of the current polygon, is improved.

After a pixel $P(x, y)$ is selected, all pixels $Q(x, y) \in \mathcal{I}$ at Euclidean distance $||Q-P|| \leq M_P(P)+M_P(Q)$ are marked as covered $M_C(Q) = 1$. The constructive iteration is repeated until all "clickable" pixels are covered or an additional polygon of minimal area cannot be added to Γ. The aggregated collection of pixels in Γ, defines the resulting Voronoi polygon tiling according to Def.9.

In summary, the problem behind Voronoi polygon tiling for maximized entropy of polygon selection can be modeled as SET PACKING – a task which is NP-complete. In order to generate effective solutions, we have derived a constructive heuristic with complexity linearly proportional to the number of "clickable" pixels in the considered image. In our experiments, the optimization of the Voronoi tiling resulted in an average 20% increase in the system entropy.

References

1. A. Adams et al. Users are not the enemy: Why users compromise computer security mechanisms and how to take remedial measures. *Comm. of the ACM*, Vol.42, no.12, pp.40–46, 1999.
2. W. Belgers. Unix password security. `http://www.ja.net/CERT/Belgers/UNIX-password-security.html`.
3. G. Blonder. Graphical passwords. United States Patent no.5559961, 1996.
4. S. Brostoff et al. Are passfaces more usable than passwords? *HCI*, 2000.
5. CVonline: Geometric Feature Extraction Methods. `http://homepages.inf.ed.ac.uk/rbf/CVonline/feature.htm`
6. D.C. Feldmeier et al. UNIX Password Security - Ten Years Later. *CRYPTO*, pp.44–63, 1989.
7. M.R. Garey and D.S. Johnson. Computers and Intractability. Freeman, 1979.
8. I. Jermyn et al. The design and analysis of graphical passwords. *USENIX Security Symposium*, pp.1–14, 1999.
9. D.V. Klein. Foiling the Cracker: A survey of, and Improvements to Password Security. *USENIX Security Workshop*, pp.5–14, 1990.
10. Passfaces. `http://www.realuser.com`
11. Password Portal. `http://www.passwordportal.net/`.
12. E.E. Schultz. Advanced Windows NT security: network security. *Computer Security J.*, Vol.15, no.3, pp.13–22, 1999.
13. J. Yan et al. The Memorability and Security of Passwords – Some Empirical Results. Tech. Report No.500, Computer Lab., University of Cambridge, 2000.