On Purely Automated Attacks and Click-Based Graphical Passwords*

Amirali Salehi-Abari, Julie Thorpe, and P.C. van Oorschot
{*asabari, jthorpe, paulv*}*@scs.carleton.ca*

Abstract

We present and evaluate various methods for purely automated attacks against click-based graphical passwords. Our purely automated methods combine click-order heuristics with focus-of-attention scan-paths generated from Itti et al.'s (1998) computational model of visual attention. Testing our method against previous work, it results in a significantly better automated attack, guessing 8-15% of passwords for two representative images using dictionaries of less than $2^{24.6}$ entries, and about 16% of passwords on each of these images using dictionaries of less than $2^{31.4}$ entries (where the full password space is 43 bits). Relaxing our click-order pattern substantially increased the efficacy of our attack albeit with larger dictionaries, allowing attacks that guessed 48-54% of passwords in less than 2^{35} guesses (compared to previous results of 0.9% and 9.1% on the same two images with 2^{35} guesses). These latter automated attacks are independent of focus-of-attention models, and in fact are based on image-independent guessing patterns. Our results show that automated attacks, which are easier to launch than human-seeded attacks and are more scalable to systems that use multiple images, pose a significant threat to Passpoints-style graphical passwords, and offer an effective alternative to human-seeded attacks.

1 Introduction

Graphical passwords are an alternative to traditional text passwords, whereby a user must remember an image (or parts of an image) in place of a word. They are motivated in part by the well-known fact that people are better at remembering images than words [18]. There are many different types of graphical passwords; among the more popular approaches is *click-based graphical passwords* [31, 15, 2, 24, 6], which require users to click on a sequence of points on one or more background images. The most effective attack strategy to date on these schemes appears to be human-seeded attacks [27], although such attacks are more difficult to arrange than attacks based on purely automated means and do not scale well for systems that use multiple images. In this paper, we pursue purely automated approaches for guessing attacks.

We pursue heuristic-based strategies for purely automated dictionary generation (e.g., based on click-order patterns), and strategies to prioritize these dictionaries using image processing methods to identify points that users are more likely to choose. We hypothesize that users will choose click-points according to a click-order pattern, to help remember the password as fewer "chunks" [8]. We further examine use of the DIAG click-order pattern [27], which captures arcs that are consistent in both horizontal and vertical directions, and a subset of this pattern that we call LINE that captures only horizontal and vertical lines. We relax the rules on these definitions, showing that a "lazy" approach to these click-order patterns is substantially more effective.

*Version: June 8, 2008. This work was carried out at Carleton University, School of Computer Science, Ottawa, Canada.

We further hypothesize that users will choose click-points based on their preference for certain points in the image, and that their preference for certain points will be influenced by how much they are naturally attracted to those points. *Attention* is the cognitive process of selectively focusing on one aspect of the environment while ignoring others, a mechanism that helps us prioritize sensory information. There are two different categories of visual attention models: bottom-up and top-down. *Bottom-up* visual attention captures how attention is drawn to the parts of a scene or image that are salient or conspicuous. It is what naturally draws us to look at the unexpected or different parts of a scene, prioritizing them from the other consistent parts. For example, if an image contains a large number of objects that are blue, and only one is yellow, human attention will instinctively focus on the yellow object. *Top-down* visual attention is task-dependent, based on cognitive, volitional control. With a priori knowledge about what object(s) to look for, our attention is brought to the parts of the scene containing those object(s). For example, if a user decides that people with dark hair are of interest for some reason, the user's attention would shift between objects with features that might indicate a dark-haired person.

Our contributions include the best purely automated attacks to date (to our knowledge) against click-based graphical passwords, an evaluation of how the model of Itti et al. [14] relates to user-selected click-based graphical passwords, and a new spatial clustering algorithm. Two different hypotheses were tested regarding how users might choose their click-points relative to Itti's model. Our methods were tested using the same field study database used by Thorpe et al. [27], allowing us to compare performance. We found that a "lazy" approach to click-order patterns produced a substantially better automated attack than previous methods with comparable dictionary sizes and images [27, 10], guessing 48-54% of passwords (compared to 0.9-9.1% previously) on two different images used in a long-term field study with a dictionary of about 2^{35} entries(which we call a 35-bit dictionary, abusing terminology somewhat). Furthermore, we were able to optimize this dictionary using Itti's model, producing dictionaries whose efficacy is comparable to human-seeded attacks [27]: using a 30.3-bit dictionary(i.e., having $2^{30.3}$ entries), the VA_1-$DIAG^{++}$ dictionary guessed 15.8% of passwords on one image, and in 31.4 bits it guessed 16.5% of passwords on a second image.

The remainder of this paper proceeds as follows. Section 2 discusses relevant background, including computational models of visual attention. We describe our purely automated attack generation methods in Section 3, results in Section 4, related work in Section 5, future work in Section 6, and conclusions in Section 7.

2 Background

We discuss computational models of visual attention in Section 2.1, distinguishable points in Section 2.2, and other background terminology in Section 2.3.

2.1 Computational Models of Visual Attention

We conjecture that a significant percentage of users will choose points that draw their attention as components of their click-based passwords, and thus that computational models of visual attention may help pick out more probable click-points. Computational models of bottom-up visual attention are normally defined by features of a digital image, such as intensity, color, and orientation [14, 13].

Computational models of top-down visual attention can be defined by training [21]. The difficulty of these models is that the top-down task must be pre-defined (e.g., find all people in the image), and then a corpus of images that are tagged with the areas containing the subject to find (e.g., people) must be used for training. Navalpakkam et al. [20] discuss an alternate method to create a top-down model, based on Guided Search [34], which weighs visual feature maps according to the top-down task. For example, with a task of locating a red object, a red-sensitive feature map would gain more weight, giving it a higher value in the resulting saliency map. In both cases, assumptions regarding what sort of objects people are looking for are required to create such a model.

In this work, we focus on bottom-up visual attention, using Itti et al.'s [14] computational model of visual attention. We use this particular model as it is quite well-known, and there is empirical evidence that it captures people's bottom-up visual attention [22]. The general idea behind this model is that areas of an image will be *salient* (or visually "stand out") when they differ from their surroundings.

We now explain Itti's model in further detail. Given an input image, it outputs a focus-of-attention scan-path to model the locations and the order in which a human might automatically and unconsciously attend them. It is composed of two stages: (stage 1) construction of a saliency map based on visual features, and (stage 2) the use of a winner-take-all neural network with inhibition of return to define a specific focus-of-attention scan-path, whose goal is to replicate the order in which a user would scan the image. Thorpe et al. [27] developed an automated attack for click-based graphical passwords that focused only on a variation of stage 1, ordering an attack dictionary based on the raw values of the resulting saliency map. The present paper uses the entire model including stage 2.

In stage 1, the saliency map is created by decomposing the original image into a set of 50 multi-level "feature maps", which extract spatial discontinuities based on color opponency (either red-green or blue-yellow), intensity, or orientation. Each level defines a different size of the center and its surround, in order to account for conspicuous locations of various sizes. All feature maps are then combined into a single saliency map.

In stage 2, a winner-take-all neural network detects the point of highest salience (as indicated by the intensity value of the saliency map), and draws the focus of attention towards this location. Once an area has been attended to, inhibition of return will prevent an area from being the focus again for a period of time. Together, the winner-take-all neural network with inhibition of return produces output in the form of spatio-temporal attentional scan-paths, which follow the order of decreasing saliency as defined by stage 1. Two different normalization types can be used with the model: *LocalMax* and *Iterative* (cf. Figure 1). In Iterative normalization, the neural network will find the next most salient area that has not been inhibited. In LocalMax normalization, the neural network will have a bias towards those areas that are closer to the previously attended location. Each normalization type produces a different scan-path; we study and compare the results of each as relates to our work.

2.2 Distinguishable Points

We hypothesize that users are more likely to choose *distinguishable points* as click-points. We define a *distinguishable point* as a point on a digital image that can be easily distinguished and relocated by a user. General ways this could be accomplished include: (1) by using *referencable* points on the image (e.g., a corner), and (2) by using *calculable* points that are based on other referencable parts of the image (e.g., object centers). In related work (see Section 5 for further detail), Thorpe et al. [27] used corner detection to find referencable points, and Dirik et al. [10] used centroids to find calculable points. Here, we use both approaches to define a *distinguishable points map* δ. We describe the details of each method below.

Figure 1: *pool* image with the first 7 items in the scan-path using (a) LocalMax normalization, and (b) Iterative normalization.

2.2.1 Corner Detection

A *corner* can be defined as the intersection of two edges, where an *edge* is defined by the points in a digital image where there are sharp changes in intensity [11]. A corner can also be defined as a point in whose local neighborhood there are two dominant and different edge directions [11].

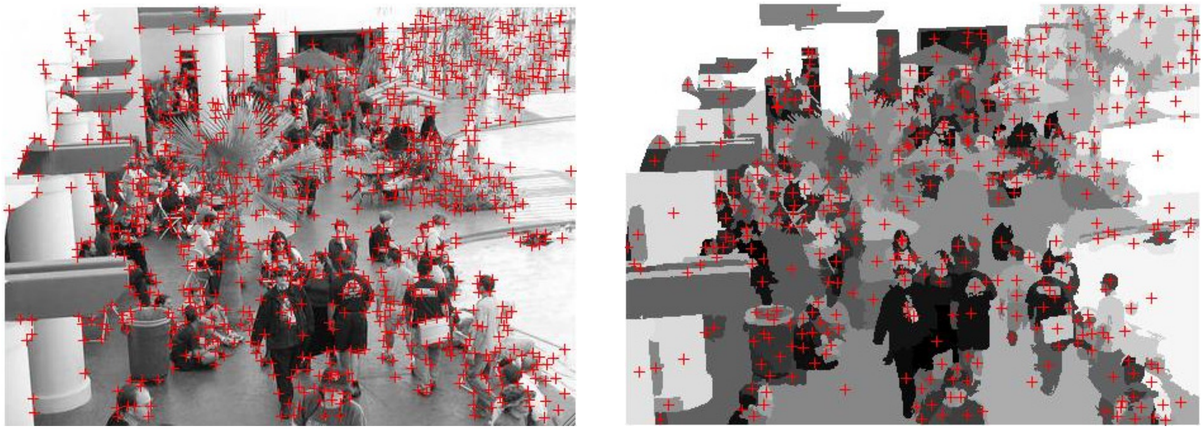
We use the harris algorithm [12] as implemented by Kovesi [17] for detecting corners. Harris corner detection first identifies the edges and then those edges are blurred to reduce the effect of any noise. Then an *energy map* is generated, based on the edges that contain local maxima and minima. A local maximum indicates the presence of a corner. We run harris corner detection with the parameters: $\sigma = 1$, $\theta = 1000$ and $r = 3$, where σ is the standard deviation of a smoothing Gaussian filter, θ is a threshold for the maximum number of corners, and r is an inhibition radius, measured in pixels around a detected corner.

Figure 2 shows the *pool* image where each detected corner is illustrated by a '+'. We also create a *binary corners map*, which is a specialized type of *binary map* (i.e., a one-to-one mapping from its pixels of value 0 or 1 to the pixels of the original image). In a binary corners map, when a pixel is a corner in the original image, its corresponding value is 1; otherwise it is 0.

2.2.2 Centroid Detection

To find the centers of objects, we first partition the digital image into segments using *image segmentation*. The goal of image segmentation is to change the representation of an image into something more meaningful and easier to analyze [25]. We use the mean-shift segmentation algorithm [7], which takes a feature (range) bandwidth, spatial bandwidth, and a minimum region area (in pixels) as input. We set these parameters to 7, 9, and 50 respectively, which we found empirically to provide an acceptable segmentation with the smallest resulting number of segments.

After segmentation, we calculate the center of each segment (centroid) by calculating the arithmetic mean of each coordinate of the segment's points. In other words, the center (X_S, Y_S) of



(a) Corner detection (b) Center detection

Figure 2: Corner and center detection output for *pool*.

segment S is calculated by $X_S = \frac{1}{n(S)} \sum_{i \in S} x_i$ and $Y_S = \frac{1}{n(S)} \sum_{i \in S} y_i$, where x_i and y_i are pixel coordinates, and $n(S)$ denotes the *total number* of pixels in segment S. The x coordinates of all points in S are involved in calculating X_S, not only those along the maximum width.

Figure 2(b) illustrates the resulting segments of the *pool* image with different shading. The center of each segment is denoted by a '+'. We also create a *centers map*, which is a binary map of the same size of the corresponding image where each pixel has the value 0 or 1. If a pixel is a center of a segment in the corresponding image, its value is 1; otherwise its value is 0. The *distinguishable points map* δ is a binary map that is the logical (inclusive) "or" of the binary centers map and binary corners map.

2.3 Other Terminology

We use the following additional terminology. Suppose that a user chooses a click-point c as part of her password. The *tolerable error* or *tolerance* t is the error allowed (in both vertical and horizontal directions) for a click-point entered on a subsequent login to be accepted as c. This defines a *tolerance region (T-region)* centered on c, which for an implementation using $t = 9$ pixels, is a 19×19 pixel square. A 19×19 T-region was used in the implementation for collecting the database [27] used herein for evaluating our results.

A *window cluster* is a square region of size $n \times n$ for some positive integer n. A *cluster* is a set of one or more points that lie within a window cluster. The center of a window cluster is representative of all the points within the window cluster. An *alphabet* is a set of distinct window centers.

3 Experimental Methodology

We pursue attacks that use click-order patterns and image processing methods for creating more efficient, ordered attack sub-dictionaries. We describe the specific click-order patterns we examine and their specification in Section 3.2. The image processing methods which we used for further optimization are described in Section 3.1, along with the *window clustering* algorithm we use to optimize the dictionary.

3.1 Visual Attention Based Image Processing Method

We used the Saliency Toolbox [30] implemented in Matlab. The weights of all feature maps used by the toolbox are set to one, to indicate that orientation, intensity and colors have the same level of importance. All the other settings of this toolbox are set to *default* except normalization type, which can be either *Iterative* or *LocalMax*. Since each of these two normalization types cause different spatio-temporal attentional scan-paths, we tested both in our experiments. For each, we examine two different styles of generating a dictionary for use in a guessing attack described below.

For each dictionary guessing style, we generate a map of candidate click-points by using our distinguishable points map δ as a bitmask to the resulting attracted regions of the image (i.e., using a logical "and"). We then refine this binary map of candidate points as follows.

Window Clustering Algorithm. We assume that an attacker's goal is to guess the largest number of passwords with the fewest guesses. After creating a set of points for a guessing dictionary (which might be used in passwords in any ordering of five clicks), many of them may be within the same tolerance region, and thus could be redundant (effectively guessing the same point). We devised a "clustering" method to normalize a set of points to a single value. The intuition is that given the system error tolerance, one point would be accepted as a correct entry for all others within its tolerance region. A previous clustering algorithm [27] centers each cluster on one of the original input points.

We introduce an alternative, *Window Clustering*, based on setting a window of fixed size (not necessarily the same size as the tolerance region) over the largest number of points it can cover. We then replace those candidate points inside the window with the geometric *center* of the window. Thus, the center of the cluster is not necessarily one of the original input points.

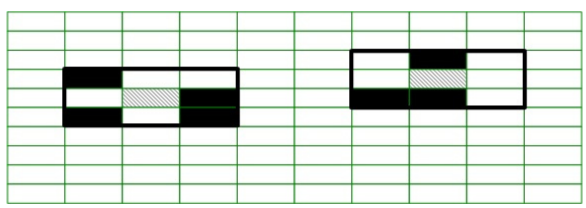


Figure 3: Window Clustering

Figure 3 shows an example set of candidate points with black squares, where each square represents a pixel. These 7 candidate points are covered with two 3×3 windows and will be represented by the centers of the two windows illustrated with grey squares.

Window Clustering is a greedy algorithm with a fixed window size. Starting with all candidate points, it finds the next position for the window that covers the maximum number of remaining points (ties are broken arbitrarily). It then stores the center of the window to represent the points in the window, and erases the corresponding points. It continues this process until no candidate points remain. In our experiments, the candidate points we use are the points with value 1 in S_b of Section 3.1.1 and B_i of Section 3.1.2. The window size is set to 19×19 in our experiments.

3.1.1 Guessing Style: Ordered by Scan-Path

The hypothesis here is that users may choose their password points from separate attracted regions, following the order of the focus-of-attention scan-path. The ordered dictionary described in this subsection is designed to test this hypothesis.

Using the visual attention tool, we generate S, a set of binary maps, where each binary map is generated in a single step of the scan-path. $S = \{A_1, A_2, ..., A_n\}$ where A_i denotes the generated binary map in step i and n is the total number of steps. Pixels of a binary map A_i have value 1 if they belong to the attracted region of step i, otherwise 0. $S_b = \{A_1 \wedge \delta, A_2 \wedge \delta, ..., A_n \wedge \delta\}$, and then the window clustering algorithm is run separately on each element in S_b to create S_c. S_c contains n sets of candidate points, each set containing the cluster centers produced from running the clustering algorithm on the corresponding element of S_b . To create each entry of the dictionary, we choose all sets of five elements of S_c and then order these elements by increasing index. Finally, we choose one point from each of these five elements, while retaining the element ordering, to put in the dictionary. Thus each dictionary entry is a five-point graphical password, where each point belongs to an element of S_c, and the five points (each belonging to a distinct scan-path element) are ordered according to the order of the 5 elements in the scan-path.

3.1.2 Guessing Style: Unordered Incremental

Here the hypothesis is that users may choose their click-points based on points that fall along the focus-of-attention scan-path, but not necessarily in the order of the scan-path. The unordered incremental (UI) dictionary is designed to test this hypothesis. We call the UI dictionary with *LocalMax* normalization VA_1, and with *Iterative* normalization VA_2.

Using the visual attention tool, we generate a set of binary maps S', where the i^{th} binary map is generated from all of the steps until step i in a scan-path: $S' = \{B_1, B_2, ..., B_n\}$ where $B_i = A_1 \vee A_2 \vee ... \vee A_i$. In other words, $B_i = B_{i-1} \vee A_i$ and $B_1 = A_1$. Next, we calculate $\{C_1, C_2, ..., C_n\}$ where $C_i = B_i \wedge \delta$, the intersection (logical "and") of each element B_i with δ, and run the window clustering algorithm on each C_i to produce D_i, the resulting set of cluster centers (which are pixel locations on the image). A sub-dictionary P_i is all 5-permutations of the elements of D_i, and so the final dictionary $P = \{P_1, P_2, \ldots, P_n\}$ is ordered by the number of steps in the scan-path that are considered, e.g., all passwords from P_2 are only guessed after those in P_1 are exhausted.

3.2 Click-order Patterns and Relaxation (Lazy, Super-Lazy)

We examine two click-order patterns alone (DIAG and LINE), and with what we call *lazy* and *super-lazy* variations that relax the definition of the patterns alone. The two click-order patterns are (1) DIAG and (2) LINE, a subset of DIAG that we introduce herein. DIAG includes any sequence of 5 click-points that follow *both* a consistent vertical and horizontal direction (e.g., straight lines in any direction, most arcs, and step-patterns). LINE includes any sequence of 5 click-points that follow either a vertical or horizontal line. More specifically, DIAG = $LR\_TB \cup LR\_BT \cup RL\_TB \cup RL\_BT$. Thus DIAG is the union of four sets of passwords. In the descriptive name of each set, the first two letters show the horizontal direction and the last two are related to vertical direction. LR and RL denote left-to-right and right-to-left respectively; TB and BT denote top-to-bottom and bottom-to-top respectively. Each of the four sets in DIAG consists of all 5-point passwords whose successive pairs of points $(x_i, y_i), (x_{i+1}, y_{i+1})$ satisfy the specified constraints. Our convention is that the positive y axis extends downward from the top-left pixel of the image.

LR_BT: $(x_i \leq x_{i+1} + \tau) \ \wedge \ (y_i \geq y_{i+1} - \tau)$
RL_BT: $(x_i \geq x_{i+1} - \tau) \ \wedge \ (y_i \geq y_{i+1} - \tau)$
LR_TB: $(x_i \leq x_{i+1} + \tau) \ \wedge \ (y_i \leq y_{i+1} + \tau)$
RL_TB: $(x_i \geq x_{i+1} - \tau) \ \wedge \ (y_i \leq y_{i+1} + \tau)$

Similarly, LINE = $LR \cup RL \cup BT \cup TB$, where the four sets in LINE consist of all passwords whose successive pairs of points satisfy analogous constraints as follows.
LR: $(x_i \leq x_{i+1} + \tau) \wedge (|y_i - y_{i+1}| \leq \tau)$
RL: $(x_i \geq x_{i+1} - \tau) \wedge (|y_i - y_{i+1}| \leq \tau)$
BT: $(y_i \geq y_{i+1} + \tau) \wedge (|x_i - x_{i+1}| \leq \tau)$
TB: $(y_i \leq y_{i+1} + \tau) \wedge (|x_i - x_{i+1}| \leq \tau)$

For both LINE and DIAG, the allowance τ serves the purpose of relaxing the pattern, since although the user might be inclined to select points along a line, the elements of that line may be influenced by which click-points the user otherwise prefers. If the image has many straight-line structures, it would seem reasonable to expect that users would choose straighter lines, but in the absence of linear structures in the image, the lines may be more of an approximation. To this end, we introduce two variations on both DIAG and LINE, that relax τ in their above definitions to allow "lazier" lines: "lazy", which uses $\tau = 19$ and "super-lazy", which uses $\tau = 28$. In the normal relaxation case, we use $\tau = 9$ (i.e., equal to the system error tolerance).

We denote a dictionary using a lazy or super-lazy τ with superscripts + and ++ respectively.

3.3 Alphabet for Evaluating Click-Order Patterns: α

For evaluating (and attacking using) our click-order patterns alone, we define an initial set of points for use in generating dictionary passwords. To guess all passwords with a given click-order pattern, we define the character set such that it covers the entire image. We ensure that none of the initial points have overlapping T-regions (recall Section 2.3); i.e., this character set is the full, non-overlapping alphabet. The set is obtained by placing a grid of 19×19 windows (i.e, the same size as the T-region) over the image and adding only the centers of each window. We call the resulting set of non-overlapping window centers α. Note that the T-region used in creating α is only dependent upon the system error tolerance, and is independent of the τ used in our different relaxation modes.

4 Experimental Results

To allow meaningful comparison, we tested our methods by trying to guess users' graphical passwords, using a previous PassPoints user study password database which we summarize for convenience in Section 4.1.

In addition to the guessing styles described in Section 3.1, we also examine the success of guessing using only the distinguishable points (i.e., corners and centers), to determine whether our hypothesis (recall Section 2.2) that users are more likely to choose distinguishable points as click-points is reasonable. As expected, our results are consistent with this hypothesis, as noted in Section 4.4.

4.1 Review of User Study

The field study used to allow comparison [27, 5] was 7-weeks or longer (depending on the user), involving 223 user accounts on a web-based implementation of PassPoints to gain access to course notes, assignment solutions, and tutorials. We focus on the field study rather than the related lab study for increased confidence that the passwords we are studying have some degree of long-term memorability. Participants were from three undergraduate classes: two first year courses for computer science students, and a first year course for non-computer science students enrolled in a science degree. Participants used one of two background images, *pool* or *cars* (see Figure 4), carefully preselected to be representative of highly detailed usable images at 451×331 pixels.

(a) *cars* (originally from [3]).

(b) *pool* (originally from [32, 31]).

Figure 4: Images used in lab study.

Passwords had 5 click-points, no two within $t = 9$ pixels of another (vertically and horizontally). Consistent with the previous study, we used only the final passwords exercised by each user (and recalled at least once). These 223 user accounts mapped to 189 distinct users (34 users were in two classes; all but one of them were assigned a different image for each account). Overall, 114 user accounts used *pool* and 109 used *cars* as a background image.

4.2 Ordered Scan-Path Results

We tested the hypothesis that users choose click-points in the order of their focus-of-attention scan-path, using the method of Section 3.1.1. We found that this method did not guess any passwords correctly. This indicates that users do *not* choose click-points along a partial ordering of the scan-path elements the model produces (under the default settings we used).

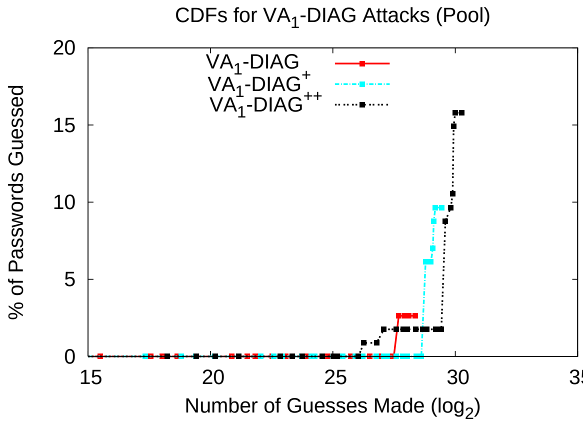
We see two possible reasons for this result: (1) users do not choose their click-points entirely based on bottom-up visual attention; or (2) the model of visual attention as used does not accurately capture bottom-up visual attention. Our results in Section 4.3 suggest that bottom-up visual attention, according to this computational model, might be a partial factor in user choice.

4.3 Main Results: Heuristic and Image Processing Methods

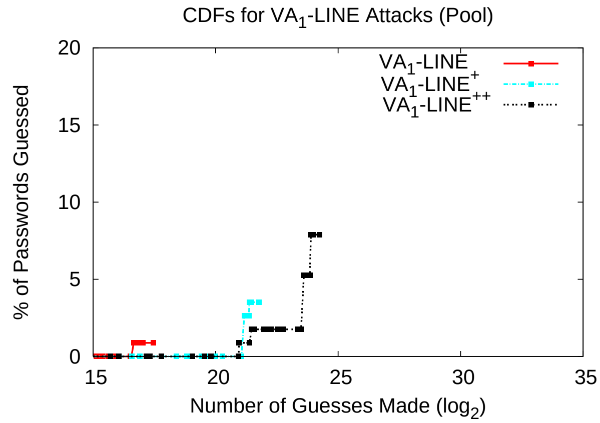
We tested our hypothesis that users choose click points based on bottom-up visual attention using VA_1, the incremental attack of Section 3.1.2 with *LocalMax* normalization. We discuss VA_2 (*Iterative* normalization) further below.

The VA_1 attack strategy uses the scan-path to prioritize the dictionary entries; all 5-permutations of points using only the first scan-path element are considered before all 5-permutations using both the first and second elements, etc. This method is applied in combination with the various click-order patterns of Section 3.2, combining VA_1 with each of DIAG and LINE click-order patterns at all laziness modes. The cumulative distribution function (CDF) of our results (until each dictionary is exhausted) are provided in Figure 5.

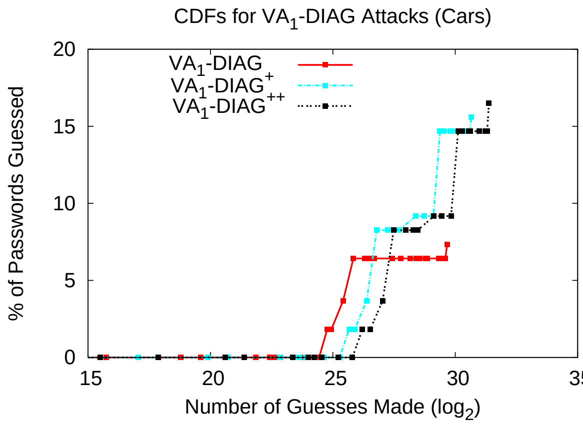
$\alpha DIAG$ and $\alpha LINE$ are dictionaries generated using the alphabet α (recall Section 3.3) and the click-order patterns DIAG and LINE respectively. These dictionaries are generated to examine



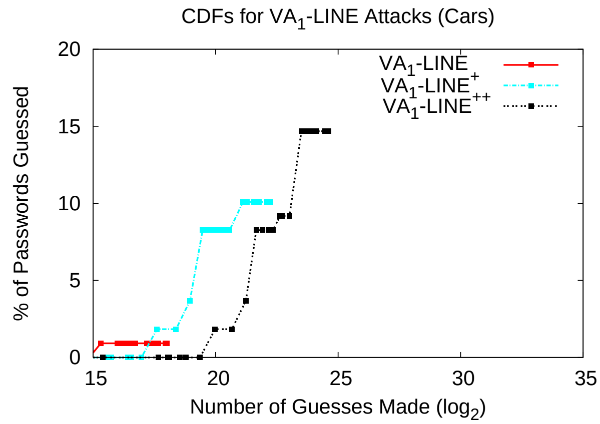
(a) VA_1-DIAG CDFs for *pool*.



(b) VA_1-LINE CDFs for *pool*.



(c) VA_1-DIAG CDFs for *cars*.



(d)VA_1-LINE CDFs for *cars*.

Figure 5: CDFs for different attacks with LocalMax normalization (i.e., VA_1).

Dictionary	*pool*		*cars*	
	bit-size	% guessed	bit-size	% guessed
$\alpha DIAG$	33.0	21.1%	33.0	27.5%
$\alpha DIAG^+$, $\alpha DIAG^{++}$	34.7	48.2%	34.7	54.1%
$\alpha LINE$	20.1	3.5%	20.1	22.0%
$\alpha LINE^+$, $\alpha LINE^{++}$	27.7	23.7%	27.7	52.3%

Table 1: Results for click-order pattern heuristics DIAG and LINE with different laziness modes.

the efficacy of using click-order pattern heuristics alone (i.e., without additional image processing methods as in Figure 5). Table 1 presents our results for DIAG and LINE. Note that the values in Table 1 for the dictionaries marked '+' and '++' are the same because the T-regions are non-overlapping, and the difference between $\tau = 19$ and 28 is not sufficient to increase the number of included T-regions.

Many interesting points emerge from the graphs in Figure 5 and values in Table 1. The most notable in terms of success rate is the $\alpha DIAG^{+}$ dictionary of Table 1, which guesses 48% of passwords for *pool* and 54% of passwords for *cars* with dictionaries of less than 35 bits (i.e., 2^{35} entries). Previous purely automated attacks [27] against these same images on the same password database, with a 35 bit dictionary, guessed 9.1% of passwords on *cars* and 0.9% of passwords on *pool*. Similarly, $\alpha LINE^{+}$ of Table 1 guesses 23.7% of passwords for *pool*, and 52.3% of passwords for *cars* in 27.7 bits. It is interesting that when using $\alpha LINE^{+}$ the percentage of passwords guessed for *cars* only drops by about 2%, while the accuracy for *pool* only drops by about 24% (about half as many passwords are guessed), despite a dictionary bitsize reduction of 7 bits. This implies that $\alpha LINE^{+}$ is the most efficient (in terms of accuracy and bit-size) click-order pattern studied to date. It is not surprising that the LINE variations work better on the *cars*, given the number of straight line structures in the image and the orientation of the cars in the parking lot. It seems more surprising to see LINE variations working as well as they do for *pool*.

The VA_1 optimization of Figure 5 does appear to create a more efficient dictionary: for the $DIAG^{++}$ variation, we are able to guess 15.8 -16.5% in dictionaries of less than 31.3 bits, and for the $LINE^{++}$ variation, we are able to guess 7.9-14.7% of passwords in less than 24.6 bits. The relative "efficiency" of the dictionaries, however, cannot be extended in their present form to guess a larger percentage of passwords, because the full dictionaries are exhausted.

We also tried our attack using VA_2. We found that it did not perform very well, indicating that what might seem like a small strategy change can have a dramatic effect. It is interesting to note that the only difference between VA_1 and VA_2 is the local vs. global bias in the neural network algorithm.

4.4 Influence of Distinguishable Points

Here we explore in a theoretical attack, the efficacy of using only the distinguishable points map δ for guessing the passwords in our database. This attack only guesses passwords with all five points in δ. We use a smaller T-region of 9×9 for evaluating each guess; as an aside, this was found in usability studies [5] to better approximate to how accurately users can enter their click-points. The full password space when using a 9×9 T-region is 55 bits, compared to 43 bits for the rest of this paper.

On average, we were able to guess 47% of passwords from the field study database [27, 5] using only the raw δ; however, the average dictionary size was 52.9 bits. These results indicate that on average we expect a full exhaustive (naive) dictionary based on points in δ to contain approximately half of user passwords but requiring a dictionary size of one quarter of the password space. This is consistent with the natural expectation that distinguishable points are somewhat more probable than other points for most images. We conclude that distinguishable points provide a starting point for a fully automated guessing attack, but by themselves offer only a very modest advantage (essentially a single bit in the attack space).

4.5 Discussion

Our overall results indicate that although essentially no users choose their click-points in the strict scan-path order of Itti's model of visual attention, when all permutations of points in the scan-path are considered, it models a meaningful percentage (from an attacker viewpoint) of user passwords. This raises interesting questions regarding how visual attention relates to user choice in graphical passwords. Our results would be consistent with the hypothesis that bottom-up visual attention is a factor in user choice for some users (and/or for some images), but not necessarily for all users.

5 Related Work

Many different types of graphical passwords have been proposed to date (see surveys [26, 19]). Here we focus on click-based graphical password schemes and other work on modeling user choice in graphical passwords.

Click-based graphical password schemes require a user to click on a set of points on one or more presented background images. In Blonder's proposal [2] users must click on a set of predefined tap regions. In *V-go*, by PassLogix [23], users must click on predefined objects in the picture in a specific sequence. In the Jansen et al. [15] variation for PDAs, users click an ordered sequence of visible grid squares imposed on a background image; the squares are intended to help the user repeat their click-points in subsequent logins.

PassPoints [32, 31, 33] allows users to click a sequence of five points anywhere on an image while allowing a degree of error tolerance using robust discretization [1]. Various studies have shown that PassPoints has acceptable usability [33, 32, 31, 5]. *visKey*, a commercial system for the Pocket PC, appears similar but allows the user to choose the number of click-points and set the error tolerance. In the Persuasive Cued Click-Points (PCCP) [4, 6] variation, a user clicks on a single point on each of five images, guided partially by a randomly placed viewport; each image displayed (after the first) is dependent on the previous click-point.

Two previous studies have examined the security of PassPoints-style graphical passwords. Dirik et al. [10] examine the efficacy of an automated tool for guessing PassPoints passwords. Their method, which does not draw on a standard computational model of visual attention, uses centroids of segments as guesses (but no corners). It was tested against a database of single-session user choices for two images. For the image with a reasonable level of detail, their method guessed 8% of passwords with a 32-bit dictionary compared to a 40-bit full space. As previously discussed, Thorpe et al. [27] examine both an automated method (based on stage 1 of Itti et al.'s [14] model of visual attention; recall Section 2.1), and a human-seeded method (which uses click-point data from a set of users' password choices).

User choice has been successfully modeled for other types of graphical passwords. Davis et al. [9] modeled user choice for Faces and Story recognition-based graphical passwords by training a dictionary using a large password database. Van Oorschot et al. [28] model user choice in "Draw-A-Secret" pure-recall graphical passwords [16] based on cognitive studies.

6 Future Work

It would be interesting to see how well top-down models using a variety of plausible assumptions would work. For example, might the first point be chosen according to bottom-up visual attention, and then the rest chosen in a top-down manner such that they are somehow similar to the first? Alternately, might the entire process be top-down, based on whether the user can find five objects that are similar in some way? Such a top-down theory would be substantially more difficult to model an attack on, but if possible to implement, its results would offer interesting insight.

The distinguishable points map δ could be enhanced in several ways, to refine the attacks presented herein. We expect the dictionaries could be further improved by incorporating other types of *calculable* points, such as north, south, east, and west on circles and squares. Also, changing the parameters to the algorithms we use to identify distinguishable points might provide better results, or changing the parameters for the visual attention model. It would also be interesting to explore whether settings can be optimized for a wide range of images, or if optimal settings are highly-image specific. Regarding our image-independent attacks which rely only on generic patterns and window clustering, exploration of other patterns may prove fruitful.

Finally, image blurring techniques (e.g., similar to CAPTCHAs [29]) may offer some protection against at least the visual attention, center detection, and corner detection portion of automated attacks. It would be interesting to determine whether such a technique can alter the image in a way that users could still choose passwords, but corner and centroid detection techniques would not succeed, even after attempts to restore the original image.

7 Concluding Remarks

We provide the best automated attack against PassPoints-style graphical passwords to date (to our knowledge). Click-order patterns, DIAG and LINE, combined with our laziest relaxation rule, yielded highly effective dictionaries. We were able to further reduce the dictionary size while retaining some accuracy by using Itti et al.'s [14] computational model of bottom-up visual attention. Our results are a significant improvement on previous work for purely automated guessing PassPoints-style graphical passwords. Using one 35-bit dictionary variation, $\alpha DIAG^{++}$, we were able to guess over 48% of user passwords for each of two images, whereas previous work was able to guess 0.9-9.1% [27] for the same images and user study password database and 8% [10] for a single (comparable) image.

Although these lazy click-order dictionary sizes are not as small as previous dictionaries used for human-seeded attacks, when we combine them with a model of visual attention [14], our results are comparable to the human-seeded results for *cars*. In a dictionary of 30.3 bits or $2^{30.3}$ entries, the VA_1-$DIAG^{++}$ dictionary guessed 15.8% of passwords on *pool*, and in 31.4 bits it guessed 16.5% of passwords on *cars*. For *cars*, the basic human-seeded attack [27] guessed 20% of passwords with a 33-bit dictionary. This suggests that automated attacks provide an effective alternative to a human-seeded attack against PassPoints-style graphical passwords. Furthermore, they allow continuation of an attack through using click-order patterns without prioritization by some other means, guessing more passwords overall than human-seeded methods. Finally, they are arguably much easier for an attacker to launch (removing the requirement of humans to index the images), especially if large image datasets are used. We emphasize that the attack dictionaries used for Table 1 including the 35-bit dictionary in the previous paragraph, do not rely on visual attention techniques or any image-specific pre-computation, implying that the actual dictionaries are the same for all images, though the attack results (i.e., their effectiveness) are image-dependent and of course depend also on the actual passwords chosen by any users in question.

We evaluated different guessing styles using Itti's model and found that using the first 30 steps of the model's scan-path output, none of the user passwords in the database followed along the scan-path order (i.e., 5 ordered points within the 30-element scan-path ordering), but a number of passwords were composed of points within other orderings of the scan-path elements. If we assume that Itti's model using the default settings is an accurate representation of bottom-up visual attention, these results are consistent with bottom-up visual attention being one part of a broader criteria for selecting click-points. Alternately, these click-points might be chosen according to some other phenomenon that happens to have a non-null intersection with this model of bottom-up visual attention.

Using *Iterative* normalization, our attack based on the visual attention model did not perform very well, indicating that the bias of proximity (local saliency vs. global saliency) can have a dramatic effect. Our success using *LocalMax* normalization suggests that users are more likely to choose successive points that are closer to one another than on the other side of the image. The difference in results suggests that the success of the *LocalMax* attack is not a chance effect, but rather it actually locates the parts of the image that users are more inclined to choose as click-points. Even better results may be possible through other (unknown) neural network strategies.

Future work includes determining whether there are top-down models that may provide an accurate representation of user choice, and whether image processing measures could effectively filter out images that are more prone to structure-based click-order patterns like those exploited in our attacks.

References

[1] J.C. Birget, D. Hong, and N. Memon. Robust Discretization, with an Application to Graphical Passwords. *IEEE Transactions on Information Forensics and Security*, 1:395–399, 2006.

[2] G. Blonder. Graphical Passwords. United States Patent 5559961, 1996.

[3] Ian Britton. `http://www.freefoto.com`, site accessed Feb. 2, 2007.

[4] S. Chiasson, A. Forget, P.C. van Oorschot, and R. Biddle. Influencing Users Towards Better Passwords: Persuasive Cued Click-Points. In *HCI*, 2008.

[5] S. Chiasson, P.C. van Oorschot, and R. Biddle. A Second Look at the Usability of Click-Based Graphical Passwords. In *3rd Symposium on Usable Privacy and Security (SOUPS)*, 2007.

[6] S. Chiasson, P.C. van Oorschot, and R. Biddle. Graphical Password Authentication Using Cued Click Points. In *European Symposium on Research in Computer Security (ESORICS)*, 2007.

[7] D. Comaniciu and P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[8] N. Cowan. The Magical Number 4 in Short-Term Memory: A Reconsideration of Mental Storage Capacity. *Behavioral and Brain Sciences*, 24:87–185, 2000.

[9] D. Davis, F. Monrose, and M.K. Reiter. On User Choice in Graphical Password Schemes. In *13th USENIX Security Symposium*, 2004.

[10] Ahmet Emir Dirik, Nasir Memon, and Jean-Camille Birget. Modeling User Choice in the PassPoints Graphical Password Scheme. In *3rd Symposium on Usable Privacy and Security (SOUPS)*, pages 20–28, 2007.

[11] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[12] C. Harris and M. Stephens. A Combined Corner and Edge Detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[13] L. Itti and C. Koch. Computational Modeling of Visual Attention. *Nature Reviews Neuroscience*, 2(3):194–203, Mar 2001.

[14] L. Itti, C. Koch, and E. Niebur. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, Nov 1998.

[15] W. Jansen, S. Gavrilla, V. Korolev, R. Ayers, and Swanstrom R. Picture Password: A Visual Login Technique for Mobile Devices. NIST Report - NISTIR7030, 2003.

[16] I. Jermyn, A. Mayer, F. Monrose, M. Reiter, and A. Rubin. The Design and Analysis of Graphical Passwords. In *8th USENIX Security Symposium*, 1999.

[17] P. D. Kovesi. MATLAB and Octave Functions for Computer Vision and Image Processing. School of Computer Science & Software Engineering, The University of Western Australia. `http://www.csse.uwa.edu.au/~pk/research/matlabfns/`.

[18] S. Madigan. Picture Memory. In John C. Yuille, editor, *Imagery, Memory and Cognition*, pages 65–89. Lawrence Erlbaum Associates Inc., N.J., U.S.A., 1983.

[19] F. Monrose and M. K. Reiter. Graphical Passwords. In L. Cranor and S. Garfinkel, editors, *Security and Usability*, chapter 9, pages 147–164. O'Reilly, 2005.

[20] Vidhya Navalpakkam and Laurent Itti. Modeling the Influence of Task on Attention. *Vison Research*, 45:205–231, 2005.

[21] A. Oliva, A. Torralba, M. Castelhano, and J. Henderson. Top Down Control of Visual Attention in Object Detection. *Journal of Vision*, 3(9):253–256, 2003.

[22] N. Ouerhani, R. von Wartburg, H. Hugli, and R. Muri. Empirical Validation of the Saliency-based Model of Visual Attention. *Electronic Letters on Computer Vision and Image Analysis*, 3(1):13–24, 2004.

[23] Passlogix. `http://www.passlogix.com`, site accessed Feb. 2, 2007.

[24] SFR IT-Engineering. The Grafical Login Solution For your Pocket PC - visKey. `http://www.sfr-software.de/cms/EN/pocketpc/viskey/index.html`, site accessed March 18, 2007.

[25] George Stockman and Linda G. Shapiro. *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[26] Xiaoyuan Suo, Ying Zhu, and G. Scott Owen. Graphical Passwords: A Survey. In *21st Annual Computer Security Applications Conference (ACSAC)*, 2005.

[27] Julie Thorpe and P. C. van Oorschot. Human-Seeded Attacks and Exploiting Hot-Spots in Graphical Passwords. In *USENIX '07: Proceedings of the 16th USENIX Security Symposium*, pages 103–118, 2007.

[28] P.C. van Oorschot and J. Thorpe. On Predictive Models and User-Drawn Graphical Passwords. *ACM Transactions on Information and System Security*, 10(4):1–33, November 2007.

[29] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems for Security. In *Advances in Cryptology - EUROCRPYT 2003: International Conference on the Theory and Applications of Cryptographic Techniques*, volume 2656/2003, 2003.

[30] Dirk Walther and Christof Koch. 2006 Special Issue: Modeling Attention to Salient Proto-objects. *Neural Network*, 19(9):1395–1407, 2006.

[31] Susan Wiedenbeck, Jim Waters, Jean-Camille Birget, Alex Brodskiy, and Nasir Memon. Authentication Using Graphical Passwords: Basic Results. In *Human-Computer Interaction International (HCII)*, 2005.

[32] Susan Wiedenbeck, Jim Waters, Jean-Camille Birget, Alex Brodskiy, and Nasir Memon. Authentication Using Graphical Passwords: Effects of Tolerance and Image Choice. In *1st Symposium on Usable Privacy and Security (SOUPS)*, pages 1–12, 2005.

[33] Susan Wiedenbeck, Jim Waters, Jean-Camille Birget, Alex Brodskiy, and Nasir Memon. PassPoints: Design and Longitudinal Evaluation of a Graphical Password System. *Int. J. Hum.-Comput. Stud.*, 63(1-2):102–127, 2005.

[34] J.M. Wolfe. Guided Search 2.0: A Revised Model of Visual Search. *Psychonomic Bulletin and Review*, 1(2):202–238, 1994.