Security Implications of Password Discretization for Click-based Graphical Passwords*

Bin B. Zhu[1], Dongchen Wei[2], Maowei Yang[3], Jeff Yan[4]
[1] Microsoft Research Asia, Beijing, China
[2,3] Sichuan University, Chengdu, Sichuan, China
[4] Newcastle University, United Kingdom
[1]binzhu@microsoft.com, [2]v-dowe@microsoft.com, [3]djyangmaowei@gmail.com, [4]jeff.yan@ncl.ac.uk

ABSTRACT
Discretization is a standard technique used in click-based graphical passwords for tolerating input variance so that approximately correct passwords are accepted by the system. In this paper, we show for the first time that two representative discretization schemes leak a significant amount of password information, undermining the security of such graphical passwords. We exploit such information leakage for successful dictionary attacks on Persuasive Cued Click Points (PCCP), which is to date the most secure click-based graphical password scheme and was considered to be *resistant* to such attacks. In our experiments, our purely automated attack successfully guessed 69.2% of the passwords when Centered Discretization was used to implement PCCP, and 39.4% of the passwords when Robust Discretization was used. Each attack dictionary we used was of approximately 2^{35} entries, whereas the full password space was of 2^{43} entries. For Centered Discretization, our attack still successfully guessed 50% of the passwords when the dictionary size was reduced to approximately 2^{30} entries. Our attack is also applicable to common implementations of other click-based graphical password systems such as PassPoints and Cued Click Points – both have been extensively studied in the research communities.

Categories and Subject Descriptors
K.6.5 [**Management of Computing and Information Systems**]: Security and Protection – *Authentication, unauthorized access.*
K.4.4 [**Computers and Society**]: Electronic Commerce – *Security.*

General Terms
Security, Experimentation.

Keywords
Graphical passwords, dictionary attack, discretization, authentication.

1. INTRODUCTION
Passwords have been widely used to authenticate users to remote servers in Web and other applications. Text passwords have been used for a long time. Graphical passwords, introduced by Blonder [1] in 1996, are an alternative to text passwords. In a graphical password, a user interacts with one or more images to create or enter a password. Graphical passwords are intended to capitalize on the promise of better memorability and improved security against guessing attacks. Graphical passwords are particularly suitable for keyboardless devices such as iPads and iPhones whereon inputting a text password is cumbersome. For example, Windows 8 recently released by Microsoft supports graphical password logon. With increasingly popularity of smart phones and slate computers, we expect to see a wider deployment of graphical passwords in Web applications.

Among a large variety of graphical password proposals, click-based graphical passwords have attracted the most attention in both the Human-Computer Interaction (HCI) and security communities. A click-based graphical password consists of a sequence of click-points on one or more images. To log in, a user clicks the same points of her password, in the correct order, on the same image(s). PassPoints [2][3] is a representative click-based graphical password scheme, wherein a password consists of a sequence of points anywhere on an image. Later studies [5]-[11] indicate that PassPoints is vulnerable to dictionary attacks which exploit image hotspots [5][6] (i.e. spots that are more likely to be selected as click-points across users) and patterns of click-points [7]. Purely automated attacks [10][11] detect corner points and centroid points as hotspots, and apply heuristics to select a set of combinations of the detected hotspots to form dictionaries of guessed passwords. The attacks on two representative images used in PassPoints guessed 7-16% of the passwords for dictionaries each with approximately 2^{26} entries, and 48-54% of the passwords for dictionaries each with approximately 2^{35} entries, whereas the full password space contained 2^{43} entries.

Lessons of hotspot-based dictionary attacks on PassPoints led to the design of two improved click-based graphical password schemes, Cued Click Points (CCP) [12] and Persuasive Cued Click Points (PCCP) [13][14]. CCP is a variation of PassPoints with improved security, and PCCP improves the security further. PCCP has been considered robust to all the reported hotspot-based dictionary attacks.

Click-based graphical password schemes such as PassPoints, CCP, and PCCP allow arbitrary click-points in a password. Due to inevitable click inaccuracy, a predefined tolerance distance is used in these schemes that a click is verified correct if it falls in the tolerance region which has a distance to the originally chosen click-point equal to or less than the tolerance distance. This would work well if the password is stored in the clear in the system. For the sake of security, a practical system typically does not store

WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1 /13/05.

* Corresponding author: Bin B. Zhu (binzhu@microsoft.com or binzhu@ieee.org). This work was done when Dongchen Wei and Maowei Yang were interns at Microsoft Research Asia.

passwords in the clear. Instead, a password is cryptographically hashed and the hash value is stored in the system. It is impossible for such a system to calculate the distance of a click to the corresponding click-point in the password since a single bit change in the input would result in a completely different hash value. Therefore the system has no way to check if the click is within the tolerance region of the click-point or not. This problem is identified in [15]. A solution is discretization of click-points using grids so that all tolerable clicks of a click-point are inside a single grid cell. A discretization scheme should guarantee a minimum tolerance range of r pixels, i.e., being r-safe: if a clicked point is within r pixels from the desirable point, the same discretization point would always be produced.

Several password discretization schemes have been proposed. Robust Discretization [15] uses three offset grids of grid-square size $6r \times 6r$ to guarantee that for every point in the image, there exists at least one grid whereby the point is r-safe. Centered Discretization [16] determines the grid for a point such that the point is the center of a grid-square of the grid. Its grid-square size is $2r \times 2r$ to maintain that the point is r-safe. Centered Discretization produces a smaller grid-square than Robust Discretization without impacting the usability of the system [16]. Optimal Discretization [17] is the same as Centered Discretization when offset is used, and suffers from the edge problem of discretization (i.e., a small perturbation may result in a wrong grid-square when the click-point is near a grid line) when no offset is used. The edge problem is what the other discretization schemes tried to avoid in their designs. As a consequence, multiple trials of possible grid-squares due to acceptable click-variation are used during authentication to address the edge problem of discretization when offset is not used in Optimal Discretization. Image-dependent discretization is proposed in [18], wherein image features are analyzed and Voronoi polygon tiling is produced that likely click-points are centered within the polygons.

In all these discretization schemes, grid information used for a password is stored in the system in plaintext so that the same grids are used to discretize clicks in authenticating a user. This information may be accessible to adversaries. A natural question arises: what is the security implication of this additional information about a password? It is believed that the additional information of the discretization grids does not lead to weaker security [16]. It has remained for years an open problem whether it is possible for adversaries to exploit such information to their advantage [14][16][19].

In this paper, we for the first time address this open question, which concerns an important aspect of applying click-based graphical password schemes including PassPoints, CCP and PCCP in real applications. Our security analysis on two representative discretization schemes, Robust Discretization and Centered Discretization, indicates that discretization does have significant security implications: it leaks information about password click-points, and thus leads to weaker security. The leaked information can be exploited to mount successful dictionary attacks on click-based graphical password schemes such as PCCP which are otherwise considered robust to the dictionary attacks. Our experimental studies on PCCP show that our purely automatic dictionary attacks using dictionaries each with approximately 2^{35} entries guessed 69.2% of the passwords when Centered Discretization was used, and 39.4% of the passwords when Robust Discretization was used, whereas the full password space was of 2^{43} entries. In addition, for Centered Discretization, our attack still successfully guessed 50% of the passwords when the attack dictionary size was reduced to approximately 2^{30}. Our work sheds light on the future design of secure yet practical discretization schemes.

The remaining paper is organized as follows. Related work is reviewed in Section 2. Technical details of discretization schemes are described in Section 3. We present our security analysis of discretization schemes in Section 4. Our dictionary attacks on PCCP using both Robust Discretization and Centered Discretization are described in Section 5. Discussions are presented in Section 6. The paper concludes in Section 7.

2. RELATED WORK

We briefly review click-based graphical passwords and their dictionary attacks in this section. Details on discretization schemes are provided in Section 3.

2.1 Click-based Graphical Passwords

Since the introduction of the first graphical password scheme by Blonder [1] in 1996, graphical passwords have become an active research topic, and a large number of graphical password schemes have been proposed. Among these schemes, click-based graphical password schemes have attracted the most attention in both HCI and security research communities. They will be briefly described here. For more detailed information of click-based graphical password schemes as well as other types of graphical password schemes, readers are referred to a recent comprehensive review of graphical password schemes [19].

In Blonder's scheme [1], users click on a set of predefined tap regions. Jansen et al. [20] proposed a variation which requires users to click an ordered sequence of visible squares imposed on a background image. The squares are used to help users repeat click-points in subsequent logins. In V-go [21], users click on a sequence of predefined objects in the image. PassPoints [2][3] is the first click-based graphical scheme that allows a user to click anywhere on an image in creating a password. It requires a user to click a sequence of l points anywhere on an image. PassPoints has studied extensively. Studies [2]-[4] indicate that $l = 5$ leads to promising usability. Thus this setting has been widely adopted in the literature. Cued Click Points (CCP) [12] is a variation of PassPoints with improved security. In CCP, a sequence of images is used in entering a password, one click per image, with the next image selected by a deterministic function. The security is further improved with Persuasive Cued Click Points (PCCP) [13][14], which requires a user to select each click-point inside a randomly positioned viewport in creating a password, resulting in more randomly distributed click-points in a password.

2.2 Dictionary Attacks on Click-based Graphical Passwords

Effective dictionary attacks have been conducted on PassPoints-style graphical password schemes by exploiting two weaknesses in human selection of a password: hotspots and patterns. Hotspots [5][6] are spots more likely selected as click-points across users, and patterns [7] are likely click orders and location relationships of click-points in users' passwords. Both are related to predictable preferences in human created passwords.

Image processing techniques have been used to locate hotspots in an image to enable automatic guessing attacks [5][10][11] on click-based graphical passwords. Dirik et al. [5] proposed automatic dictionary attacks on PassPoints, whereby mean-shift image segmentation [22][23] was applied to detect centroids of segments that are not too large or too small, and then grid-squares were sorted according to their probabilities to be in a password, calculated by applying a user attention model, to build attack dictionaries. Experiments on a representative image found 8.45% of the passwords using a dictionary of approximately 2^{31} entries whereas the full password space was of 2^{40} entries [5]. Salehi-Abari et al. [10] proposed an automatic hotspot-based dictionary attack against PassPoints-style graphical passwords, which were subsequently improved by van Oorschot et al. [11]. In these attacks, both corner and centroids are detected to form a set of predicted click-points. This set is too large to build an effective attack dictionary by traversing all combinations of the predicted click-points. Heuristic patterns of click-points in a password and salient regions detected using a visual attention model are used to select likely combinations of the predicted click-points in building attack dictionaries. Experiments using two representative images with a full password space of 2^{43} entries found 7-16% of the passwords using dictionaries each of approximately 2^{26} entries, and 48-54% of the passwords using dictionaries each of approximately 2^{35} entries [11].

In human-seeded attacks [8][9] on PassPoints, click-points from a small set of users are harvested for targeted images, and attack dictionaries are constructed using a first-order Markov model or an independent probability model. In a lab study [8][9] on two representative images with the full password space of 2^{43} entries, the method found 20-36% of the passwords using dictionaries each of 2^{31} to 2^{33} entries built with the independent probability model and 4-10% of the passwords within 100 guesses using the first-order Markov model.

An analysis of user-selected passwords reported in [7] revealed that for CCP, users still tended to select click-points falling within known hotspots, but the patterns of click-points exploited in successful dictionary attacks on PassPoints were eliminated; on the other hand, PCCP had eliminated both common patterns and hotspots. Therefore, PCCP was considered to be robust to all the known dictionary attacks.

3. DISCRETIZATION SCHEMES

We briefly described several discretization schemes in Section 1. Since Optimal Discretization [17] is the same as Centered Discretization [16] when the edge problem of discretization is avoided and the image-dependent discretization scheme [18] lacks detailed information for an actual implantation, our security analysis will focus only on Robust Discretization and Centered Discretization. In the following, we describe these two schemes in detail.

3.1 Robust Discretization

Robust Discretization [15] uses three offset grids to guarantee that for each point in the image, there exists at least one grid in which the point is r-*safe*. Specifically, each grid $G_k, k \in \{1,2,3\}$ has a grid-square size of $6r \times 6r$, with an offset from each other by a distance of $2r$ in both directions. Figure 1 shows an example of the three grids in Robust Discretization, along with two points A and B, wherein point A is r-safe in grid G_0, and point B is r-safe in both G_1 and G_2. When creating a password, one of the three grids is selected for each click-point. When a click-point is r-safe in more than one grid, a selection algorithm is needed to select one from the candidate grids. An optimal algorithm [16] selects the grid wherein the point is closest to the center of the grid-square in order to minimize the occurrence of false accepts and false rejects of passwords.

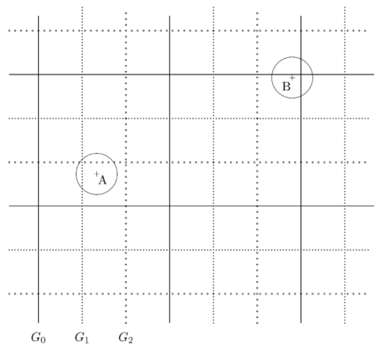

Figure 1. Robust Discretization: Three grids G_0, G_1, and G_2, wherein A is r-safe in G_0 and B is r-safe in both G_1 and G_2 (taken from [15]).

With Robust Discretization, each click-point in a password is associated with an image, a grid selected for the click-point, and a grid-square in the grid where the click-point lies in. Therefore a password can be represented by a sequence of grid-squares in the selected grids with corresponding images associated with the click-points of the password; and a cryptographic hash of the sequence is stored in the system. Identifiers of the selected grids for the password's click-points are also stored in the system but in plaintext. During authentication, the stored grid identifier for each click-point is retrieved to determine the exact grid-square which a user-clicked point actually lie in. The hash of the resulting sequence of the grid-squares with the corresponding images is calculated and compared with the stored hash to determine if authentication is a success or failure.

3.2 Centered Discretization

Centered Discretization [16] finds, for each click-point in a password, a grid wherein the click-point is the center of a grid-square in the grid. This grid can be uniquely determined by an offset to the grid aligned with both x-axis and y-axis. This offset is stored in plaintext in the system, and will be used to reconstruct the grid during authentication. To make the point be r-*safe*, each grid has a grid-square size of $2r \times 2r$. The grid's offset is For every click-point, Centered Discretization has the same maximal tolerance level of click-variations, which is r pixels on each direction; whereas for click-points at different locations, Robust Discretization has various maximal tolerance levels of click-variations, in the range from r to $3r$. If the click-point is at the center of its grid-square, it can accept click-variations within $3r$ pixels on each direction. That maximal tolerance level reduces gradually to r when the click-point moves from the center towards an edge of the grid-square.

For the same guaranteed tolerance range r of click-variations, a grid-square in Centered Discretization is one third in size of that in Robust Discretization along each direction, resulting in the size of the full password space in Centered Discretization about $9^5 \approx 2^{16}$ times of that in Robust Discretization for passwords of 5 click-points. For the same grid-square size, Centered Discretization has a guaranteed tolerance level three times that Robust Discretization offers, whereas both have the same size for the full password space.

Security of Centered Discretization is compared with that of Robust Discretization in [16] using PassPoints, with the conclusion that both discretization schemes have the same level of security when both have the same size of grid-squares, and that Centered Discretization is more secure when they are both *r-safe*.

4. A SECURITY ANALYSIS

In this section, we present a security analysis of Robust Discretization and Centered Discretization. We first explain theoretical models assumed in our analysis, and describe key observations that motivated our attacks. Next, we introduce our attack methodology, which consists of two stages. In the first stage, image processing techniques are used to detect potential click-points in each image. In the second stage, discretization information stored in the system is exploited to build attack dictionaries. This security analysis is generic, as it is independent from any specific click-based graphical password scheme.

4.1 Theoretical Models

4.1.1 Threat Model

HTTP Authentication [24] has been widely used in Web applications. It contains two types of authentication protocols, Basic Access Authentication and Digest Access Authentication. The latter is more secure than the former. In Digest Access Authentication, password hash is calculated at the client side. As a consequence, the discretization grid information for each click-point of a password needs to send to client when a discretization scheme is used in Digest Access Authentication, and thus accessible to adversaries. Since discretization grid information is stored in plaintext, it is also accessible to adversaries in offline dictionary attacks. A password guess can be verified with the system for online dictionary attacks or with the stored password hash for offline dictionary attacks. A user ID is frequently stored in a Web browser and thus accessible to adversaries.

In summary, the threat model in our studies is as follows: *Adversaries have access to everything except passwords or the information accessible only with passwords. Particularly, adversaries have access to the discretization information, user IDs, and hash values of passwords stored in the system.*

We note that the above threat model is generic, not necessarily tied with Web applications. For example, the threat model is applicable for offline dictionary attacks whereby adversaries have access to the authentication information stored in the authentication server.

4.1.2 Independent Model of Click-Points

For generality, we assume that click-points are mutually independent. For some click-based graphical password schemes such as PassPoints, there might exist some correlations among the click-points in a password, as exploited by successful dictionary attacks on PassPoints described in Section 2.2. The correlations can be exploited to improve efficiency of the dictionary attacks to be presented in this paper. For example, the heuristic patterns of click-points used in [10][11] or the first-order Markov model used in [8][9] can be applied to improve our dictionary attacks on discretization with PassPoints.

4.2 Key Observations

Our security analysis was motivated with the following two observations. The first is on click-points likely selected in click-based graphical passwords, and the second is on the distribution of human click-variations.

4.2.1 Click-Points of Graphical Passwords

A study [7] reveals that hotspots exist in both PassPoints and CCP, but are eliminated in PCCP, thanks to the requirement that a click-point is selected within a randomly positioned viewport. We conceive that click-points in PCCP are likely salient points in viewports that should be detectable with image processing techniques. That conception led us to using corners and centroids in images to predict click-points for all click-based graphical password schemes.

There are typically too many detected salient points (i.e., corners and centroids) that a dictionary built by traversing all their combinations is too large to mount a meaningful dictionary attack. Patterns of click-points in a password and other techniques such as a user attention model have been used to select only likely salient points and their combinations in dictionary attacks on PassPoints [5][10][11]. These techniques can no longer be used to attack PCCP, as concluded in [7]. With the independent model of click-points assumed in this paper, detected corners and centroids cannot in general mount an effective dictionary attack on a click-based graphical password scheme. We need to reduce the size of dictionaries significantly to mount a meaningful dictionary attack.

4.2.2 Distribution of Human Click Variations

People have different accuracy in re-clicking a point. Figure 2 shows the result of a study by Chiasson et al. [12] with 24 university students on the accuracy that users re-entered click-points for both stages of password confirmation and login. More than 70% of the users had small variations, in the range from 1 to 3 pixels, in re-entering click-points. Click-variations for the remaining users distribute in the long tail in the figure, ranging from 4 to more than 51 pixels.

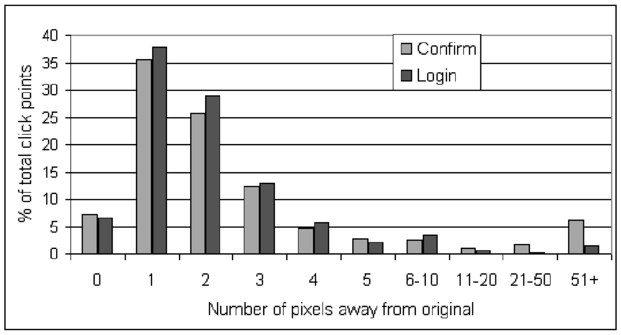

Figure 2. Accuracy in re-entering click-points (taken from [12]).

For acceptable usability, a practical system should select r of a discretization scheme conservatively so that most users should be able to enter their passwords correctly. In other words, r should be selected significantly larger than most users' click-variations. For example r should be 4 pixels or larger from Figure 2. We conceive that the disparity between a large tolerance range of the system and small click-variations for most people can be exploited to remove salient points unlikely related to the click-point currently seeking for: salient points far away from likely locations of the current click-point are removed. Additionally, if the location of a click-point is known within a grid-square as in Centered Discretization or predictable as in the image-dependent discretization scheme [18], salient points can be rank-ordered by their distances to the click-point's locations inside their respective grid-squares: a salient point with a shorter distance is more likely to be the click-point. This conception led to our method to build effective dictionaries from detected salient points by exploiting the location information of a click-point leaked by a discretization scheme, as to be described in Section 4.4.

4.3 Prediction of Click-Points

Like [10][11], corners and centroids are detected with image processing techniques and used as salient points to predict click-points for click-based graphical password schemes such as PCCP, CCP, and PassPoints. A corner is the intersection of two edges, and a centroid is the center point of an object. Corner detection and centroid detection we recommend to use are described as follows.

4.3.1 Corner Detection

Harris corner detection [25] implemented by Kovesi [26] is used to detect corners. Kovesi's implementation contains three parameters, "sigma" (σ), "thresh" (θ), and "radius" (r), where σ is the standard deviation of a smoothing Gaussian filter, θ measures the strength of non-maxima-suppression: all corners with weight lower than θ are suppressed, r is an inhibition radius, measured in pixels around a detected corner. For large θ, Kovesi's detector detects strongly distinctive corners. Weakly distinctive corners can also be detected when the value of θ reduces. The following unique method is applied to detect corners in an image:

1. *Detection of strongly distinctive corners.* Kovesi's detector is applied to the image with the following parameters: $\sigma = 1, \theta = 65, r = 2$. The set of corners detected in this step is denoted by S_1.
2. *Detection of weakly distinctive corners.* Kovesi's detector is applied to the image with the following parameters: $\sigma = 1, \theta = 5, r = 4$. The set of corners detected in this step is denoted by S_2.
3. *Detection of weakly distinctive corners in 2X window.* The image is smoothed using a Gaussian filter with the standard deviation of 1.0, and then down-sampled. Kovesi's detector is applied to the resulting image with the following parameters: $\sigma = 1, \theta = 7, r = 2$. The set of corners detected in this step is denoted as S_3. This is equivalent to using Harris detector with a window twice the size in both directions of the default window used in Kovesi's implementation.
4. *Detection of weakly distinctive corners in both scales.* For each point p in S_2, if there exist a point in S_3 within a distance of 7 pixels from p, then p is kept in S_2. Otherwise p is removed from S_2. The set of survived corners in S_2 is denoted by S_2'.
5. *Detected corners.* The union of S_1 and S_2', $S = S_1 \cup S_2'$, is output as the detected corners of the image.

The detected corners in S are either strongly distinctive corners or weakly distinctive corners of large structures. Figure 3 shows the detected corners for image "pool" which has been widely used in security studies of PassPoints. Each corner is marked by a '+' on the image. Compared to the detected corners of the same image shown in [11], our corner detection produces a slightly smaller number of corners. The major difference between the two corner detection methods is around the tree leaves in the image: weak corners of small structures are suppressed in our corner detection.

4.3.2 Centroid Detection

Centroids are detected with the same method as in [11] except using a different implementation of mean-shift segmentation algorithm [22][23] for our convenience. We use the implementation [27] to segment images with the following values of the three parameters used by the implementation:$sigmaS = 7$, $sigmaR = 9$, and $minRegion = 100$. The last parameter is a threshold for the minimal segment area (in pixels). For the resulting segments, we calculate only the segments with an area of 500 pixels or less. Segments too large or too small are ignored. Only the segments with an area in the range from 100 pixels to 500 pixels are used to calculate centroids.

A segment with an irregular shape may result in a centroid outside the segment. This type of centroid is unlikely selected as a click-point, and thus should be removed. Removing this type of centroid is achieved by checking if the centroid is outside its segment or not. If the centroid is outside the segment, the centroid is removed. Otherwise the centroid is kept. The survived centroids are output as the detected centroids.

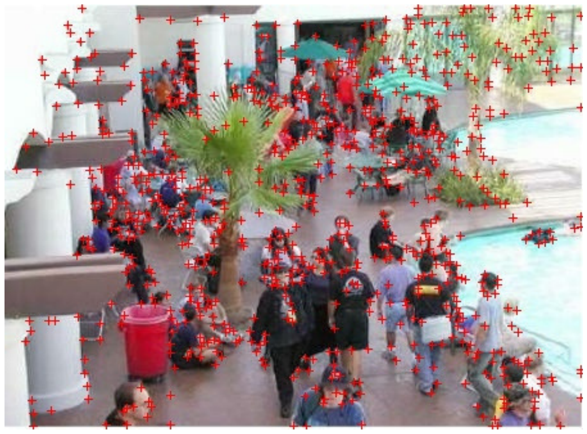

Figure 3. Detected corners (each corner is marked by a '+').

4.4 Construction of Dictionaries

For each click-point in a password, once salient points are detected, the discretization information stored in the system is exploited to remove salient points that are unlikely to be the click-point for Robust Discretization, or to rank-order combinations of salient points by their predicted probabilities to be the password for Centered Discretization. This section represents the key insights behind our attacks.

4.4.1 Dictionaries for Centered Discretization

In Centered Discretization, a click-point is the center of some grid-square of the grid associated with the click-point. According to the distribution of click variations shown in Figure 2, a point closer to but not at the center of its grid-square has a higher probability to be the click-point. A password consists of multiple click-points. For each password, we can build a model using distances of salient points to the centers of their grid-squares to predict the probability that a sequence of the salient points is the password.

Before applying the model, we prune salient points so that each grid-square contains at most one salient point. If a grid-square contains more than one salient point, only the salient point closest to the center of the grid-square is kept and all the other salient points in the grid-square are removed. Then one of the following two models is used to build attack dictionaries with entries ordered by their probability to be the password.

Zooming-out Window Model. In this model, we use a series of concentered windows with different sizes to determine the order of grid-squares in constructing a dictionary. More specifically, we place a selecting window W_d of size $2d \times 2d$ at the center of each grid-square, where $0 \le d \le r$, and assign a salient point a priority value d if the salient point is in window W_d but not in window W_{d-1}, where we assume that W_{-1} is empty. A salient point with a smaller priority value is closer to the center of its grid-square, and thus considered to be more probably to be the click-point. Salient points not in any selecting window are removed. The priority value of a grid-square is the priority value of the salient point inside the grid-square. Grid-squares containing no salient point are removed. The survived grid-squares are traversed for all possible combinations of grid-squares. Based on the independent model of click-points, the priority value of a sequence of grid-squares is assigned to be the sum of the priority values of the constituent grid-squares. The sequences with priority value s form a set denoted as G_s. A sequence in set G_i is more likely to be the password than a sequence in G_j if $i < j$. A series of dictionaries $\{D_s | s = 0,1,\cdots\}$ can thus be constructed as follows:

$$D_s = \cup_{k=0}^{s} G_k \tag{1}$$

These dictionaries have the following relationship:

$$D_0 \subseteq D_1 \subseteq D_2 \subseteq \cdots \tag{2}$$

Password guesses in a dictionary D_s can therefore be partitioned into a layered order: the guesses in D_0 are ordered as the first layer and tested first; the guesses in D_1 but not in D_0 (i.e., the guesses in G_1) are ordered as the second layer and tested next. This process continues until all the guesses in D_s are ordered and tested. In other words, guesses in a dictionary D_s are tested according to their priority values, from low to high. Guesses with the same priority value can be tested in any order.

Probability Model. If we ignore the value at 0, we can approximate the distribution of user click-variations shown in Figure 2 with a normal distribution

$$f(d) = a \cdot exp(-\frac{d^2}{2\sigma^2}) \tag{3}$$

where a is a normalization coefficient and d $(0 \le d \le r)$ is the distance of the point to the center of its grid-square. For simplicity, Eq. (3) is also used for the case $d = 0$ in our studies, although $f(d)$ at $d = 0$ deviates considerably from the value shown in Figure 2. We believe that such a deviation will be tolerated by our probability model, as this model is much more accurate than the zooming-out window model. The belief is confirmed true by our experimental results which will be reported later: even the zooming-out window model has produced good results.

For each grid-square containing a salient point, we calculate a probability that the click-point actually lies in the grid-square by using Eq. (3), with d being the distance of the salient point to the center of its grid-square. For every possible password guess constructed from the grid-squares containing salient points, according to the independent model of click-points, we calculate its probability to be the password as multiplication of the probabilities of the constituent grid-squares. These guesses are then sorted by their likelihood of being the password. An attack dictionary can select any number of most probable guesses, and guesses are tested in the order of their probability of being the password.

4.4.2 Dictionaries for Robust Discretization

In Robust Discretization, different schemes can be used to select a grid when more than one grid satisfies the guaranteed tolerance range r. The maximum tolerable range of click-variations runs from r to $3r$, depending on the location of the click-point in its grid-square. To have a good balance between usability and the size of the full password space, it is typical to choose the optimal grid that maximizes the safe distance of the click-point to the edges of the grid-square. This optimal Robust Discretization was used in [16] to study security of Robust Discretization, and has been adopted in our studies too. The optimal Robust Discretization leads to an eligible region that a click-point may lie in. Figure 4 shows the eligible region surrounded by the dashed lines inside the greyed grid-square if grid G_0 is used for the click-point. The eligible region is about 36% of the grid-square. With this information, we remove salient points that are not inside any eligible region. Then we select all the grid-squares that contain at least one salient point, and build a dictionary by traversing all their combinations using the independent model of click-points.

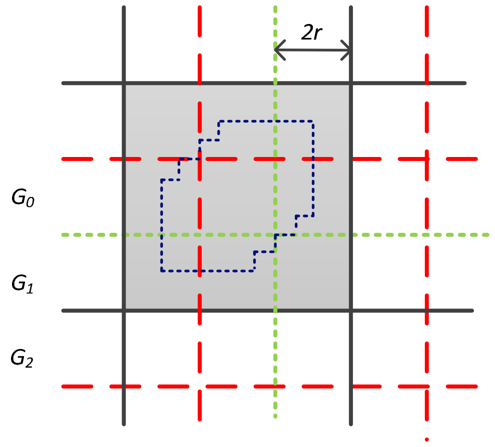

Figure 4. The eligible region that a click-point can lie inside a grid-square for grid G_0.

5. DICTIONARY ATTACKS ON PCCP USING DISCRETIZATION

5.1 Why do We Target PCCP?

PCCP was selected as the click-based graphical password scheme in our studies of the two representative discretization schemes since it was considered the most secure click-based graphical password scheme to date, robust to all the dictionary attacks reported on click-based graphical passwords. It was estimated in [13] that hotspot-based dictionary attacks would guess only 0.03% of PCCP passwords, a percentage that is extremely small as compared to the ratios of passwords guessed by dictionary attacks reported on PassPoints.

For PCCP, next image I_{next} is selected by a deterministic function f of current image I_{curr}, the current tolerance square, and the user ID:

$$I_{next} = f(I_{curr}, gridSquare, userID) \tag{4}$$

Eq. (4) makes PCCP rely on a discretization scheme in selecting a next image so that approximate clicks would result in the same next image. Due to this image selection mechanism, we cannot build a general dictionary to attack all users as in attacking PassPoints. Instead, we need to build personalized attack dictionaries for each individual user, which makes dictionary attacks much more complex than those on PassPoints. Like reported security studies on PassPoints, we assume that each password consists of 5 click-points in our studies.

According to the generic threat model in Section 4.1.1, user IDs are known to adversaries. Once a user ID is known, a single login attempt can obtain the first image used by the account. We also assume that a service is available to adversaries to determine the next image for every input triple consisting of a current image, a clicked grid-square, and a user ID. The generic threat model in Section 4.1 is thus refined as follows for PCCP: *Adversaries have access to everything except passwords or the information accessible only with passwords. Particularly, adversaries have access to the discretization information, user IDs, and hashed password stored in the system. For PCCP, adversaries have access to each user's first image and the image database. The deterministic function is used as an oracle that adversaries can query to get the index of the next image used in entering a password but have no access to and would not try to find the internal logic or parameters of the deterministic function. Adversaries do not have access to any password or any of the remaining images any user uses.*

5.2 Building Personalized Attack Dictionaries

To build personalized attack dictionaries for each participant in our studies, we detected salient points for the first image, processed these points using the grid information associated with the image, as described in Section 4.4, and obtained the grid-squares, $g_k^1|k = 0, 1, \cdots$, which contain at least one salient point. Each of the grid-squares is a password guess of the first click-point. These grid-square form a set $D^1 = \{g_k^1|k = 0, 1, \cdots\}$, which comprises of guesses of the user's password up to the first click-point.

Suppose we have finished the construction of set $D^i = \{g_k^i|k = 0, 1, \cdots\}$ comprising of password guesses of the user's password up to *i*-th click-point, $g_k^i|k = 0, 1, \cdots$, where $i < 5$. We use the following procedure to build the set D^{i+1} comprising of the password guesses of the user's password up to the (*i+1*)-th click-point.

- Initialize D^{i+1} to empty, and a count k to 0.
- For each entry $g_k^i \in D^i, k = 0, 1, \cdots$, we obtain the next image by querying the oracle with the current image, the current grid-square, and the user ID. Then we detect and process the salient points of the obtained image in the same way as for the first image. Each of the grid-squares containing at least one salient point for the obtained image is a guess of the (*i+1*)-th click-point of the user's password, and combined with the guess of the preceding click-points of g_k^i. Each resulting guess g_k^{i+1} is inserted into D^{i+1}, with the current value of the count k as the lower index k of the guess. After a guess is inserted into D^{i+1}, the count k is increased by 1.

The above procedure is applied until all the five click-points have been guessed. The resulting set D^5 is the attack dictionary, $D = D^5$, built for the user.

5.3 Experimental Setting

We searched Internet for images with similar complexity as the representative image shown in Figure 3, which has been widely used in security studies on click-based graphical password schemes, and collected 1200 images for our experimental studies on PCCP. These images were cropped to size of 451×331 pixels if necessary. The grid-square sizes in our studies ranged from 9×9 to 19×19. For passwords of 5 click-points, the size of the full password space for an individual user was 2^{43} if the largest grid-square 19×19 was used, or 2^{54} if the smallest grid-square 9×9 was used. The detection algorithms described in Section 4.3 were then applied to each image to detect the salient points in the image. The detected click-points were saved together with the image for later studies. We also implemented PCCP based on the description in [13]. In our implementation, MD5 hash function was used as the deterministic mapping function in Eq. (4) to select the next image, by dividing the resulting hash value by 1200, the total number of images in in our studies. A count starting from 0 was added to the list of parameters of the hash function. The count was increased by 1 when a new image was selected in entering a password, including dropped images as explained next. When the next image selected by the function repeated a preceding image in entering a password, the image was dropped and a new image was selected using the previously clicked grid-square. This process repeated until an image different from all the preceding images was selected. This process would result in a unique sequence of 5 different images for each user and her password.

Since the index of a grid-square depends on the size of the grid-square, Eq. (4) will select a different image and result in a different password for different sizes of grid-squares in our studies even if a user uses the same current image and the same click-point on the image in her password. This would complicate our experiments on different grid-square sizes since a participant would have to create and remember one password for every grid-square size, and we could not rule out interference in memorizing multiple passwords by a participant. To simplify the tasks each participant needed to perform in our studies and to avoid possible interference, we made a minor modification to Eq. (4) as follows:

$$I_{next} = f(I_{curr}, pointCoordinates, userID), \qquad (5)$$

where $pointCoordinates$ is the coordinates of the recovered click-point calculated from the user's click. For Centered Discretization, the recovered click-point is the center of the clicked grid-square. For Robust Discretization, the recovered click-point is the point with a given bias from the center of the clicked grid-square, with the bias being the coordinates of the click-point relative to the center of the grid-square it lies in. The bias for each click-point is stored in the oracle to determine the next images for the user. If a user re-clicks the same grid-square in a login attempt, the recovered click-point is the actual click-point, and thus the correct next image is selected by Eq. (5). If a wrong grid-square is clicked, the recovered click-point is different from the actual click-point, and thus a wrong image is selected. Since Eq. (5) depends only on the coordinates of a click-point, which does not change for different grid-square sizes, a participant would create a single password valid for both Centered Discretization and Robust Discretization with different grid-square sizes, greatly simplifying our experiments. Such a modification has no impact on our security studies of the discretization schemes with PCCP when the images in the database are of similar complexity.

5.4 Password Collection

We conducted an in-lab user study with 38 voluntary participants (29 males and 9 females) to collect PCCP passwords. All the participants were engineering students ranging from undergraduate senior to Ph. D. students. All of them were good at using computers and Web browsing but none of them had studied computer or network security, or used the graphical password schemes under study. The participants were trained to get familiar with the experimental tasks in advance. They used Web browsers on their own PCs to create passwords and log into a remote authentication server. Each participant was required to select a password of 5 click-points in length. The selected password was confirmed immediately after creation with all the discretization schemes and configurations under test. Each participant was given up to three trials to pass the confirmation test. If a confirmation login failed three times with any discretization scheme and any configuration, the participant was asked to recreate a new password. This process was repeated until the created password was confirmed successfully. Our experimental results confirmed the distribution of click-variations shown in Figure 2: most participants could pass the confirmation test in a single trial, implying that their re-click variations were 3 pixels or less on each direction from the click-points.

Once a password had been successfully confirmed, the participant was required to pass two login tests, occurred at 24 hours and 7 days, respectively, after creation of the password. In each login test, a participant was required to log in successfully within three trials for each of the discretization schemes and configurations under test. Otherwise the login test was claimed a failure. A participant who failed in either login test was required to re-do the above procedure, only for the failed discretization schemes and configurations, until he or she passed the two login tests successfully for all the discretization schemes and configurations under test. This process was designed to ensure that every participant selected a PCCP password that could be remembered for a reasonably long time. The PCCP passwords which had successfully completed the two login tests were collected.

For each participant, we built personalized dictionaries as described in Section 5.2. Each guess in a personalized dictionary was tested in the following order until the password was found, the dictionary was exhausted, or the termination condition was met: randomly for Robust Discretization; from highest probability to the lowest probability for Centered Discretization using the probability model; or from lowest layer to the highest layer and randomly within the same layer for Centered Discretization using the zooming-out window model. The attack result was recorded.

5.5 Experimental Results

5.5.1 Attack Results for Robust Discretization

For Robust Discretization, only the grid-square size of 19×19, with $r = 3$ pixels, was studied since a smaller grid-square size leads to poor usability according to [16]. For this setting, 15 out of the 38 passwords were found by our attack, with a success rate of 39.4%, and with the attack dictionaries each of approximately 2^{35} entries.

To find out how effectively Robust Discretization has helped reduce dictionary sizes, we constructed the attack dictionaries traversing all combinations of the detected salient points, i.e., without exploiting Robust Discretization. These dictionaries were all of approximately 2^{39} entries. For each user, the individual full search space contains 2^{43} entries for the size of images we used, which is the same size as the full password space for all users in PassPoints. This is because that once the user ID and the first image are determined, each grid-square would select a next image, and there are 391 (= $\lfloor 451/19 \rfloor \times \lfloor 331/19 \rfloor$) grid-squares per image. Therefore, the first click-point has 391 possibilities. For each grid-square in the first image, there are 391 grid-squares in the second image, which contributes 391 possibilities. Therefore the first two click-points contribute $391 \times 391 = 391^2$ possibilities. This procedure repeats until all the five click-points have been taken into consideration, resulting in $391^5 \approx 2^{43}$ possibilities, which is the full search space for an individual user. The prediction of click-points using the corner and centroid detection algorithms described in Section 4.3 has thus effectively reduced the password search space by a factor of 2^4 (= $2^{43} / 2^{39}$), i.e. 4 bits, and Robust Discretization has further reduced the search space by a factor of 2^4 (= $2^{39} / 2^{35}$), i.e., 4 additional bits.

On the other hand, Robust Discretization removes an average of $1.0 - \sqrt[5]{2^{35}/2^{39}} = 42.6\%$ of grid-squares per image when constructing attack dictionaries, which roughly agrees with the 36% area ratio of the eligible region to the grid-square size (see Section 4.4.2).

5.5.2 Attack Results for Centered Discretization

For Centered Discretization, grid-square sizes ranged from 9×9 to 19×19 were used, leading to $r = 4.5, 5.5, \ldots, 9.5$ pixels. For each r, dictionaries were built respectively with the zooming-out window model using d running from 2 to $\lfloor r \rfloor$, and with the probability model where the size of a dictionary is controlled at the end of the range of an integer power of 2.

Table 1 summarizes our attack results achieved with the zooming-out window model. The leftmost column shows attack success rates (the ratio of the number of guessed passwords to the total number of passwords). Note that for the same d, the success rate remains constant for different grid-square sizes, since a click-point always remains at the center of its grid-square, which is also the center of the selecting window with a size of $2d \times 2d$. When the grid-square size is fixed, the attack success rate is correlated with the dictionary size; the larger dictionaries are used, the higher the attack success rate can be achieved. With attack dictionaries each of approximately 2^{35} entries, our attack successfully guessed 63.2% of the passwords when the grid-square size was 19×19, whereas the full password space was of 2^{43} entries.

Table 1. Attack success rates using the zooming-out window model

Success rate (%)	Dictionary size (bits) per grid-square size					
	19x19	17x17	15x15	13x13	11x11	9x9
76.4	38					
71.1	37	38				
71.1	36	37	39			
63.2	35	36	38	40		
57.9	33	35	37	39	41	
39.5	30	32	34	36	38	41
7.9	27	29	31	33	35	38
2.6	21	23	25	27	30	33

Table 2 summarizes our results achieved with the probability model. The leftmost column is dictionary size (n bits). Attack results for some n values are omitted to avoid an overly lengthy table. When the grid-square size is fixed, the attack success rate is correlated with the dictionary size; the larger dictionaries are used, the higher the success rate can be achieved. For the grid-square size of 19×19, and thus the full search space of 2^{43} entries, the probability model successfully guessed 69.2% of the passwords with attack dictionaries of approximately 2^{35} entries, and guessed 50% of the passwords with attack dictionaries each of approximately 2^{30} entries.

In addition to a better controlled dictionary size, the probability model also produced better success rates than the zooming-out window model did, as we can easily see by comparing Table 1 and Table 2.

Table 2. Attack success rates using the probability model

Dictionary size (bits)	**Attack success rate (%) per grid-square size**					
	19x19	**17x17**	**15x15**	**13x13**	**11x11**	**9x9**
43					65.8	57.9
40		76.3	73.7	71.1	60.5	34.2
39	81.6	76.3	71.1	65.8	52.6	23.7
38	74.4	76.3	71.1	63.2	50	18.4
35	69.2	65.8	52.6	36.9	18.4	7.9
33	63.2	55.3	39.5	18.4	7.9	5.3
32	60.5	50	31.6	10.5	7.9	0
31	52.6	34.2	18.4	7.9	5.3	0
30	50	31.6	10.5	7.9	5.3	0
29	31.6	18.4	7.9	5.3	0	0
28	21.1	10.5	7.9	5.3	0	0
26	10.5	7.9	5.3	0	0	0
23	5.3	2.6	0	0	0	0

Table 3. Full password space and salient point dictionary size for different grid-square sizes

	Grid-square size					
	19x19	**17x17**	**15x15**	**13x13**	**11x11**	**9x9**
Salient point dictionary size (bits)	39	40	41	42	43	44
Full password space (bits)	43	44	46	48	51	54

For each grid-square size, we also constructed attack dictionaries using all the detected salient points, but without exploiting any information leaked by Centered Discretization. Table 3 shows the size of each of such dictionaries and the corresponding full password space size. By comparing Table 1 with Table 3 and Table 2 with Table 3, we see how much the information leakage by Centered Discretization has contributed to our attacks, and how much our salient point detection methods have contributed to our attacks, respectively. For example, when the grid-square size was 19×19, our salient point detection reduced the search space by a factor of 2^4 ($= 2^{43} / 2^{39}$), i.e. 4 bits, as illustrated in Table 3. When the information leakage by the discretization was exploited, we could further reduce the search space by a factor of 2^9 ($= 2^{39} / 2^{30}$), i.e. 9 bits, but still successfully guessed 39.5% of the passwords using the zooming-out window model, and 50% of the passwords using the probability model. This clearly suggests that Centered Discretization can leak a significant amount of password information.

To compare the success rates for both Robust Discretization and Centered Discretization with the same 19×19 grid-square size, our attack on Centered Discretization produced a much higher success rate for similar dictionary sizes or required much smaller dictionaries to achieve similar success rates. This result contradicts the previous conclusion in [16] that they both have the same level of security to dictionary attacks. This contradiction is due to the fact that the information leakage caused by discretization was ignored in [16].

The significantly different success rates between the two discretization schemes for the same grid-square size can be explained by the fact that the discretization schemes leak click-point information at different uncertainty levels: *whether a click-point is in the center of a grid square* in Centered Discretization vs. *whether a click-point is inside an eligible region, i.e. 36% of a grid square* in Robust Discretization. That is, Centered Discretization leaks much more click-point information than Robust Discretization does, and thus the former suffers more from our dictionary attacks.

On the other hand, for the same r, Centered Discretization tends to have a better security than Robust Discretization, as the former allows a much larger theoretical password space. For example, when $r = 3$, our attack guessed 39.4% of the passwords when Robust Discretization (with grid-square size of 19×19) was used, and 7.9% of the passwords when Centered Discretization (with grid-square size of 9×9) was used, both used dictionaries each of approximately 2^{35} entries.

6. DISCUSSIONS

We have not conducted experimental studies on the Image-Dependent Discretization (IDD) scheme [18] due to lack of details to implement the scheme. However, it is possible to apply our ideas for attacking Centered Discretization to mount dictionary attacks on this IDD as well.

In the IDD scheme, image features are analyzed with image processing and computer vision techniques to predict likely click-points, which are then placed at centers of Voronoi polygon tiles. When such prediction is reasonably accurate, we can use the distance of a salient point to the center of a Voronoi polygon tile to model the probability that the salient point is a click-point. In this case, we can actually use the same techniques as we used for Centered Discretization to rank-order sequences of Voronoi polygon tiles that contain salient points. This way, we will be able to achieve a much improved success rate in password guessing than that can be achieved without exploiting click-point information leaked by IDD.

On the other hand, if the prediction of click-points is not very accurate, then the IDD scheme may not be as attractive as other discretization schemes, since a wrong prediction may result in a click-point far away from the center of a Voronoi polygon tile, leading to a significant reduction of the tolerance range of click-variations, and thus may cause unacceptable false accepts and false rejects.

7. CONCLUSION

We have shown that two representative discretization schemes, Robust Discretization and Centered Discretization, both leak information on password click-points, weakening the security of click-based graphical password systems. This is an important issue that had been neglected by researchers and practitioners before our study.

Our purely automated dictionary attacks have successfully exploited click-point information leaked by each of the discretization schemes. In our experiments, our attack successfully guessed 69.2% of the passwords in PCCP when Centered Discretization was used, and 39.4% of the passwords when Robust Discretization was used, with attack dictionaries each of approximately 2^{35} entries whereas the full password space was of 2^{43} entries. In addition, for Centered Discretization, our attack still successfully guessed 50% of the passwords in PCCP when the attack dictionary size was reduced to approximately 2^{30}. These results clearly suggest that both representative discretization schemes leak a significant amount of password information.

PCCP was considered robust to all known dictionary attacks, and thus the most secure click-based graphical password scheme. However, our results for the first time show that PCCP is vulnerable to dictionary attacks when its implementation uses either Centered Discretization or Robust Discretization.

Our attacks are certainly also applicable to other click-based graphical passwords such as PassPoints and CCP. What our attacks have exploited is a fundamental security problem with the underlying discretization mechanisms, and it is straightforward to apply our attacks to PassPoints, CCP or any other click-based graphical passwords. It is worthwhile to highlight that since our attack assumes that password click-points are mutually independent, which is a conservative assumption, our results as reported just indicates a conservative success rate for schemes like PassPoints and CCP (like a lower bound). For example, the heuristic patterns of click-points used in [10][11] or the first-order Markov model used in [8][9] can be applied to significantly boost our attack's success rate on PassPoints. That is, when these attacks are combined with ours, they will lead to a dramatically improved success rate on guessing PassPoints passwords.

As such, our work calls for the design of countermeasures to address our attack, or the design of better discretization schemes or other alternatives. For example, for the sake of security, a discretization scheme should leak as little information as possible. On the other hand, a discretization scheme offering good usability tends to leak click-point information. A good balance between these two contradicting requirements remains a research issue.

This line of exploration is essential for implementing usable and secure click-based graphical passwords in various contexts (including Web applications with increasingly popular keyboardless client devices), and it is our ongoing work.

8. REFERENCES

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

[2] Wiedenbeck, S., Waters, J., Birget, J. C., Brodskiy, A., and Memon, N. 2005. Authentication using graphical passwords: Effects of tolerance and image choice. In *Proc. Symp. on Usable Privacy and Security (SOUPS'05)*.

[3] Wiedenbeck, S., Waters, J., Birget, J. C., Brodskiy, A., and Memon, N. 2005. PassPoints: Design and longitudinal evaluation of a graphical password system. *Int. Journal of Human-Computer Studies (Special Issue on HCI Research in Privacy and Security)*. 63, 102–127.

[4] Chiasson, S., van Oorschot, P. C., and Biddle, R. 2007. A second look at the usability of click-based graphical passwords. In *Proc. Symp. on Usable Privacy and Security (SOUPS'07)*.

[5] Dirik, A., Menon, N., and Birget, J. 2007. Modeling user choice in the PassPoints graphical password scheme. In *Proc. Symp. on Usable Privacy and Security (SOUPS'07)*.

[6] Golofit, K. 2007. Click passwords under investigation. In *12th European Symposium on Research in Computer Security (ESORICS'07)*. LNCS vol. 4734 (Sept. 2007).

[7] Chiasson, S., Forget, A., Biddle, R., and van Oorschot, P. C. 2009. User interface design affects security: Patterns in click-based graphical passwords. *Int. Journal of Information Security*. Springer, 8, 6 (2009), 387-398.

[8] Thorpe, J., and van Oorschot, P. C. 2007. Human-seeded attacks and exploiting hot-spots in graphical passwords. In *USENIX Security*.

[9] van Oorschot, P. C., and Thorpe, J. 2011. Exploiting predictability in click-based graphical passwords. *Journal of Computer Security*, 19, 4 (2011), 669-702.

[10] Salehi-Abari, A., Thorpe, J., and van Oorschot, P. C. 2008. On purely automated attacks and click-based graphical passwords. In *Proc. 24th Annual Computer Security Applications Conference (ACSAC)*.

[11] van Oorschot, P. C., Salehi-Abari, A., and Thorpe, J. 2010. Purely automated attacks on PassPoints-style graphical passwords. *IEEE Trans. Information Forensics and Security*. 5, 3 (2010), 393-405.

[12] Chiasson, S., van Oorschot, P. C., and Biddle, R. 2007. Graphical password authentication using cued click Ppoints. In *ESORICS'2007*. LNCS, vol. 4734/2007, 359–374.

[13] Chiasson, S., Forget, A., Biddle, R., and van Oorschot, P. C. 2008. Influencing users towards better passwords: Persuasive cued click-points. In *Proc. of HCI*. British Computer Society.

[14] Chiasson, S., Stobert, E., Forget, A., Biddle, R., and van Oorschot, P. C. 2012. Persuasive cued click-points: Design, implementation, and evaluation of a knowledge-based authentication mechanism. *IEEE Trans. on Dependable and Secure Computing*. 9, 2 (March/April 2012), 222-235.

[15] Birget, J. C., Hong, D., and Memon, N. 2006. Graphical passwords based on robust discretization. *IEEE Trans. Information Forensics and Security*. 1, 3 (2006), 395-399.

[16] Chiasson, S., Srinivasan, J., Biddle, R., and van Oorschot, P. C. 2008. Centered discretization with application to graphical passwords. In *Proc. 1st Conf. on Usability, Psychology, and Security (UPSEC'08)*.

[17] Bicakci, K. 2008. Optimal discretization for high-entropy graphical passwords. In *23rd Int. Symp. on Computer and Information Sciences* (Istanbul, Turkey, 2008).

[18] Kirovski, D., Jojie, N., and Roberts, P. 2006. Click passwords. In *IFIP Int. Federation for Information Processing, vol. 201, Security and Privacy in Dynamic Environments*. 351-363.

[19] Biddle, R., Chiasson, S., and van Oorschot, P. C. 2012. Graphical passwords: Learning from the first twelve years.

ACM Computing Surveys. 44, 4 (August 2012), Article 19, 1-41.

[20] Jansen, W. A. 2003. Authenticating users on handheld devices. In *Proc. of Canadian Information Technology Security Symposium* (2003).

[21] Passlogix. http://www.passlogix.com, site accessed May 6, 2011.

[22] Cheng, Y. 1995. Mean shift, mode seeking, and clustering. *IEEE Trans. on Pattern Analysis and Machine Intelligence*. 17, 8 (1995), 790-799.

[23] Comaniciu, D., and Meer, P. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*. 24, 5 (2002), 603-619.

[24] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. 1999. HTTP authentication: Basic and digest access authentication, *RFC 2617*.

[25] Harris, C. G., and Stephens, M. J. 1988. A Combined corner and edge detector. In *Proc. of the Fourth Alvey Vision Conference*. 147–151.

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

[27] *Edge Detection and Image Segmentation (EDISON) System*. http://coewww.rutgers.edu/riul/research/code/EDISON/doc/segm.html.