A Shoulder-Surfing Resistant Password Scheme

Shushuang Man, Dawei Hong, Jean-Camille Birget, and Manton Mathews

Abstract

We propose a password scheme. This scheme is obtained by adding a light graphic layer to the traditional text-based password scheme. In this graphic layer, we randomly generate patterns of icons which are easy for a user who owns the password to recognize and difficult for a shoulder-surfing attacker to find out. We prove that the proposed password scheme is shoulder-surfing resistant. To implement this scheme, we conducted experiments in which participants of all groups, from students to senior citizens, were comfortable in using the password system. A main security feature is as follows: Suppose that a shoulder-surfing attacker launched 480 times of attacks on a user. With a confidence level greater than 0.9 our implementation prevents the attacker from obtaining the password.

Keywords: password security, shoulder-surfing attack, randomized algorithm

1 Introduction

Computer security depends largely on passwords in order to authenticate users. We expect a good password scheme to comply with two conflicting requirements: (i) Usability, it should

*S. Man is with the Department of Mathematics and Computer Science, Southwest Minnesota State University, Marshall, MN 56258. Her research is supported by NSF grant CCR 0301571.

†The corresponding author. email: dhong@camden.rutgers.edu

‡D. Hong and J-C. Birget are with the Department of Computer Science, Rutgers University-Camden, Camden, NJ 08102. Their research is supported by NSF grant CCR 0310793 and in part by an ISATC pilot grant from Rutgers University. email: birget@camden.rutgers.edu

§M. Mathews is with the Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208

1
not be difficult for both user and system to implement; and (ii) Security, it should be secure. The traditional text-based password scheme, which today are used by a vast of majority of computer users, consists of a user-ID as a claim of the identity and a password as a proof that supports the claim. Here, both user-ID and password are strings of characters.

Studies have shown that, as a result, users tend to choose and handle their text-based passwords insecurely (cf. [6], [11], section 10.2, 10.6 of [13], and chapter 2 of [20]). To solve this problem, since mid 1990s, researchers have developed various graphical password schemes. Blonder [1] proposed a graphical password scheme in which a password is a sequence of mouse-clicks at points on a predetermined image. Jermyn et al [10] proposed DAS (Draw-A-Secret) scheme in which a password is a picture drawn on a 2-dimensional grid. The coordinates of the grids in which the picture touched are recorded in temporal order of the drawing. As long as same cells are crossed with same order, a user is authenticated. Let-Me-In is a graphical password interface by Microsoft, which works in a similar fashion to DAS [14]. The Map Authentication scheme [5] is based on navigation through a virtual world consisting of several sites. Recently, graphical password schemes are also applied to mobile devices such as PDA, for example, PicturePIN, a commercial product [16], and Picture-Password, a recently proposed scheme to set a standard for graphical password space used by PDA [9].

The proposed graphical password schemes above have not given a satisfactory answer to the usability and security. The proposed schemes suggest to entirely change the text-based password scheme which has been used and developed for three decades, and is now well built in most computer systems. Users are used to apply this password scheme. It would be hard to ask users to follow such an entire change [15]. From systems perspective, there are also some questions. For example, in current computer systems text-based passwords are encrypted (and then stored). But how to encrypt a sequence of “mouse-clicks” (which constitute passwords in most proposed graphical password schemes) is a question. None of the proposed graphical password schemes above has been proven more secure than the text-based password scheme so that costs of changing computer systems and retraining users can be justified [15].
A weak spot of the text-based password scheme is vulnerable to the shoulder-surfing attack. In this paper, we define shoulder-surfing attack as follows: A login process is recorded by an attacker who then can analyze the record to retrieve the password. Such a record is supposed to contain every move made by a user during a login process, for example, an attacker taped the process by a hidden video camera. Indeed, shoulder-surfing attack does not happen often. But its consequence is severe. In particular, a shoulder-surfing attacker may have no contact with the computer system, which makes it much harder to detect when and where such an attack happened.

Can we add a light graphic layer to the text-based password scheme to make it provably shoulder-surfing resistant? Here, by shoulder-surfing resistant we mean the following: With a record obtained by a shoulder-surfing attack, an attacker has only a small probability to either successfully retrieve the password, or to directly login, without alerting the computer system. Here, “direct login” means that an attacker tries to login using a record obtained by a shoulder-surfing without analyzing it. We note that even if an attacker was lucky enough to directly login, it does mean the attacker can obtain or change the password.

The commercial software, PassFaces, by the Real User Corporation has been around for more than three years [17]. This software adds a light graphic layer to the text-based password scheme to enhance the security. It works as follows: To complete a login, a user must correctly apply two steps.

**Implementation of Passfaces**

Step 1 Enter the user-ID and password in the same way as in the text-based password scheme.

Step 2 Pass a series of scenes. At a scene, a number of face photos randomly displayed on the screen. To pass a scene, choose the right one from these photos, and click on it.

Typically, Step 2 consists of six scenes each of which has nine photos displayed in a grid. A user of PassFaces needs to remember, in addition to a user-ID and a password as in the text-based password scheme, six faces.

This software well balances the usability and security. Users are able to remember faces well, and this ability is not linked to other factors such age, education, or intelligence. From
systems perspective, it is not heavy for today’s computer to render a series of scenes, and more importantly, the text-based password system, which have been built up for decades, is “reused”. Though a login process in PassFaces takes longer time, the security is enhanced. With stolen user-ID and password an attacker still could not pass the series of scenes. But, PassFaces is vulnerable under the shoulder-surfing attack. At each scene, though it is randomly displayed with other face photos, a face photo (used as a part of the password) is exposed when a user is clicking on it. At this point, the ability of remembering faces well turns around to against the user.

We propose a password scheme by adding a light graphic layer to the text-based password scheme. The graphic layer is as light as that in PassFaces. In this scheme we use three scenes. At each scene, instead of face photos we use icons to create random patterns which are easy for the user who owns the password to recognize, but difficult for a shoulder-surfing attacker to find out. Also, we propose two implementations of the scheme:

Implementation 1 targets a wide range of users. We prove its security as follows: With a record obtained by a shoulder-surfing attack, the attacker has a probability less than $1.4 \times 10^{-3}$ to directly login without alerting the computer system.

Implementation 2 targets higher security demands, but it needs some user training. We prove its security as follows: With a record obtained by a shoulder-surfing attack, the attacker a probability less than $2.2 \times 10^{-5}$ to directly login without alerting the computer system.

A main security feature of both Implementation 1 and 2 is as follows: Suppose that a shoulder-surfing attacker launched 480 times of attacks on user. Then with a confidence level greater than 0.9 our implementation prevents the attacker from retrieving the password.

This paper is organized as follows: In section 2, we first describe our proposed password scheme, then discuss its usability according to our experiments, and analyze its security under the shoulder-surfing attack. In section 3 we present a mathematical algorithm for our password scheme, then prove the soundness of the algorithm, and discuss how we implement it. In section 4 we solve a problem whose solution is used as the basis of the soundness of the algorithm. Because of the close relation between this problem and a classic problem in geometric probability, our solution should have its own right of interests.
2 A proposed password scheme

2.1 Description

Three insights, in combination, comprise the essential basis of our proposed password scheme. (I) Users are able to remember graphical elements such as screen icons well. In particular, with hints by graphical elements users are able to use long passwords. For example, it is not that easy to use a password “Stayman5CardMajorTransfer”. But, suppose that the password is divided into three substrings, “Stayman”, “5CardMajor” and “Transfer”, and that these substrings are individually entered each with a hint from graphical elements provided by the system. Then it will be easier for a user to use such a long password of 25 characters. (II) With today’s technology, it is not heavy to add a graphic layer to the text-based password scheme which has been built in computer systems. The commercial software, PassFaces, is a good example. (III) Mathematically, we can achieve the following: We ask a user to choose $k$ icons as a part of his password. Then we randomly position $N \gg k$ icons, including the $k$ chosen ones (as a part of a password), on a screen. In the sequel, we call an icon an object, in particular, a chosen icon a pass-object. Then with the positions of the $k$ pass-objects, we will almost surely have three and only three different cases, each of which happens with probability $\frac{1}{3}$. Since the $k$ pass-objects are a part of the password, the user surely knows which case is happening, and can correctly respond (enter the corresponding substring). In the mean time, since the $k$ pass-objects are hidden in the $N$ objects randomly positioned on the screen, it is hard for a shoulder-surfing attacker to make the correctly response, and more difficult to obtain the password.

Some of these ideas appeared in our early work [12]. Since then, we conducted a number of experiments to find proper choices of $N$ and $k$. The larger $N$ and $k$ are, the more secure our password scheme can be, but the more difficult is to use. We found that most users from a variety of groups (college or high school teachers, staffs and students, and senior citizens) are comfortable with $N = 120$ and $k = 4$. We take the two numbers. We will set a mathematical model to prove that by randomly positioning 120 objects on a screen, with
confidence level greater than \( 1 - 2e^{-54} \) we have the following three (and only the three) cases happen with equal probability \( \frac{1}{3} \):

Case 1 4 pass-objects form a convex hull that contains the center of the screen, Figure 1;
Case 2 4 pass-objects are all in the right half of the screen, Figure 2;
Case 3 4 pass-objects are all in the left half of the screen, Figure 3.

All figures are in section 5 which is placed at the end of this paper. We note that, in practical implementation, circles around the 4 pass-objects ought be erased, and that the eye-shape icon at the center is just a mark which does not count as one of the \( N \) object.

For both Implementation 1 and 2 we have conducted a number of experiments. In Implementation 1, a password consists of the following: a set \( \{s_1, s_2, s_3\} \) of three strings and a one-to-one mapping

\[
m_1 : \{\text{Case 1, Case 2, Case 3}\} \rightarrow \{s_1, s_2, s_3\}
\]

**Implementation 1**

Step 1 Enter user-ID.

Step 2 Pass a series of three senses as follows: enter \( s_{m_1(\text{Case } i)} \) if Case \( i \) is rendering on the screen, \( i = 1, 2, 3 \).

For the example (in (I) at the beginning of this section), we can have \( s_1 = \text{"5CardMajor"}, s_2 = \text{"Stayman"}, \) and \( s_3 = \text{"Transfer"}. \)

In Implementation 2, we let each object has two variations. In Figure 4 we show two variations of the 4 pass-objects. We note that all other \((120 - 4) = 116\) objects must also have two variations (to confuse attackers). A user specifies one pass-object, and remember its two variations as first and second variation. A password consists of the following: a set \( \{s_{1,1}, s_{1,2}, s_{2,1}, s_{2,2}, s_{3,1}, s_{3,2}\} \) of six strings and a one-to-one mapping

\[
m_2 : \{(\text{Case } k, \text{Variation } l) \mid k = 1, 2, 3; \ l = 1, 2\} \rightarrow \{s_{i,j} \mid i = 1, 2; \ j = 1, 2, 3\}
\]

**Implementation 2**

Step 1 Enter user-ID.
Step 2 Pass a series of three senses as follows: enter $s_{m_2(\text{Case } k, \text{Variatio } l)}$, if Case $k$, $k = 1, 2, 3$, is rendering on the screen, and the $l$th variation of the specified pass-object, $l = 1, 2$, is being displayed.

This implementation requires user training but it meets higher security than implementation 1.

The computer system carries out implementation 1 and 2 in the same way.

**System procedure**

Phase 1 Ask for user-ID.

Phase 2 Carry out Step 2 as follows: For $r = 1$ to 3, randomly and independently produce a scene of Case $i$, $1 \leq i \leq 3$, and let the user try to pass the scene.

Phase 3 Trigger an alert, if the user failed to pass Phase 2.

Phase 4 In case when a user requests changing of the password, ask the user to identify the password including the set of strings, the 4 pass-objects, and the mapping $m_1$ (or $m_2$).

### 2.2 Usability

The design of our proposed password scheme was a long process of working with people of different backgrounds from students to senior citizens. Our idea is using icons to generate random patterns to confuse a shoulder-surfing attacker; in the mean time, it should be easy for a user who owns the password to recognize “meanings” of these patterns. It soon became clear that icons ought be used just as marks whose contents should not be much used. This led us to using positions of some icons, which we call pass-objects, to form patterns. Our participants suggested that overall these patterns must be random, but to a user there should be only a few easy ones. This was how Case 2 and 3 (all pass-objects are on the left and right half) were taken, because they are easy to be recognized. For the sake of security, we need a third case. Why Case 1 was chosen was interesting. It seemed to us that a natural choice for a third case is the complement of Case 2 and 3, that is, 4-pass objects are on both of the left and right half. But many our participants treated the complement as two sub-cases: balanced - 2 of 4 pass-objects on each half, and unbalanced - 3 of 4 pass-objects on one half. And they felt difficult to combine the two sub-cases into one. They prefer that
with some visual help the two sub-cases can be easily viewed as one. We placed a eye-shape
icon at the center of a screen to see if this distinguishing mark could help. To our surprise,
using the distinguishing mark they combined the two sub-cases into one case which they
name “center”. None of us has expertise in psychology of human visual perception, and is
able to make scientific arguments for this choice. What we did is simply accept the choice
and put it in further experiments.

For Implementation 1, to participants we simply explained how to use it. Then we let
them choose and use their passwords to login. Our findings are as follows:

All participants from a variety of groups tended to choose the set \( \{s_1, s_2, s_3\} \) of three
strings such that one is associated to “center” as for Case 1, other two are associated to
“right” or “left” as for Case 2 or 3. How to associate “center”, ‘right” or “left” to Case
1, 2, or 3 varies from person to person. For example, one senior citizen fellow chose \( s_1 =
\text{“5CardMajor”}, s_2 = \text{“Stayman”}, \) and \( s_3 = \text{“Transfer”}. \) Since this lady is a good bridge
player, it is easy for her to remember: her password is about the standard American bidding
system. Five card major (“5CardMajor”) is the main feature of the bidding system, so it
is at the center. Stayman and Transfer are two frequently applied rules, so they are on
the right and left respectively. Though young users (students) had many different ways,
they all implicitly or explicitly associated \( s_1, s_2 \) and \( s_3 \) to “center”, ‘right” and “left”. Our
observation is: users do tend not to use contents rather positions of 4 pass-objects to set
their passwords. This is good for both the usability and security. Users only need to pick
up 4 pass-objects by their positions without looking at their contents in detail, which makes
Implementation 1 easy to use; in the mean time, since contents of all objects are not directly
relevant to passwords, an attacker will be forced to find 4 pass-objects only by their positions
which is difficult.

The total length of \( s_1, s_2 \) and \( s_3 \) are between 9 - 27 longer than that in the text-based
password scheme, between 6 - 20. In the example above, the total length of \( s_1, s_2 \) and \( s_3 \)
is 25. A short one is \( s_1 = \text{“Ted”}, s_2 = \text{“Kay”} \) and \( s_3 = \text{“Joe”} \) by a retired lady. Ted is her
husband’s name, Kay and Joe are names of her son and daughter. Since \( s_1, s_2 \) and \( s_3 \) are
individually entered with hints, “center”, ‘right” and “left”, by 4 pass-objects, users of all

8
groups are quite comfortable with Implementation 1.

For Implementation 2, in our further experiments, the senior citizen group refused to use it, simply because of its complexity. But young users (in particular, students) seemed not have difficulty to handle it. We set an one-hour training seminar in which we discussed with them how to associate

\[ s_{1,1}, s_{1,2} \text{ to ("center", variation 1), ("center", variation 2)} \]
\[ s_{2,1}, s_{2,2} \text{ to ("right", variation 1), ("right", variation 2)} \]
\[ s_{3,1}, s_{3,2} \text{ to ("left", variation 1), ("left", variation 2)} \]

One example is by a student majoring in computer science: \( s_{1,1} = \text{"Java"}, s_{1,2} = \text{"C++"}, s_{2,1} = \text{"MIS"}, s_{2,2} = \text{"PHP"}, \) and \( s_{3,1} = \text{"ITT"}, s_{3,2} = \text{"JSP"}. \) For the student, "center" is Java and C++ programming; right is MIS (management of information systems) and PHP; and left is ITT (information technology) and JSP.

We found that it was not difficult for young users to adopt Implementation 2, but it was hard for them to get used to it. For example, all participants were able to set and use their passwords for Implementation 2 after the one-hour training seminar, but in one week, in a follow-up seminar 53\% of them had hard time to recall how to use Implementation 2, though, in five minutes, they could regain the capability. In the second follow-up seminar 41\% had hard time; in the third, there was only 11\%; in the fourth, it decreased to 5\%. Our observation is: after special training young users are able to use Implementation 2, but need time to get used to it.

From the system perspective, Implementation 1 and 2 add only a light graphic layer to the existing text-based password scheme. As summary, in terms of the usability we compare Implementation 1 and 2 with the commercial software, PassFaces, and the text-based password scheme as follows:
<table>
<thead>
<tr>
<th></th>
<th>text-based</th>
<th>PassFaces</th>
<th>Implement 1</th>
<th>Implement 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of scenes</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>size of pwd</td>
<td>6 - 20</td>
<td>6 - 20</td>
<td>9 - 27</td>
<td>18 - 24</td>
</tr>
<tr>
<td>to user</td>
<td>easy</td>
<td>easy</td>
<td>easy</td>
<td>hard</td>
</tr>
<tr>
<td>to system</td>
<td>+ 1 graph. layer</td>
<td>+ 1 graph. layer</td>
<td>+ 1 graph. layer</td>
<td></td>
</tr>
</tbody>
</table>

2.3 Security

We analyze how Implementation 1 is resistant to a shoulder-surfing attack. From a record of such an attack, the attacker will have three scenes and three strings, \( s_{i_1}, s_{i_2}, s_{i_3}, 1 \leq i_1, i_2, i_3 \leq 3 \). When only when \( i_1, i_2, i_3 \) are different from each other, the attacker can have the set \( \{ s_1, s_2, s_3 \} \). Since the system randomly and independently produces three scenes, and since a scene is uniquely for a Case \( i \) which uniquely maps to a string \( s_i \) \( (i = 1, 2, 3) \), we have

\[
\Pr \{ \text{for attacker to obtaining the set } \{ s_1, s_2, s_3 \} \} = \frac{3!}{3^3} = \frac{2}{9}
\]

(1)

Now, let us assume that the attacker had the set \( \{ s_1, s_2, s_3 \} \). This also implies that the attacker has three scenes: at one scene the 4 pass-objects form a convex hull containing the center the screen; at other one the 4 pass-objects are all in the right half of the screen; at another one the 4 pass-objects are all in the left half of the screen. To retrieve the password, the attacker must identify the one-to-one mapping \( m_1 : \{ \text{Case 1, Case 2, Case 3} \} \mapsto \{ s_1, s_2, s_3 \} \). The attacker is supposed to know the algorithm for Implementation 1. That is, he knows the three cases are determined by positions of 4 pass-objects. As mentioned in the previous subsection, users tend not to use contents of 4 pass-objects when they create their passwords. Thus, the attacker is forced to use only positions of the 4 pass-objects to find them.

But any 4-object group may consist exactly of the 4 pass-objects, if at one scene the 4-objects form a convex hull containing the center the screen, at other one the 4-objects are all in the right half of the screen, and at another one the 4-objects are all in the left half of the screen.
At a scene there are 120 objects on the screen. In next section we will present an algorithm for generating scenes. This algorithm demonstrates that with a confidence level greater than $1 - 2e^{-54}$ we have that at each scene the 120 objects are randomly and independently positioned on the screen. In the sequel, we take this confidence level $> 1 - 2e^{-54}$ as 1. Thus, for one scene, in average there are $120 \div 2 = 60$ objects on each of right and left half of the screen. Therefore, for two scene, in average there are $60 \div 2 = 30$ objects that switch from one half to the another. In section 4 we will show that if we randomly and independently position 4 objects on the screen, then the probability of that the 4 objects form a convex hull containing the center of the screen is greater than $\frac{1}{3}$. Hence, we have

**Proposition 2.1** For the three scenes, in average, there are more than $C^4_{30} \div 3 = 9,135$ different 4-object groups each of which equally likely consists of the 4 pass-objects.

The average number of 4-object groups that the attacker will face does not fully characterizes the security. We need to show that with high probability the attacker will face at least five 4-object groups each of which equally likely consists of the 4 pass-objects. (We note that it needs at least 5 objects to have five 4-object groups.)

**Proposition 2.2** Given the three scenes, with probability greater than $1 - 1.259064 \times 10^{-3}$, the attacker will face at least five 4-object groups each of which equally likely consists of the 4 pass-objects.

**Proof:** We estimate the number of these 4-object groups. For each group, at one scene the 4-objects form a convex hull containing the center the screen, at other one the 4-objects are all in the right half of the screen, and at another one the 4-objects are all in the left half of the screen.

Since the three scenes were independently generated, we can estimate the number as follows: Fix an order of the three scenes. For the first scene, we estimate the number $n_1$ of objects that in the left half; For the second scene, we estimate the number $n_2$ of objects that were in the left half and now are in the right half. For the third scene, we estimate the number $n_3$ of 4-object groups which formed by these objects that were respectively in the left and right half in the first and second scene.
Let us consider 120 i.i.d random variables $X_i$ where $X_i = 0$ or 1 with equal probability $\frac{1}{2}$. Then we have $n_1 = \sum_{i=1}^{120} X_i$. By Hoeffding’s inequality [8], we have for all $s_1 > 0$

$$
\Pr \left\{ n_1 = \sum_{i=1}^{120} X_i \geq 120 \times \frac{1}{2} - s_1 \right\} \geq 1 - \exp \left( -\frac{2s_1^2}{120} \right)
$$

Letting $s_1 = 120 \times 0.18$ we have

$$
\Pr \{ n_1 \geq 39 \} \geq 1 - 4.19688 \times 10^{-4}
$$

Then by Hoeffding’s inequality we have for all $s_2 > 0$

$$
\Pr \left\{ n_2 \geq \sum_{i=1}^{39} X_i \geq 39 \times \frac{1}{2} - s_2 \left| \ n_1 \geq 39 \right. \right\} \geq 1 - \exp \left( -\frac{2s_2^2}{39} \right)
$$

Letting $s_2 = 39 \times 0.3157410887$ we have

$$
\Pr \{ n_2 \geq 8 \ | \ n_1 \geq 39 \} \geq 1 - 4.19688 \times 10^{-4}
$$

For 8 objects, there are $C_8^4 = 70$ 4-object groups. Let us consider 8 i.i.d random variables $Y_i$ where $Y_i = 0$ with probability $\frac{2}{3}$, 1 with probability $\frac{1}{3}$. Since the 8 objects are randomly and independently positioned on the screen and since with probability greater than $\frac{1}{3}$ each 4 of the 8 objects form a convex hull containing the center of the screen, we can set $n_3 = \sum_{i=1}^{70} Y_i$. By Hoeffding’s inequality we have for all $s_3 > 0$

$$
\Pr \left\{ n_3 \geq \sum_{i=1}^{70} Y_i \geq 70 \times \frac{1}{3} - s_3 \left| \ n_2 \geq 8 \left| \ n_1 \geq 39 \right. \right. \right\} \geq 1 - \exp \left( -\frac{2s_3^2}{70} \right)
$$

Letting $s_3 = 70 \times 0.235675$ we have

$$
\Pr \{ n_3 \geq 5 \ | \ n_2 \geq 8 \ | \ n_1 \geq 39 \} \geq 1 - 4.19688 \times 10^{-4}
$$

Combining (2), (3) and (4), with the fact that the three scenes are independently generated we have

$$
\Pr \{(n_3 \geq 5) \land (n_2 \geq 8) \land (n_1 \geq 39)\} \geq (1 - 4.19688 \times 10^{-4})^3 \geq 1 - 1.259064 \times 10^{-3}
$$

Proposition 2.1 and 2.2 together characterize the security of Implementation 1. It is difficult for attacker to retrieve a password by a shoulder-surfing attack, since it is hard to
find the 4 pass-objects even if the attacker was lucky to have the set \(\{s_1, s_2, s_3\}\). Indeed, the former proposition claims that in average, there are about \(C^4_{30} \div 3 = 9,135\) different 4-object groups each of which equally likely consists of the 4 pass-objects. If the attacker randomly chooses a 4-object group hopping it consists exactly the 4 pass-objects, then the probability for him to fail is \(1 - \frac{1}{9,135} \geq 1 - 1.1 \times 10^{-4}\). The later claims that the attacker cannot much improve this probability of failure by analyzing the three scenes: with a probability greater than \(1 - 1.259064 \times 10^{-3}\) the attacker cannot identify the 4 pass-objects, and will fail again.

**Proposition 2.3** For Implementation 1, with a record obtained by a shoulder-surfing attack, the attacker has only a probability less than \(2.8 \times 10^{-4}\) to successfully retrieve the password, and has a probability less than \(1.4 \times 10^{-3}\) to directly login without alerting the computer system.

**Proof:** By (1) with probability \(\frac{2}{9}\) the attacker can have the set \(\{s_1, s_2, s_3\}\); otherwise, he cannot retrieve the password at all. And by Proposition 2.2 we know that even with the set \(\{s_1, s_2, s_3\}\) the probability for the attacker to successfully retrieve the password is less than \(1.259064 \times 10^{-3}\). Thus, the probability for the attacker to succeed is less than \(\frac{2}{9} \times 1.259064 \times 10^{-3} \leq 2.8 \times 10^{-4}\).

If the attacker tries to directly login, then he will face the following. The record obtained by him contains three strings, \(s_{i_1}, s_{i_2}, s_{i_3}\). At each login, the system randomly and independently generates three scenes for Case 1, 2 and 3, each of which uniquely corresponds to a string. Hence, there are \(3^3 = 27\) different ways to generate three scenes for a login which possibly requires 27 different ways to enter three strings. \(s_{i_1}, s_{i_2}, s_{i_3}\) is a random one of the possible 27 ways. When the attacker tries to directly login, he is trying to hit a match of two random ones of the possible 27 ways. Thus, the probability for him to succeed is \(\frac{1}{27^2} \leq 1.4 \times 10^{-3}\). \(\square\)

We note that even if the attacker was lucky enough to login he will still have difficulty to change the password, since, at least, he needs to know what the 4 pass-objects are; but, as we showed in the proof above, the probability for the attacker to know the 4 pass-objects is less than \(2.8 \times 10^{-4}\).
Now, we analyze how Implementation 2 is resistant to a shoulder-surfing attack.

**Proposition 2.4** For Implementation 2, with a record obtained by a shoulder-surfing attack, the attacker has zero probability to successfully retrieve the password, and has a probability less than $2.2 \times 10^{-5}$ to directly login without alerting the computer system.

**Proof:** Recall that a password in this implementation has a set $\{s_{i,j} \mid i = 1, 2; \ j = 1, 2, 3\}$ of six strings. With a record of a shoulder-surfing attack, at most three of six strings were exposed. Thus, it is impossible for the attacker to retrieve the password with such a record. If the attacker tries to directly login, then by analysis similar in the proof of Proposition 2.3 we have the probability for him to succeed is $\frac{1}{(6^3)^2} \leq 2.2 \times 10^{-5}$. □

Subject to the shoulder-surfing attack, we compare the security of our proposed password scheme with the text-based scheme and PassFace. In the table below, the percentage in an entry indicates the probability of success of a shoulder-surfing attacker.

<table>
<thead>
<tr>
<th></th>
<th>text-based</th>
<th>PassFaces</th>
<th>Implement 1</th>
<th>Implement 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>get password</td>
<td>100%</td>
<td>100%</td>
<td>0.028%</td>
<td>0%</td>
</tr>
<tr>
<td>direct login</td>
<td>100%</td>
<td>100%</td>
<td>0.14%</td>
<td>0.0022%</td>
</tr>
</tbody>
</table>

It is not clear what we can say about the security if a shoulder-surfing attacker launched multiple attacks on a user. In this case, it is obvious that the attacker will observe all strings used a part of the password. Thus, the only defense is to prevent the attacker from finding the 4 pass-objects. This relies on how we really implement our proposed password scheme.

A complication to present our implementation is as follows: The theoretical foundation for generating Case 1 requires a solution to a problem in geometric probability which is closely related a classic problem in that field. Both problems are in continuous probability space. We decide to first present a randomized algorithm of how to generate Case 1, 2 and 3. This randomized algorithm is purely mathematical, but it introduces the problem in geometric probability. Then we present our implementation of the proposed password scheme as a variation of the randomized algorithm. We will carry out all these in next section. Here, we present the result a proof of which will be in subsection 3.3.
Theorem 2.5 Suppose that a shoulder-surfing attacker launched 480 times of attacks on a user. Then with a confidence level greater than 0.9 our implementation prevents the attacker from obtaining the password in both Implementation 1 and 2.

3 An algorithm for the scheme

3.1 The algorithm

We need an algorithm to carry out the following: (i) Case 1, 2 and 3 are generated with equal probability \( \frac{1}{3} \), and (ii) with a high confidence level the \( N = 120 \) objects randomly positioned on a screen. The technical challenge is that (i) and (ii) must be carried out at the same time. A question is: what is the probability of \( k = 4 \) randomly positioned objects forming a convex hull that contains the center of a screen? We need this probability for the design of such an algorithm. This question is related to a classic problem in geometric probability to which we will give an answer in section 4.

Since the shape of an object cannot be predetermined, we regard an object as a point, and consequently, a screen as a near-square rectangle. That is, the algorithm to be presented is purely mathematical, which can be used as a guideline for practical implementation. We will discuss how we implement this algorithm in subsection 3.3.

Let \( C \) be a near-square rectangle, and let \( N \) and \( k \) be two positive integers with \( N \gg k \geq 4 \). Consider \( N \) points. Among the \( N \) points, we need to specify \( k \) points for the positions of the \( k \) pass-objects. We want an algorithm that randomly positions \( N \) points in a \( C \) such that

(Q1) with probability \( \frac{1}{3} \) each of the following three events happens: Case 1. the convex hull of the \( k \) specified points contains the center of \( C \); Case 2. the \( k \) specified points are all in the right-half of \( C \); Case 3. the \( k \) specified points are all in the left-half of \( C \).

(Q2) with a high confidence level, the \( N \) points are uniformly and independently placed in \( C \).

We denote by \( p_k \left( \frac{a}{2}, \frac{b}{2} \right) \) the probability of the convex hull of \( k \) points, which are uniformly
and independently placed in $C$, contains the center of $C$. Here, $a$ and $b$ are respectively the width and height of $C$. In section 4 we will show that $p_k \left( \frac{a}{2}, \frac{b}{2} \right) = 1 - \frac{k}{2^n - 1}$. Thus, for $k \geq 4$ we can define

$$0 < \gamma_1 = \frac{1}{3} \cdot \frac{1}{p_k \left( \frac{a}{2}, \frac{b}{2} \right)} < 1$$

(5)

Then we consider a biased three-face dice $D$. This dice has face 1 with probability $\gamma_1$, and has face 2 and 3 with the same probability $\gamma_2$. Here, $\gamma_1 + 2\gamma_2 = 1$.

We describe the algorithm by a procedure below, which generates one scene. For a login, the algorithm calls the procedure three times.

**Generator:**

- uniformly and independently choose $N$ points in $C$;
- label the $N$ points by $v_1, \ldots, v_N$ according to the order in which they are chosen;
- toss the dice $D$;
- if face 1
  - call Generator 1;
- if face 2
  - call Generator 2;
- if face 3
  - call Generator 3;

**Generator 1:**

- pick $v_1, \ldots, v_k$ from the $N$ points;
- if the convex hull of the $k$ points contains the center of $C$
  - return the $k$ points as specified;
- else
  - call Spare 1;

**Generator 2:**

- if the number of points in the right-half of $C$ is no less than $k$
  - pick $k$ points with the $k$ smallest indices from the points in the right-half of $C$;
  - return the $k$ points as specified;
- else
  - call Spare 2;
Generator 3:
if the number of points in the left-half of $C$ is no less than $k$
    pick $k$ points with the $k$ smallest indices from the points in the left-half of $C$;
    return the $k$ points as specified;
else
    call Spare 3;

Spare 1:
toss a fair coin
if head
    call Generator 2;
if tail
    call Generator 3;

Spare 2:
erase the $N$ points currently in $C$;
uniformly and independently choose $k$ points in the right-half of $C$;
return the $k$ points as specified;
uniformly and independently choose $N - k$ points in $C$;

Spare 3:
erase the $N$ points currently in $C$;
uniformly and independently choose $k$ points in the left-half of $C$;
return the $k$ points as specified;
uniformly and independently choose $N - k$ points in $C$;

3.2 Soundness of the algorithm

We need to prove that the algorithm presented in the previous subsection does carry out (Q1) and (Q2) at the same time.

Lemma 3.1 Each of Case 1, 2 and 3 is generated with probability $\frac{1}{3}$.

Proof: Case 1 is generated only in Generator 1. The only caller of Generator 1 is Generator which uniformly and independently chooses $N$ points, and then tosses the dice $\mathcal{D}$.  

17
These two actions are independent on each other. Hence, any event from the former action is independent on any event from the later action. Let us consider the following two events: “the convex hull of the \( k \) specified points containing the center of \( C \)” (from the former action) and “face 1” (from the later action).

Since \( k \) specified points are uniformly and independently chosen in \( C \), the probability of “the convex hull of the \( k \) specified points containing the center of the rectangle” is \( p_k \left( \frac{3}{2}, \frac{1}{2} \right) \). And the probability of “face 1” is \( \gamma_1 \). Therefore, the probability of the intersection of the two events above is \( p_k \left( \frac{3}{2}, \frac{1}{2} \right) \times \gamma_1 = \frac{1}{3} \) by (5), that is, Case 1 is generated with probability \( \frac{1}{3} \).

Case 2 is generated only in \textbf{Generator 2}. There are two sources that call \textbf{Generator 2}: \textbf{Generator} and \textbf{Spare 1}. We note that the two sources call \textbf{Generator 2} in two separate events. Thus, the probability of Case 2 being generated is the sum of the probabilities of the former and the later sources calling \textbf{Generator 2}. The probability of the former source (\textbf{Generator}) calling \textbf{Generator 2} is \( \gamma_2 = \frac{1-n}{2} \).

The later source is \textbf{Spare 1} which itself is called only in \textbf{Generator 1} when Case 1 is not generated. The probability of \textbf{Generator 1} being called is \( \gamma_1 \). As we analyzed above, the probability of Case 1 being generated is \( \frac{1}{3} \). Thus, the probability of \textbf{Spare 1} being called is \( \gamma_1 - \frac{1}{3} \). In \textbf{Spare 1}, \textbf{Generator 2} is called with probability \( \frac{1}{2} \). Therefore, the probability of the later source (\textbf{Spare 1}) calling \textbf{Generator 2} is \( \frac{1}{2} \cdot (\gamma_1 - \frac{1}{3}) \).

Finally, we have the probability of Case 2 being generated is \( \frac{1-n}{2} + \frac{1}{2} \cdot (\gamma_1 - \frac{1}{3}) = \frac{1}{3} \).

With analysis similar to what for Case 2 we have the probability of Case 3 being generated is \( \frac{1}{3} \). \( \square \)

The lemma above shows that the algorithm does carry out (Q1). Now, we prove that it also carries out (Q2) at the same time. We note that at the beginning of the algorithm, Generator uniformly and independently chooses \( N \) points in \( C \). These \( N \) points will be kept unless either Spare 2 or Spare 3 are called. Thus, all we need to prove is that the probability of either Spare 2 or Spare 3 being called is small.

\textbf{Lemma 3.2} \textit{The probability of \textbf{Spare 2} being called is less than } \exp \left( -\frac{(N-2k+2)^2}{2N} \right), \text{ and so is the probability of \textbf{Spare 3} being called. Consequently, the probability of \textbf{Spare 2} or \textbf{3} being called is less than } 2 \exp \left( -\frac{(N-2k+2)^2}{2N} \right). \text{ 18}
Proof: We consider the probability of **Spare 2** being called. The probability of **Spare 3** being called can be obtained in the same way. **Spare 2** is called only by **Generator 2** which calls **Spare 2** only when the following event happens: Choosing \( N \) points uniformly and independently in \( C \), the number of points in the right-half of \( C \) is less than \( k \). Thus, the probability of **Spare 2** being called is less than the probability of “choosing \( N \) points uniformly and independently in \( C \), the number of points in the right-half of \( C \) is less than \( k \”).

Let us consider \( N \) i.i.d random variables \( X_i \) where \( X_i = 0 \) or 1 with equal probability \( \frac{1}{2} \). Then the number of points in the right-half of \( C \) is \( \sum_{i=1}^{N} X_i \). We have for all \( 1 \leq k \leq \frac{N}{2} \)

\[
\Pr \left\{ \sum_{i=1}^{N} X_i \leq k - 1 \right\} = \Pr \left\{ \frac{1}{2} \cdot N - \sum_{i=1}^{N} X_i \geq \frac{1}{2} \cdot N - (k - 1) \right\} \\
= \Pr \left\{ N \cdot E[X] - \sum_{i=1}^{N} X_i \geq \frac{N - 2k + 2}{2} \right\} \quad \text{since } E[X] = \frac{1}{2} \\
\leq \exp \left( -\frac{(N - 2k + 2)^2}{2N} \right) \quad \text{by Hoeffding’s inequality}
\]

that is, the probability of **Spare 2** being called is less than \( \exp \left( -\frac{(N - 2k + 2)^2}{2N} \right) \). \( \square \)

Letting \( N = 120 \) and \( k = 4 \) we have \( \exp \left( -\frac{(N - 2k + 2)^2}{2N} \right) < e^{-54} \). That is, the algorithm carries out (Q2) at a confidence level greater than \( 1 - 2e^{-54} \).

### 3.3 Implementation of the algorithm

The algorithm presented in subsection 3.1 is a randomized mathematical algorithm. Though it cannot be implemented by computer, the algorithm provides a guideline for its practical implementation on computer.

The key idea of the randomized algorithm is to generate random patterns of \( N \) objects so that among these patterns there are some objects behave in the same way as the \( k \) pass-objects do. And the security of our proposed password scheme relies on this fact. Indeed, in the proof of Proposition 2.2, combining (2), (3) and (4) one can conclude that with high probability there is at least one object, which is not pass-object but behaves as if it is a pass-object.
In the randomized algorithm an object is treated as a point with no size. We must count the size of an icon used as an object in practical implementation. Moreover, in the randomized algorithm with $N$ (finite number) points one can generate infinitely many patterns, but in a practical implementation with $N$ icons we can generate only finite number of patterns. This typically is a technique challenge when we implement a randomized mathematical algorithm. There are several options. We choose an implementation which bears the key idea of the randomized algorithm, and is also easy for users.

In our implementation, we line up the 120 objects and place an eye-shaped icon at the center as shown in Figure 1, 2 and 3 (see these large pictures in section 5 at the end of this paper). That is, we place 121 objects (icons) in a $11 \times 11$ matrix. There clearly are 55 objects in the left and right half, respectively. As for the 10 objects in the central column, we count the top five objects as in the left half and the bottom five as in the right half. Also, we tell users to do so. Therefore, we have 60 objects respectively in the left and right half, and the eye-shaped icon is always at the center. The question is how to randomly position the $2 \times 60 = 120$ objects so that Case 1, 2 and 3 are generated with equal probability $\frac{1}{3}$ and, in the mean time, there are some objects behave in the same way as the 4 pass-objects.

We choose 16 non pass-objects called companions. At each login, 10 non pass-objects other than the 16 companions are randomly chosen. We have total 30 objects: 4 pass-objects, 16 companions, and 10 non pass-objects. Then we respectively implement **Generator 2** and **3** as follows. Randomly distribute the 30 objects in the right (respectively, left) half, then randomly choose and place another 30 objects in the same half, and finally randomly place the remaining 60 objects in the other half. It is easy to see that in such an implementation, **Spare 2** and **3** will never be called. This implies that our implementation actually promotes the confidence level $1 - 2e^{-54}$ to 1. But, the question now is how to implement **Generator 1**.

Recall that in the randomized algorithm **Generator 1** is called with probability $\gamma_1 = \frac{1}{3} \cdot \frac{1}{p_k (\frac{3}{2}, 2)}$ (see (5)). As shown in the proof of Lemma 3.1, the soundness of the randomized algorithm relies on the existence of $\gamma_1$ which came from $p_k (\frac{4}{2}, 2) > \frac{1}{3}$. In section 4 Theorem 4.1 provides a solution $p_k (\frac{4}{2}, 2) = 1 - \frac{k}{2e^2}$. When $k = 4$ we have $p_4 (\frac{4}{2}, 2) = \frac{1}{2} > \frac{1}{3}$. This
solution is theoretical, since it holds under a condition that objects are treated as points. In our implementation, objects are icons with sizes. And we place \( N = 121 \) objects (including the eye-shaped mark) on a screen. If we let \( N \) go to infinity then our implementation will meet the condition in which objects can be treated as points. Thus, our implementation should have an approximation of \( p_4 \left( \frac{a}{2}, \frac{b}{2} \right) \). Now, if we closely look at the formulas (7), (8) and (9) in Theorem 4.1, then we can see that \( p_4 \left( \frac{a}{2}, \frac{b}{2} \right) \) is a continuous function when we move \( \left( \frac{a}{2}, \frac{b}{2} \right) \) away from the center. This continuity indicates that the approximation will not be bad. To find the approximation, using a computer program we directly calculate \( p_4 \left( \frac{a}{2}, \frac{b}{2} \right) \). Assuming that each entry of the \( 11 \times 11 \) matrix is a square, we obtain \( p_4' \left( \frac{a}{2}, \frac{b}{2} \right) \approx 0.35 \).

In the randomized algorithm we use \( p_4 \left( \frac{a}{2}, \frac{b}{2} \right) \) to obtain \( \gamma_1 \) with which we call Generator 1. \( \gamma_1 \) was necessary since we have infinitely many of 4 points whose convex hulls contain the center. In our implementation 4 entries of the \( 11 \times 11 \) matrix will be chosen. The question is how we use \( p_4' \left( \frac{a}{2}, \frac{b}{2} \right) \) to choose these 4 entries. When we calculate the approximation, using an index table we also have obtained the set of all 4 entries which form convex hulls that contain the center. There are totally \( p_4' \left( \frac{a}{2}, \frac{b}{2} \right) \cdot C^4_{120} \approx 2,875,000 \) of such 4 entries. Using the index table, we randomly choose one of the 2,875,000. In the mean time, we randomly divide the 16 companions into 4 groups (each with 4 objects), and for each group we randomly choose, without repetition, one of the 2,875,000. There are \( 120 - 20 = 100 \) non-pass objects remaining. We randomly place them in entries that are not occupied. Based upon what described above we set \( \gamma_1 \) as \( \frac{1}{3} \).

In summary, in our implementation of the randomized algorithm we have

- Case 1, 2 and 3 are generated randomly and independently with equal probability \( \frac{1}{3} \).
- In each case, there are 20 objects, 4 pass-objects and 16 companions, behave exactly in the same way.
- By straightforward calculation we have the following: for Case 1 there are \( P^5_r \times (4!)^5 \times 100! \) different patterns that can be randomly displayed, where \( P^5_r = r(r-1)(r-2)(r-3)(r-4) \) and \( r = \left[ 0.35 \times C^4_{120} \right] \); and for Case 2 as well as for Case 3 there are \( 60! \times 60! \) different patterns that can be randomly displayed.

**Proof of Theorem 2.5**: We note that for each of Case 1, 2 and 3 there are huge number of
different patterns that can randomly displayed. In both Implementation 1 and 2, for a user there are always 20 objects, 4 pass-objects and 16 companions, behave exactly in the same way. Thus, with one attack the best that a shoulder-surfing attacker can get is to identify the 20 objects.

There are $C^4_{20} = 4845$ different groups of 4-objects that can come out 20 objects. Each of these groups equally likely consists of the 4 pass-objects. Thus, after one attack, the probability for the attacker to successfully identify the group that consists of the 4 pass-objects is $\frac{1}{4845}$.

Now, suppose that the attacker attacks $t$ times. Recall that at each login, the system randomly and independently generates Case 1, 2 and 3. Thus, after $t$ times of attacks, the probability for the attacker to fail in identifying the group that consists of the 4 pass-objects is $(1 - \frac{1}{4845})^t$. That is, with a confidence level $(1 - \frac{1}{4845})^t$ our implementation prevents the attacker to identify the group that consists of the 4 pass-objects.

Letting $t = 480$ we have $(1 - \frac{1}{4845})^{480} > 1 - \frac{480}{4845} > 0.9$. The Theorem follows $\square$

It is easy to see the following: our implementation keeps Proposition 2.4 and 2.3 hold except in the later the probability for a shoulder-surfing attacker to successfully retrieve the password is reduced from $0.28 \times 10^{-4}$ to $\frac{1}{4845} \approx 2.1 \times 10^{-4}$.

Our implementation is just one of possible realizations of the randomized algorithm. There are many others. From user and systems perspective, our implementation has some advantages. Users are confortable with the fact that all objects are lined up. As for systems, 120 objects can serve 30 users with 4 objects per a user as pass-objects. In Implementation 1 the 120 objects without variations need 10MB of memory; in Implementation 2 the 120 objects with variations need 20MB. Suppose that we experiment Implementation 1 in public. Then for each user only $(10 ÷ 30) < 0.3$MB of extra memory is needed. This to today’s computer system is not heavy. An disadvantage of our implementation is that rendering 120 objects takes time. For each login, the system renders three scenes. Using a standard buffering technique, we can well speed up rendering of the second and third scenes. But the first one is a bit slow.
4 A problem in geometric probability and its solution

We need to solve the following problem.

**Problem 1** Let $C$ be a rectangle in $\mathbb{R}^2$ whose width and height are respectively $a$ and $b$. Consider $k$ uniformly and independently chosen points in $C$. Find the probability $p_k \left( \frac{a}{2}, \frac{b}{2} \right)$ of the center of $C$ contained in the convex hull of the $k$ points.

We consider a generalized version of Problem 1 as follows:

**Problem 2** Let $C$ be a convex body with a piece-wise smooth boundary in $\mathbb{R}^2$, and let $v_0 = (x_0, y_0)$ be a given point inside of $C$. Let $v_i = (x_i, y_i)$, $1 \leq i \leq k$, be $k$ uniformly and independently chosen points in $C$. Find the probability $p_k(x_0, y_0)$ of $v_0$ contained in the convex hull of $v_1, \ldots, v_k$.

Problem 2 is closely related to a classic problem in geometric probability. This classic problem was formally addressed by Rényi and Sulanke in 1963 [19]. Here, we state a generalized version of the problem by Rogers [18]:

**Problem 3** Given an absolutely continuous probability distribution on $\mathbb{R}^2$, we choose $l$ and $k$ points independently according to the distribution. We have two convex hulls of the $l$ and $k$ points respectively. Find the probability $p_{l,k}$ of the two convex hulls being disjoint.

There are a number of articles published on this topic (cf. [7], [2], [3] and references therein). All existing results on $p_{l,k}$ focused on its integral or asymptotic representations. Problem 2 is closely related to Problem 3. Taking $l = 1$, Problem 3 asks for the probability $p_{1,k}$ of a randomly chosen point $v_0 = (x_0, y_0)$ not contained in the convex hull of $k$ randomly independently chosen points; while, Problem 2 asks for the probability for the complement of $v_0$ not contained in the convex hull when $v_0$ is given. Thus, Problem 2 asks for the density, i.e., the Radon-Nikodym derivative of $p_{1,k}$ with respect to a uniform distribution on a bounded body in the plane (cf. [4]).

[18] did not provide a solution to Problem 2. Moreover, it was mentioned that Cartesian coordinates are not natural to express $p_{1,k}$. A key technique used in [18] was to have a proper geometric transformation. This approach is influential. Using a well designed transformation (which was carried along with so-called equiaffine inner parallel curves) Buchta and Reitzner [3] derived a series expression for $p_{1,k}$ when $C$ is a polygon. Finding a Radon-Nikodym
derivative is usually not trivial. Furthermore, for our applications we need to express the derivative in Cartesian coordinates. Hence, we need to work out a solution for Problem 2.

We now introduce notations which are all illustrated in Figure 5 in section 5. (To have large pictures we place section 5 at the end of this paper.) Fix a point \( v_0(x_0, y_0) \). We consider vector crossing \( v_0 \) and having its two end points on the boundary of \( C \). See Figure 5 below.

The range of the angle \( t \) from the \( X \) axis anti-clockwisely to such a vector is restricted to \([0, 2\pi)\). For such a vector we denote by \( r_1(v_0, t) \) the segment between \( v_0 \) and the end point along the direction of the vector, and by \( r_2(v_0, t) \) the segment between \( v_0 \) and the end point along the opposite direction of the vector. With harmless notation abuse we will use \( r_i(v_0, t) \), \( i = 1, 2 \), for the length of the segment. We denote the vector by \( \langle r_1(v_0, t), r_2(v_0, t) \rangle \). And we let \( s(v_0, t) \) denote the area of the region in \( C \) which is on the right side of the vector. We denote by \( \{v_0 \text{ not in } | C, k\} \) the event in which the given point \( v_0(x_0, y_0) \) is not in the convex hull of \( k \) random points, and by \( \{v_0 \text{ in } | C, k\} \) its complement. We use area[\cdot] for the area of a plane body.

**Theorem 4.1** For \( v_0(x_0, y_0) \) in \( C \) we have

\[
1 - p_k(x_0, y_0) = \Pr\{v_0 \text{ not in } | C, k\} = \frac{k}{2 \cdot \text{area}[C]^k} \int_0^{2\pi} r_1(v_0, t)^2 s(v_0, t)^{k-1} dt \tag{6}
\]

\[
= \frac{k}{2 \cdot \text{area}[C]^k} \int_0^{2\pi} r_2(v_0, t)^2 s(v_0, t)^{k-1} dt \tag{7}
\]

\[
= \frac{k}{4 \cdot \text{area}[C]^k} \int_0^{2\pi} (r_1(v_0, t)^2 + r_2(v_0, t)^2) s(v_0, t)^{k-1} dt. \tag{9}
\]

A proof of this theorem is presented in Appendix. By anyone of (7), or (8) or (9) with a starightfoeward calculation we have \( p_k \left( \frac{a}{2}, \frac{b}{2} \right) = 1 - \frac{k}{2^{2k-1}} \) where \( C \) is a rectangle and \( \left( \frac{a}{2}, \frac{b}{2} \right) \) is it center.

**Acknowledgments**

We thank all people who participated in our experiments on Implementation 1 and 2. In particular, their comments on how to set the interface were invaluable.
References


Figure 1: Case 1
Figure 2: Case 2
Figure 3: Case 3
Figure 4: Two Variations

Figure 5: Notations
Appendix: a proof for Theorem 4.1

We shall use word “in” for a point being an interior point of a convex body. We need two propositions. The first one follows directly from real analysis.

**Proposition 5.1** $r_1(v_0, t), r_2(v_0, t)$ and $s(v_0, t)$ are uniformly continuous in $\{v_0 : v_0 \in C\} \times [0, 2\pi)$.

![Diagram](image)

Figure 6: Proof

We observe that $v_0$ is not in the convex hull of $k$ points $v_1, \ldots, v_k$ when and only when there is a cone such that (i) it has $v_0$ as its vertex and has its angle not greater than $\pi$, and (ii) it contains all the $k$ points. We take the minimum cone satisfying (i) and (ii). See Part 1 of Figure 6. We treat the two edges of the minimum cone as two vectors in such a way that all $v_1, \ldots, v_k$ locate on the right sides of the two vectors. Note that each of the two vectors crosses $v_0$ and has its two endpoints on the boundary of $C$. The angles $t_1$ and $t_2$ anti-clockwise from the X axis respectively to the two vectors are restricted to $[0, 2\pi)$. We denote the two vectors by $\langle r_1(v_0, t_1), r_2(v_0, t_1) \rangle$ and $\langle r_1(v_0, t_2), r_2(v_0, t_2) \rangle$. We have
$0 \leq t_1 < t_2 < 2\pi$. For any fixed $v_0$, $\langle r_1(v_0, t_1), r_2(v_0, t_1) \rangle$ (respectively, $\langle r_1(v_0, t_2), r_2(v_0, t_2) \rangle$) uniquely determines a set of $k$ randomly and independently chosen points $v_1, \ldots, v_k$ such that the convex hull of the $k$ points does not contain $v_0$ in it. Since $k$ points $v_1, \ldots, v_k$ are randomly and independently chosen, we can write them as a vector $(v_1, \ldots, v_k)$. That is, we view $v_1, \ldots, v_k$ are chosen according to a product probability measure in $\mathbb{R}^k$.

We let

$$H_1(t) = \{(v_1, \ldots, v_k) \in C^k : t_1 = t \text{ for } \langle r_1(v_0, t_1), r_2(v_0, t_1) \rangle\}$$

and let

$$H_2(t) = \{(v_1, \ldots, v_k) \in C^k : t_2 = t \text{ for } \langle r_1(v_0, t_2), r_2(v_0, t_2) \rangle\}$$

It is straightforward to check out that

$$\{v_0 \text{ not in } | C, k\} = \bigcup_{0 \leq t < 2\pi} H_1(t) = \bigcup_{0 \leq t < 2\pi} H_2(t)$$

and that for all $0 \leq t < t' < 2\pi$

$$H_i(t) \cap H_i(t') = \emptyset, \ i = 1, 2.$$

Hence, we have

**Proposition 5.2** $(10)$ gives two partitions for $\{v_0 \text{ not in } | C, k\}$. 

We now present a proof for Theorem 4.1 which follows a standard procedure in real analysis, using the outer and inner measures (cf. p. 421 of [4]).

**Proof of Theorem 4.1**: By Proposition 5.2 we have that $\bigcup_{0 \leq t < 2\pi} H_1(t)$ is a partition of $\{v_0 \text{ not in } | C, k\}$. We note that this partition is uncountably infinite with $\Pr \{H_1(t)\} = 0$ for all $0 \leq t < 2\pi$. Take a large positive integer $m$. We let $a_0 = 0$ and $a_j = \frac{2\pi j}{m}$, $j = 1, \ldots, m$. Then we write

$$\{v_0 \text{ not in } | C, k\} = \bigcup_{1 \leq j \leq m} \bigcup_{a_{j-1} \leq t < a_j} H_1(t)$$

We have two vectors, $\langle r_1(v_0, a_{j-1}), r_2(v_0, a_{j-1}) \rangle$ and $\langle r_1(v_0, a_j), r_2(v_0, a_j) \rangle$, for each $j$, $1 \leq j \leq m$. See Part 2 of Figure 6. $r_1(v_0, a_{j-1}), r_2(v_0, a_{j-1}), r_1(v_0, a_j)$ and $r_2(v_0, a_j)$ form several cones in $C$. We denote by cone($\cdot$, $\cdot$) (or, cone($\cdot$, $\cdot$)) each of these cones, and read it anti-clockwisely. For example, the region $A$, which is formed from $r_1(v_0, a_{j-1})$ anti-clockwisely to
$r_1(v_0, a_j)$ in Part 2 of Figure 6, is denoted by $\text{cone}[r_1(v_0, a_{j-1}), r_1(v_0, a_j)]$. The close left-end [ means the cone includes $r_1(v_0, a_{j-1})$; the open right-end ) means the cone excludes $r_1(v_0, a_j)$. Also, for $\text{cone}[, \cdot]$ (or, $\text{cone}(:, \cdot)$) we denote its area by $\text{area}[\text{cone}[, \cdot]]$ (or, $\text{area}[\text{cone}(:, \cdot)]$).

For each $j$, $1 \leq j \leq m$, we define a subset $L_s(j)$ of $C^k$:

$$\{(v_1, \ldots, v_k) : \exists v_i \in \text{cone}[r_1(v_0, a_{j-1}), r_1(v_0, a_j)] \land \forall v_i \in \text{cone}(r_2(v_0, a_j), r_1(v_0, a_j))\}$$

It is straightforward to check out that $L_s(j)$, $j = 1, \ldots, m$, have the following three properties:

1. $L_s(j) \subset \cup_{a_{j-1} \leq t < a_j} H_1(t)$ for all $j = 1, \ldots, m$.
2. $\Pr \{L_s(j)\} = \frac{k}{\text{area}[C]^k} \cdot \text{area}[\text{cone}[r_1(v_0, a_{j-1}), r_1(v_0, a_j)] \cdot s(v_0, a_j)^{k-1}}$
3. $L_s(j_1) \cap L_s(j_2) = \emptyset$ for all $1 \leq j_1 < j_2 \leq m$.

Then we have

$$\Pr \{v_0 \text{ not in } | C, k\} = \Pr \{\cup_{1 \leq j \leq m} \cup_{a_{j-1} \leq t < a_j} H_1(t)\} \text{ by (11)}$$

$$\geq \Pr \{\cup_{1 \leq j \leq m} L_s(j)\} \text{ by property (1.1)}$$

$$= \sum_{1 \leq j \leq m} \Pr \{L_s(j)\} \text{ by property (1.3)}$$

With this and property (1.2) we have

$$\Pr \{v_0 \text{ not in } | C, k\} \geq \frac{k}{\text{area}[C]^k} \sum_{1 \leq j \leq m} \text{area}[\text{cone}[r_1(v_0, a_{j-1}), r_1(v_0, a_j)] \cdot s(v_0, a_j)^{k-1}} \text{ (12)}$$

We show that, as $m$ goes to infinity, the right side of (12) has a limit equal to

$$\frac{k}{2 \cdot \text{area}[C]^k} \int_0^{2\pi} r_1(v_0, t)^2 s(v_0, t)^{k-1} dt$$

Indeed, since $C$ has a piece-wise smooth boundary, we can have a sequence $\{\epsilon_m\}_{m \geq 1}$, in which $1 > \epsilon_m > 0$ for all $m \geq 1$ and depend only on $v_0$ and $C$, such that $\lim_{m \to \infty} \epsilon_m = 0$, and such that for every large enough $m$ and all $j = 1, \ldots, m$,

$$1 - \epsilon_m \leq \frac{\text{area}[\text{cone}[r_1(v_0, a_{j-1}), r_1(v_0, a_j)]]}{\frac{1}{2} r_1(v_0, a_j) r_1(v_0, a_{j-1}) \sin(a_j - a_{j-1})} \leq 1 + \epsilon_m$$
Hence, for all large enough $m$ the right side of (12) is within
\[
(1 \pm \epsilon_m) \left\{ \frac{k}{2 \cdot \text{area}[C]^k} \sum_{1 \leq j \leq m} r_1(v_0, a_j) r_1(v_0, a_{j-1}) \sin (a_j - a_{j-1}) \cdot s(v_0, a_j)^{k-1} \right\}
\]
By Proposition 5.1 we have
\[
\lim_{m \to \infty} \sum_{1 \leq j \leq m} r_1(v_0, a_j) r_1(v_0, a_{j-1}) \sin (a_j - a_{j-1}) \cdot s(v_0, a_j)^{k-1} = \int_0^{2\pi} r_1(v_0, t)^2 s(v_0, t)^{k-1} dt
\]
which proves that, as $m$ goes to infinity, the right side of (12) does have the limit as claimed above.

Now, letting $m \to \infty$ in (12) we have
\[
\Pr \{v_0 \text{ not in } | C, k\} \leq \frac{k}{2 \cdot \text{area}[C]^k} \int_0^{2\pi} r_1(v_0, t)^2 s(v_0, t)^{k-1} dt \tag{13}
\]
The proof for the other direction of the inequality (13) is parallel. We define for each $j$, $1 \leq j \leq m$, a subset $L^*(j)$ of $C^k$:
\[
\{(v_1, \ldots, v_k) : \exists v_i \in \text{cone}[r_1(v_0, a_{j-1}), r_1(v_0, a_j)] \land \forall v_i \in \text{cone}[r_2(v_0, a_{j-1}), r_1(v_0, a_j)]\}
\]
It is straightforward to check out that $L^*(j)$, $j = 1, \ldots, m$, have the following two properties:

(2.1) $L^*(j) \supset \cup_{a_{j-1} \leq t \leq a_j} H_1(t)$ for all $j = 1, \ldots, m$.

(2.2) For all $j = 1, \ldots, m$
\[
\Pr \{L^*(j)\} = \frac{k}{\text{area}[C]^k} \cdot \text{area}[\text{cone}[r_1(v_0, a_{j-1}), r_1(v_0, a_j)]] \cdot Q(v_0, a_{j-1}, a_j)^{k-1}
\]
where $Q(v_0, a_{j-1}, a_j) = s(v_0, a_j) + \text{area}[\text{cone}[r_2(v_0, a_{j-1}), r_2(v_0, a_j)]]$.

Then we have
\[
\Pr \{v_0 \text{ not in } | C, k\} = \Pr \{\cup_{1 \leq j \leq m} \cup_{a_{j-1} \leq t \leq a_j} H_1(t)\} \text{ by (11)} \leq \Pr \{\cup_{1 \leq j \leq m} L^*(j)\} \text{ by property (2.1)} \leq \sum_{1 \leq j \leq m} \Pr \{L^*(j)\}
\]
With this and property (2.2) we have
\[
\Pr \{v_0 \text{ not in } | C, k\} \leq \frac{k}{\text{area}[C]^k} \sum_{1 \leq j \leq m} \text{area}[\text{cone}[r_1(v_0, a_{j-1}), r_1(v_0, a_j)]] \cdot Q(v_0, a_{j-1}, a_j)^{k-1}
\]
34
With Proposition 5.1 we have that, as \( m \to \infty \), the right side of the inequality above has a limit equal to 

\[
\frac{k}{2 \cdot \text{area}[C]} \int_0^{2\pi} r_1(v_0, t)^2 s(v_0, t)^{k-1} dt.
\]

Now, letting \( m \to \infty \) in the inequality above we have

\[
\Pr\{v_0 \text{ not in } | C, k \} \leq \frac{k}{2 \cdot \text{area}[C]} \int_0^{2\pi} r_1(v_0, t)^2 s(v_0, t)^{k-1} dt
\]

(7) of the theorem follows from this and (13).

By Proposition 5.1 and calculus we can have

\[
\int_0^{2\pi} k \cdot \frac{r_1(v_0, t)^2 - r_2(v_0, t)^2}{2} \cdot s(v_0, t)^{k-1} dt = \int_0^{2\pi} ds(v_0, t)^k = \lim_{t \to 2\pi} s(v_0, t)^k - s(v_0, 0)^k = 0
\]

(8) follow from this and (7). And (9) follows directly from (7) and (8). \( \square \)