On the Security of Graphical Password Schemes [1]

P.C. van Oorschot, Julie Thorpe
School of Computer Science, Carleton University, Canada

Abstract

In commonplace textual password schemes, users typically choose passwords that are easy to recall, exhibit patterns, and are thus vulnerable to brute-force dictionary attacks. This leads us to ask what classes of *graphical* passwords users tend to choose because they are more memorable, with particular focus on the "Draw-A-Secret" (DAS) graphical password scheme of Jermyn et al. (1999). We postulate a set of such classes based on *password complexity factors* (e.g., reflective symmetry and stroke-count), supported by a collection of cognitive studies on visual recall. We suggest that an attacker would prioritize an attack dictionary for graphical passwords based on these classes. We analyze the size of these classes for DAS (under reasonable parameter choices), showing their combined bit-size ranges from 31 to 41 – a surprisingly tiny proportion of the full password space (58 bits). Our results suggest that DAS (and other graphical password schemes) may well be less secure than previously believed, unless measures such as password rules are employed. For a given security level in DAS, this translates into a requirement for longer passwords with a higher stroke-count than previously believed. Finally, we examine methods to decrease susceptibility to graphical dictionary attacks. Our results have implications beyond DAS, to graphical password schemes in general; they can be directly applied to graphical password guidelines, proactive graphical password checking, and in the design of graphical password user studies.

1. INTRODUCTION

The ubiquitous use of textual passwords for user authentication has a well-known weakness: users tend to choose passwords with predictable characteristics, related to how easy they are to remember. This often means passwords which have "meaning" to the user. Unfortunately, many of these "higher probability" passwords fall into a tiny subset of the full password space. Although its boundaries vary depending on its exact definition and the probabilities involved, we refer to this smaller subset as the *probable password space*.

Ideally, users would choose passwords equi-probably from a large subset of the overall password space, to increase the cost of a *dictionary attack*, i.e., a brute-force guessing attack involving candidate guesses from a prioritized list of "likely passwords". If a password scheme's probability distribution is non-uniform, its entropy is reduced. In Klein's case study [Klein 1990], 25% of 14000 user passwords were found in a dictionary of only 3×10^6 words; the Morris Worm [Spafford 1989] used a dictionary of only 432 words in addition to the 1988 UNIX online dictionary (about 25000 words [Spafford 1992]) with remarkable success: some sites reported that 50% of passwords were correctly guessed. This suggests that a password scheme's security is linked more closely to the size of the probable password space than that of the full password space (which, e.g., for 8-character passwords of digits and mixed-case letters, is about 2×10^{14}).

Graphical password schemes (e.g., [Jermyn et al. 1999; Wiedenbeck et al. 2005; Dhamija and Perrig 2000; Real User Corporation 2004]) require users to remember picture-based information instead of text, motivated in part by the fact that humans have a remarkable capability to remember pictures. If the number of possible pictures is sufficiently large, and the diversity of picture-based passwords can be captured, the probable password space may exceed that of text-based schemes and offer better security. What remains to be shown is what sort of pictures people are likely to *select* as graphical passwords. We explore this issue in the present paper.

We explore the idea of a probable password space (defined in §3), motivated by the questions: (1) How might an attacker build a *graphical dictionary*? (i.e., an attack dictionary against a graphical password scheme); and (2) How successful would a brute-force attack using such a dictionary be? As mentioned, the high success rate of brute-force dictionary attacks against textual passwords is believed to be strongly related

[1] Version: December 19, 2005. Preliminary versions of parts of this paper appeared as [Thorpe and van Oorschot 2004a] and [Thorpe and van Oorschot 2004b]

to the recall capabilities of humans and how this affects password selection: meaningful and thus more easily remembered strings are frequently chosen as passwords. We suggest that a clever attacker would narrow down the password space (and prioritize guesses) to pictures or picture elements that people are more likely to choose as passwords, based on being easier to recall or recognize.

Understanding how certain password characteristics affect the probable password space is an important step in understanding the security graphical passwords provide. We define a password *complexity property* to be a property that affects password memorability (and by conjecture, the chance of selection). We would like to identify such properties, and to know how a significant proportion of users choosing passwords with these properties would impact the size of the probable password space. To find complexity properties that an attacker might use in building a graphical dictionary, we consult psychological studies on visual memory. Since our focus is on recall-based graphical password schemes, we review cognitive studies indicating the types of images people are most likely to recall.

Motivated by pattern complexity factors from Attneave [Attneave 1955a], we introduce a set of graphical password complexity properties: password length, number of components (i.e., the visually distinct parts of the graphical password), and symmetry. We introduce classes of probable graphical passwords based on these complexity properties.

The "Draw-A-Secret" (DAS) scheme [Jermyn et al. 1999] is of particular interest as it claims a large password space, implicitly suggesting a large probable password space. We apply two of our aforementioned classes of probable graphical passwords to DAS (individually and combined), to determine if these properties would reduce the size of the probable password space such that it is susceptible to a brute-force guessing attack. Our results show that DAS appears to be less secure against such attacks than previously believed. Under reasonable parameter choices, where the size of the full DAS password space is 58 bits, the size of our combined classes is surprisingly tiny - only between 31 and 41 bits.

Our Contributions. To graphical password schemes in general, our contributions include: the identification of graphical password complexity properties, the definition of two broad classes of probable graphical passwords, and the introduction of graphical dictionaries. To DAS specifically, our contributions include: an analysis of the size of a conjectured probable password space of DAS, an understanding of how much security can be gained by increasing grid size in DAS, and the proposal of *grid selection*: a two-factor enhancement to the DAS scheme from which we expect good security pay-back. We determine that increasing the grid size in DAS appears to be less effective than previously believed. Our results motivate password rules for DAS and proactive graphical password checkers. We propose a preliminary set of DAS password rules based on our analysis, the most significant of which is to increase a password's stroke-count.

Organization. The sequel is organized as follows. §2 briefly discusses related work. §3 presents a password complexity model for graphical passwords, provides an overview of attacker strategies, and defines classes of probable graphical passwords and their corresponding graphical dictionaries (based on our password complexity model). §4 explains how we apply our graphical dictionaries to DAS. §5 presents our results and analysis of the security of DAS; the most compelling result is given in §5.3. §6 discusses methods to increase graphical password security, with a particular focus on DAS. §7 discusses additional observations, possible extensions to this work, and concluding remarks (including practical considerations).

2. RELATED WORK

The security for a password scheme depends at least in part on its resistance to dictionary attack. To prevent on-line dictionary attacks, Pinkas and Sander discuss human-in-the-loop methods [Pinkas and Sander 2002]; see also [Stubblebine and van Oorschot 2004]. One defence against off-line dictionary attacks is to reduce the probability of cracking through enforcing password policies and proactive password checking. Yan discusses some popular proactive textual password checkers [Yan 2001] such as *cracklib*. To perform effective proactive textual password checking, it is important to understand available textual password cracking dictionaries and tools (e.g., *Crack* [Muffett 2004] and *John the Ripper* [Openwall Project 2004a]).

Graphical password schemes proposed to date can be generally categorized as recognition-based or recall-based. This categorization is also used by Suo et al.'s recent survey of graphical passwords [Suo et al. 2005]. Monrose and Reiter's overview of graphical passwords [Monrose and Reiter 2005] provides a different (yet similar) categorization.

One recognition based scheme using hash visualization [Perrig and Song 1999] was implemented in a program called Déjà Vu [Dhamija and Perrig 2000]. Generally, in this scheme a user has a portfolio of

pictures of cardinality F that they must be able to distinguish within a group of presented pictures of cardinality T. Another recognition-based scheme called *Passfaces* [Real User Corporation 2004] requires that a user select a set of human faces as their password. Similar to Déjà Vu, the user is expected to correctly select each of the faces in their password from a set (or sets) of presented faces. *Story* [Davis et al. 2004] is a recognition-based scheme similar to *Passfaces*, but uses a variety of photo categories (e.g., everyday objects, locations, food, and people), and the user is asked to create a story to help them remember their portfolio.

Recall-based schemes include that proposed by Birget et al. [Birget et al. 2003] (and later implemented [Wiedenbeck et al. 2005]), which requires a user to click on several points on a background picture, and the DAS scheme ([Jermyn et al. 1999]; see §4.1), which involves user-defined drawings. Both schemes require exactly repeatable passwords (as defined within each scheme), allowing the password to be stored as the output of a one-way function, or used to generate cryptographic keys. Given reasonable-length passwords in a 5×5 grid (i.e., length ≤ 12), the full password space of DAS was shown to be larger than that of the full textual password space for text passwords of length ≤ 8.

Regarding memorability issues for graphical passwords, Davis et al. [Davis et al. 2004] examine user choice in two recognition-based schemes, showing a strong bias in user choice. Jermyn et al. [Jermyn et al. 1999] argue that the DAS scheme has a large memorable password space by modelling user choice as those passwords describable by a short algorithm. They also examine the size of the password space for combinations of one or two rectangles, and show that this is comparable to the size of many textual password dictionaries.[2]

Jermyn et al. [Jermyn et al. 1999] suggest that the security of graphical password schemes benefit from the current lack of knowledge of their probability distribution; this motivates our present work, which apparently weakens this argument.

Relevant psychology literature on human memory is discussed in §3.1.

3. PASSWORD COMPLEXITY MODEL

Since textual password dictionaries focus on words people recall better, we are lead to consider what types of images people recall more easily (and thus by conjecture would presumably choose as graphical passwords). We assume that users will find graphical passwords that minimize their complexity, and model memorable (and thus probable) graphical passwords as those that have low complexity.

Based on pattern complexity factors from Attneave [Attneave 1955a], we introduce a set of graphical password complexity properties: password length, number of components (i.e., the visually distinct parts of the graphical password), symmetry, and number of turns in each component; in this paper, we focus on all but the latter. Many studies support that mirror symmetric images are more easily recalled, as reviewed in this section.

3.1 Relevant Psychological Memory Studies

Generally, free recall is ordered along the concreteness continuum: concrete words are recalled more easily than abstract words, pictures more easily than concrete words, and objects better than pictures [Madigan 1983]. Various studies support this result (e.g., [Kirkpatrick 1894; Calkins 1898; Madigan and Lawrence 1980]). Another study [Bower et al. 1975] found that a series of line drawings is poorly remembered if the subject is unable to interpret the drawings in a meaningful way. The more concrete a drawing, the more meaningful it will be to the viewer.

Patterns in what *types* of images people recall better than others could be used to create classes of memorable and thus probable passwords, if such classes are sufficiently small.

There appears to be little existing research that examines the *types* of pictures people recall better. However, one cognitive study with interesting implications showed experimentally how visual recall progressively changed over time toward a symmetric version of the image [Perkins 1932]. Given a set of asymmetrical, geometric images, when test subjects were asked to draw the image from recall, all changes made from the originals were in the direction of some balanced or symmetrical pattern. This change was progressive over time toward a symmetric pattern. That people recall images as increasingly symmetric with time suggests to us that people prefer images that are symmetric. This changed our direction from exploring what specific

[2]Note that rectangles are a subclass of our Class 1 probable passwords. We note that all of the illustrative pictures from the original DAS paper also fall within our Class 1 or 2 probable passwords (see §7 for details).

images people are more likely to recall, to finding evidence that people have better recall for patterns and images that are symmetric.

A representative overview of literature for human symmetry perception [Tyler 1996] notes that many objects in our environment are symmetric. There is also significant evidence [Wagemans 1996] that mirror symmetry has a special status in human perception over other symmetry types such as repetition, translation or rotational symmetry, which were found to require scrutiny; in contrast, mirror symmetry is "effortless, rapid, and spontaneous" [Tyler 1996].

The classical studies mentioned above found better recall for pictures than words, and better recall for objects than pictures. If people recall objects best, and most objects are mirror symmetric, this suggests that people may recall mirror symmetric patterns best. This is supported by an observation by Attneave [Attneave 1955b]: when subjects were given random patterns and symmetric patterns of dots, the symmetric ones were more accurately reproduced than random patterns with the same number of dots. Attneave theorized that this may indicate that some perceptual mechanism is capable of organizing or encoding the redundant pattern into a simpler, more compact, less redundant form [Attneave 1955b]. In a separate study, French [French 1954] observed that dot patterns that were symmetric were more easily remembered. Intuitively, this is no surprise – in the case of mirror symmetry, a subject must only recall half of the image and its reflection axis in order to reconstruct the entire image.

Mirror symmetry has a special meaning to human's visual perception, particularly when the axis is about the vertical and horizontal planes. Mirror symmetry has been found to be more easily perceived as having meaning when it is about the vertical axis, followed by when it is about the horizontal axis [Wagemans 1996]. Note that most living organisms and plants, as well as almost all forms of human construction are mirror symmetric (reflective) about a vertical axis.

Attneave's findings of pattern complexity [Attneave 1955a] also imply that people are better at recalling a low number of components. We define "low" as at most 4, based on a collection of studies described below. In 2004, Vogel and Machizawa [Vogel and Machizawa 2004] found neurophysiological evidence that the human visual short term memory is limited to 3-4 symbols. Similar values were obtained for the number of dots recalled in grids of different sizes (recall decreased significantly after 3 or 4 dots) [Ichikawa 1982].

This motivated the examination of stroke-count in an informal user study of 16 students [Nali and Thorpe 2004] for DAS passwords, which (although very small) suggested a user tendency to choose passwords with a small stroke-count: 80% of users chose passwords with a stroke-count of 1-3 and 90% chose passwords with a stroke-count of 6 or less.

3.2 Model Motivated by Studies

Supported by these collective studies, we propose the following.

Conjecture 1 *Since people are more likely to recall symmetric images and patterns, and people perceive mirror symmetry as having a special status, a significant subset of users are likely to choose mirror symmetric patterns as their graphical password.*

More specifically, we propose that the mirror symmetric patterns chosen are more likely to be about vertical or horizontal axes. For graphical passwords, this leads us to define a *Class 1 probable password* (Definition 1). Findings that people are more likely to recall a low number of *components* (Definition 3) leads us to define a *Class 2 probable password* (Definition 2).

Definition 1 ***(Class 1 probable password).*** A *Class 1 probable password* (C_1-password) is a graphical password that exhibits mirror symmetry about a vertical or horizontal axis in its components (see Definition 3). Thus each component is either mirror symmetric in its own right, or is part of a mirror symmetric pair of components.

Definition 2 ***(Class 2 probable password).*** A *Class 2 probable password* (C_2-password) is a graphical password with a small number $3 \leq c \leq 8$ of components (see Definition 3).

Definition 3 ***(component).*** A *component* is a visually distinct part of an image.

The *Class 1 probable password space* is composed of the set of encoded representations of C_1-passwords , which form a graphical dictionary (i.e., a *Class 1 graphical dictionary*). *Class 2 probable password space* and *Class 2 graphical dictionary* are defined analogously.

3.3 Attack Strategy and Defining Graphical Dictionaries

There are two main attack strategies that an adversary can follow when trying to crack passwords: attacking a specific user account, or attacking a system by guessing the passwords for some larger set of (e.g., random) accounts. If the probability distribution for user choice within the password space is known, one can estimate the expected number of guesses for a specific user password using statistical expectation, or estimate the number of guesses required to guess a password on a system by using entropy (see [Massey 1994]). We use neither approach since we do not know these probabilities.

If C_1-passwords and C_2-passwords are significantly more probable in practice as we conjecture, the probability distribution of the graphical password space of some graphical password schemes (e.g., DAS) is highly non-uniform, significantly reducing the entropy of the password space. Similarly, the entropy is adversely affected if users favour passwords with characteristics that define other relatively small subsets of the full password space. We define graphical dictionaries for guessing attacks based on various subsets of passwords further in this section.

A clever attacker doing a brute-force guessing attack would prioritize candidate password guesses according to their probability of being chosen. We believe our graphical dictionaries would be the basis of such an ordering. We suggest that a clever attacker may prioritize a multi-class graphical dictionary according to passwords with an increasing number of components, and belonging to our classes as follows:

(1) Class 1 ∩ Class 2,

(2) Class 2 − (Class 1 ∩ Class 2),

(3) Class 1 − (Class 1 ∩ Class 2),

(4) Full password space − (Class 1 ∪ Class 2)

Each class might also be internally prioritized as discussed in their respective sections, below.

Class 1 Dictionaries. A logical way to prioritize the Class 1 dictionary is to assume that it is more likely for a user to choose a single reflection axis, or reflection axes that are close together. If an image's components are symmetric about axes that are far apart (e.g., Fig. 1a), the image does not appear to be symmetric as a whole; we say such images are *locally symmetric*. When an image's components are symmetric about axes that are close together (e.g., Fig. 1b), it is increasingly symmetric as a whole, producing an image that is *pseudo-symmetric*. When all components of an image are symmetric about the same axis (e.g., Fig. 1c), it produces an image that is symmetric as a whole (i.e., *globally symmetric*). The set of globally symmetric passwords best captures the symmetry discussed in §3.1, and our intuition suggests that global symmetry (Fig. 1c) is more likely than pseudo-symmetry (Fig. 1b), which is more likely than local symmetry (Fig. 1a).

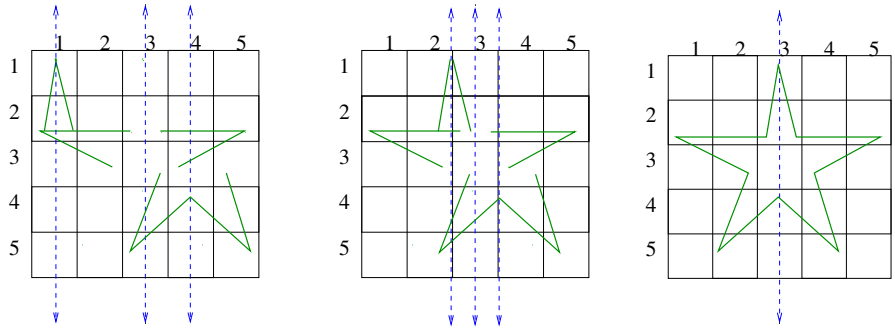


Fig. 1. Example Class 1 probable DAS passwords containing the same components, symmetric about different patterns of axes: (a) 3 different, scattered axes, (b) 3 different, nearby axes, and (c) a single axis.

This leads us to define sub-classes of Class 1, based on the number of axes that the image's components are symmetric about. Assuming pseudo-symmetry is more likely than global symmetry, an attacker may place passwords that are composed of strokes symmetric about a small set of close axes at a higher priority in the graphical dictionary. Additionally, for drawing-based schemes (e.g., DAS), if the user subconsciously uses the input area to frame the drawing (i.e., using the grid as part of the drawing's overall symmetry), the resulting drawings would be symmetric about the centermost axes.

Definition 4 ***(Class 1a).*** *Class 1a* is the subset of passwords in Class 1 that use only the center 3 of each set of horizontal or vertical axes (e.g., the marked axes in Fig. 2), producing pseudo-symmetric images.

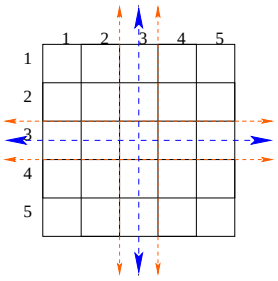


Fig. 2. Class 1a reflection axes for the DAS scheme. The thickest axes are the vertical and horizontal center axes. Adjacent axes are marked as thinner.

Definition 5 ***(Class 1b).*** *Class 1b* is the subset of passwords in Class 1 that use only the center of each set of horizontal or vertical axes, producing globally symmetric images.

Class 1b captures all passwords that are globally symmetric and centered about the grid (vertically and/or horizontally), plus those that have components symmetric about the center vertical and horizontal axes (e.g., the coffee cup in Fig. 3).

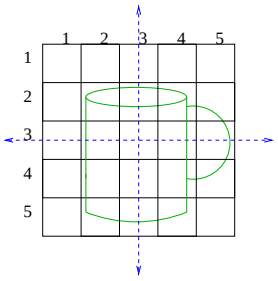


Fig. 3. Example of a Class 1b DAS drawing. One component (the handle) is symmetric about the center horizontal axis, and another (the cup), is symmetric about the center vertical axis.

Class 1b is a subset of Class 1a, which is a subset of Class 1. We expect that an attacker would order Class 1b passwords first in a Class 1 graphical dictionary, followed by the remaining Class 1a passwords, and finally the remaining passwords in Class 1.

Class 2 Dictionaries. An obvious attack strategy for Class 2 graphical dictionaries is to prioritize based on the number of components, in increasing order. A dictionary attack would thus try all entries with one component, then two, etc. Given the evidence to show a threshold number of 3 or 4 components is more memorable (recall §3.1), we focus on those C_2-passwords with $c = 4$ for our analysis. An attacker with a particular account in mind might expand a dictionary to also consider larger values of c.

Class 3 and 4 Dictionaries. Here we identify two other major types of symmetry, although according to cognitive studies, they do not hold the same special status as mirror symmetry: repetitive and rotational.

We define *Class 3 probable passwords* and *Class 4 probable passwords* as passwords exhibiting repetitive (e.g., Fig. 4) and rotational (e.g., Fig. 5) symmetry respectively. We note that these classes (and possibly many others) might also be placed in a graphical dictionary. We do not pursue their details in this paper.

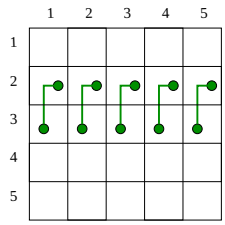


Fig. 4. Example DAS password exhibiting repetitive symmetry, and not included in Class 1 or 2.

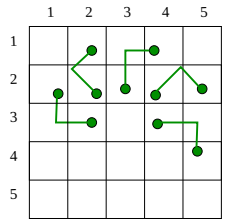


Fig. 5. Example DAS password exhibiting rotational symmetry, and not included in Class 1 or 2.

4. APPLYING GRAPHICAL DICTIONARIES TO DAS

To augment the original (and to our knowledge the only previous) security evaluation of DAS [Jermyn et al. 1999], we determine the size of the more probable subsets of the DAS Class 1 and Class 2 probable password spaces of §3.2, i.e., the number of encoded DAS passwords (see §4.1) representing at least one Class 1 (respectively Class 2) probable password. This is based on the reasoning that the number of entries in a "successful" attack dictionary provides a measure of security.

The DAS graphical password scheme relies on a user's ability to recall their DAS password "exactly" (as defined by the resolution of the encoding scheme). For this analysis, we only consider passwords that are allowed within DAS (see §4.1). What users must recall can be divided into two parts: the temporal order and number of strokes used in the drawing; and the final appearance of the drawing. C_1-passwords consider the latter, and C_2-passwords partially consider the former (a user must recall an increasing amount of temporal order information with an increasing number of components).

Assumptions concerning the temporal order of C_1-passwords in DAS (i.e., the order of the input of cells) are made in §4.2 for our analysis, leading us to define the set S_1 (Definition 11). In order to map C_1-passwords to DAS, we must discuss these assumptions about temporal order: §4.2.1 discusses our terminology and general approach, and §4.2.2 discusses additional cases. §4.3 discusses the mapping of C_2-passwords to DAS, leading us to define the set S_2 (Definition 14).

4.1 Review of DAS

The DAS scheme [Jermyn et al. 1999; Monrose 1999] decouples the position of password input from the temporal order, producing a larger password space than textual password schemes with keyboard input (where the order in which characters are typed predetermines their position).

A DAS password is a simple picture drawn on a $G \times G$ grid. Each grid cell is denoted by two-dimensional coordinates $(x, y) \in [1 \ldots G] \times [1 \ldots G]$. For DAS, an *encoded* password is a sequence of coordinate pairs listing the cells through which the drawing passes, in the order in which it passes through them. Each time the pen is lifted from the grid surface, this "pen-up" event is represented by the distinguished coordinate

pair $(G+1, G+1)$. Two drawings having the same encoding (i.e., crossing the same sequence of grid cells with pen-up events in the same places in the sequence) are considered equivalent.[3] Drawings are divided into equivalence classes in this manner.

DAS disallows passwords considered difficult to repeat exactly (e.g., passwords involving user input lying close to a grid boundary). The definition of "close to a grid boundary" is imprecise [Jermyn et al. 1999]; we define it as any part of a stroke for which the cell it lies within is indiscernible, meaning it lies within the *fuzzy boundary* of a grid line. Any stroke is invalid if it starts or ends on a fuzzy boundary, or if it crosses through the fuzzy boundary near the intersection of grid lines. We reuse the following terminology.

—The *neighbours* $N_{(x,y)}$ of cell (x, y) are $(x-1, y), (x+1, y), (x, y-1)$ and $(x, y+1)$.

—A *stroke* is a sequence of cells $\{c_i\}$, in which $c_i \in N_{c_{i-1}}$ and which is void of a pen-up.

—A DAS *password* is a sequence of strokes separated by pen-ups.

—The *length of a stroke* is the number of coordinate pairs it contains.

—The *length of a DAS password* is the sum of the lengths of its strokes (excluding pen-ups).

Jermyn et al. [Jermyn et al. 1999] recursively compute the (full) password space size, i.e., the number of distinct encoded graphical passwords in DAS. This gives an upper bound on the size of the probable password space and thus on the security of the scheme. It is assumed that all passwords of total length greater than some fixed value have probability zero. They compute the full password space size for passwords of total length at most L_{max}. For $L_{max} = 12$ and a 5×5 grid, this is 2^{58}, exceeding the number of textual passwords of 8 characters or fewer constructed from the printable ASCII codes ($\sum_{i=1}^{8} 95^i < 2^{53}$).

4.2 Class 1 DAS Graphical Dictionaries

In this section we describe how we map the visual mirror symmetry from Class 1 to DAS passwords. Our general approach and additional cases are described in §4.2.1 and §4.2.2 respectively, leading us to define our mapping from Class 1 to DAS in §4.2.3.

4.2.1 *Basic Terminology and General Approach.* Our approach is to model each Class 1 password in DAS as a series of strokes (each representing a single component or pair of components) drawn in a symmetric manner per Definition 12. Each such stroke is modelled by a *defining stroke* from virtual start point $s = (x, y)$ to virtual end point $e = (x, y)$. We enumerate all possible values of s and e for each reflection axis, using these values as a model of the symmetry, and then consider the ways the resulting (user-drawn) stroke can be drawn in a symmetric manner. We emphasize that s and e are used to model the symmetry, and are not necessarily the start and end points of the user-drawn stroke.

To capture mirror symmetric DAS passwords, we first consider which reflection axes to use. We assume that the user references the grid lines for the symmetry in the drawing, since if the reflection axis is a point of reference, the password will be easier to repeat exactly. Therefore, the reflection axes considered are those that cut a set of grid cells (Fig. 6a), or are on a grid line (Fig. 6b). This means that any symmetric password drawn such that its axis is off-center within a set of cells is not considered. For example, the password in Fig. 7a is visually symmetric when the grid is not in place, but we do not consider it part of the set of Class 1 probable passwords in DAS since its reflection axis is not on a grid line or centered in a set of cells as shown in Fig. 7b. We justify this assumption as follows: it is more difficult for a user to draw an exactly repeatable symmetric password without a visible point of reference on the grid for the reflection axis.

We thus define the set of axes within a $W \times H$ grid (width W, height H): $A = A_h \cup A_v$; $A_h = \{1, 1.5, 2, \ldots, (H-1).5, H\}$; $A_v = \{1, 1.5, 2, \ldots, (W-1).5, W\}$. Here $i.5$ is the grid line separating rows i and $i+1$, or columns i and $i+1$ respectively.

Definition 6 ***(symmetric area).*** The *symmetric area* (given a reflection axis a), is the area between a and the closest grid boundary parallel to a, reflected about a (see Fig. 8).

One way to draw a stroke in a symmetric manner is to draw a stroke within the symmetric area (possibly crossing over the reflection axis), then draw its reflection about the reflection axis as shown in Fig. 9a. We call the initial stroke from virtual start point s to virtual end point e that the reflection is based upon the

[3]We note that this implies a many-to-one mapping of user drawings to encoded DAS passwords.

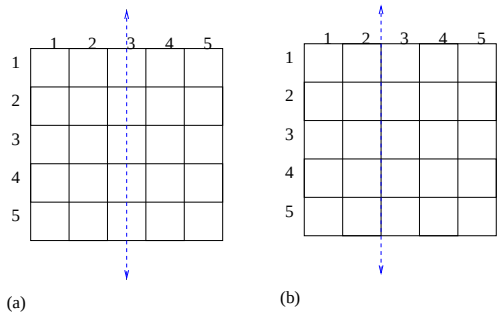


Fig. 6. Possible axes can (a) cut a set of cells; or (b) be on a grid line between sets of cells.

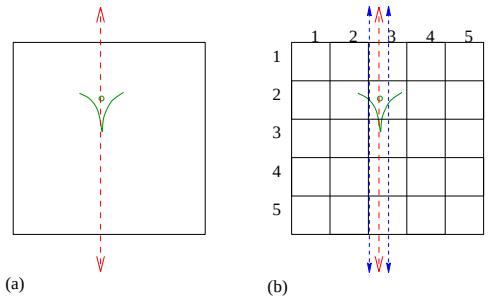


Fig. 7. Drawing that is symmetric about a difficult to reference axis. Assuming the v is drawn before the dot, the encoding of (b) is (2,2), (3,2), (3,3), (3,2), pen-up, (3,2), pen-up. If shifted slightly right to be symmetric about the vertical axis $x = 3$, it has symmetric encoding: (3,2), (3,3), (3,2), pen-up, (3,2), pen-up.

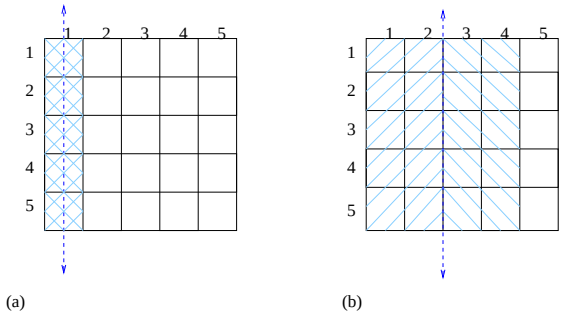


Fig. 8. Example symmetric areas for (a) the axis $x = 1$; and (b) $x = 2.5$

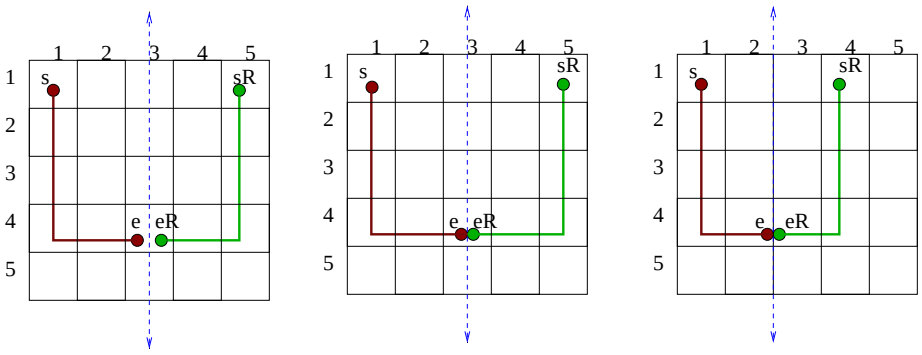
defining stroke, and the reflection the *reflected stroke*, which can be drawn from s^R (the reflection of s) to e^R (the reflection of e) or vice versa. When the defining stroke is drawn from e to s, we consider (and count) it a different defining stroke, since input order is relevant in DAS.

Definition 7 ***(symmetric stroke).*** A *symmetric stroke* is a stroke (or pair of strokes) drawn in a symmetric manner (see Definition 12). Whether actually drawn as a single stroke or pair of strokes, it is modelled by the combined result of a defining stroke and a reflected stroke about an axis a, remaining within the bounds of the symmetric area defined by a.

Definition 8 ***(disjoint case).*** The *disjoint case* consists of two user-drawn strokes holding the property of exact reflection. Given a defining stroke z, its reflected stroke z^R (relative to an axis a) is said to be an *exact reflection* if z^R is z's mirror image about a and they are separated by a pen-up.

Exact reflection is not required to have a symmetric stroke (see §4.2.2).

As a stroke pair that falls within the disjoint case has the property of exact reflection, its length will always be even. The product of the number of ways to draw a defining stroke and the number of ways to draw its reflected stroke provides the number of ways to draw that stroke in the disjoint case.



(a) Disjoint case (b) Continuous case (i) (c) Continuous case (ii)

Fig. 9. Disjoint and Continuous Cases. Symmetric strokes consist of a defining stroke (solid line from s to e) and reflected stroke (solid line from s^R to e^R). The last two, visually representing the letter 'U', show continuous cases where: (b) the axis cuts a set of cells; and (c) the axis is on a grid line.

4.2.2 *Continuous and Enclosed Cases.* A point $p = (x, y)$ in an encoded defining stroke is *potentially continuous* if it denotes a cell that is either cut by the reflection axis a in question, or adjacent to a when a is on a grid line. If p is potentially continuous, its reflection p^R is in the same cell as p or in a neighbouring cell, and thus the stroke can be drawn directly from p to p^R without a pen-up. When the start and end points of a defining stroke are potentially continuous, the three most apparently straightforward ways to draw the resulting symmetric stroke are as follows: disjointly, as one continuous stroke, or as one continuous enclosed stroke (see Definitions 8, 9, and 10).

A symmetric stroke can be drawn as a continuous case when the defining stroke's end point is potentially continuous.

Definition 9 ***(continuous case).*** The *continuous case* consists of one user-drawn stroke, whereby the defining stroke continues through the axis to the reflected stroke, in a single, continuous stroke.

For example, the encoding for Fig. 9b would be: (1,1), (1,2), (1,3), (1,4), (2,4), (3,4), (4,4), (5,4), (5,3), (5,2), (5,1), ending with a pen-up. The stroke could also be drawn in the reverse order. Examples of the same visual representation of a 'U', with one disjoint and the other continuous, are shown in Figures 9a and b. Note that the continuous case's encoding is different, depending on whether the axis a cuts a set of cells or is on a grid line. If a cuts a set of cells as in Fig. 9b, the defining stroke's endpoint e is the same as its reflection e^R. Since there is no pen-up to separate e from e^R, it cannot appear in the encoding twice, thus e^R does not appear in the resulting encoding. If a is on a grid line (Fig. 9c), e and e^R reside in different cells, and both e and e^R appear in the resulting encoding.

A symmetric stroke can be drawn as an enclosed case when both the defining stroke's start and end points are potentially continuous (e.g., Fig. 10).

Definition 10 ***(enclosed case).*** The *enclosed case* consists of one user-drawn stroke, whereby the defining stroke continues through the reflection axis to the reflected stroke, and then ends up back in the same cell as the start of the defining stroke, essentially creating an enclosed shape. When a drawing is enclosed, the user-drawn stroke may start and end at any point in the shape (e.g., Fig. 10a).

As with the continuous case, the enclosed case's encoding is different, depending on whether the axis a cuts a set of cells or is on a grid line. The continuation of the defining stroke into the reflected stroke will be encoded as in the continuous case; the difference between these two cases is the encoding to join the reflected stroke back into the defining stroke. When a is on a grid line, the start point of the defining stroke

is repeated as the last point of the user-drawn stroke (e.g., Fig. 10b). When a cuts a set of cells (e.g., Fig. 10a), it is the same as the continuous case since $s = s^R$, enclosing the shape. Thus, to avoid double-counting, we must (and do) exclude the cases where s is potentially continuous from the continuous case.

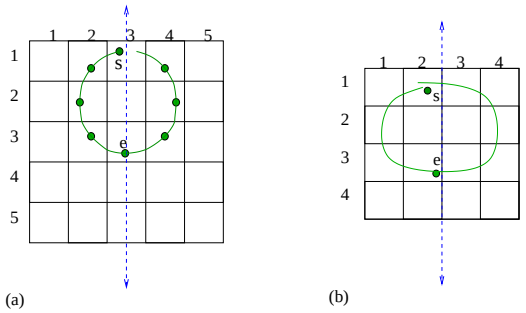


Fig. 10. Different types of the enclosed case. The reflection axis in (a) cuts a set of cells, and (b) is on a grid line. Case (a) shows all possible user-drawn start/end points in the symmetric stroke modelled by s and e.

4.2.3 *Classes S_1, S_{1a}, and S_{Ib} .* The definition of C_1-passwords takes into account only their final visual appearance. There is a one-to-many relationship between a given C_1-password and the number of ways it can be drawn in the DAS scheme (which are then mapped to possibly fewer unique DAS encodings). We believe there are some more likely *ways* that users will draw mirror symmetric components in their DAS passwords; we use S_1 to denote this "more probable" subset of unique DAS encodings of C_1-passwords defined as follows.

Definition 11 ***(S_1).*** S_1 is the DAS-related subset of Class 1 passwords, containing only those passwords whose components' composite stroke(s) are drawn in a *symmetric manner* (Definition 12).

Definition 12 ***(symmetric manner).*** A DAS stroke is drawn in a *symmetric manner*, if it follows one of the *disjoint case*, *continuous case*, or *enclosed case* (per Definitions 8, 9, and 10). For context, see Fig. 11.

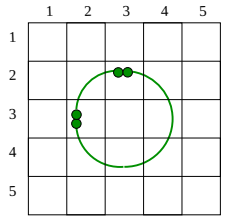


Fig. 11. Example DAS password that is in Class 1, but is *not* drawn in a symmetric manner.

Definition 13 ***(S_{1a} and S_{1b}).*** S_{1a} and S_{1b} are subsets of S_1 that belong to Class 1a and Class 1b respectively. More formally, $S_{1a} = S_1 \cap$ Class 1a; $S_{1b} = S_1 \cap$ Class 1b.

Preliminary user studies have shown that the temporal order has an adverse effect on user's ability to recall a DAS password [Goldberg et al. 2002]. If so, then we expect that users will choose DAS passwords with less complexity (e.g., fewer strokes). We believe that S_1 captures the easiest (and thus most likely to be chosen) ways to draw C_1-passwords, although not all possible ways (see §5 for discussion of the implications of this approximation). The details of how we enumerated the DAS Class 1 space (or equivalently, graphical dictionaries) can be found in Appendix C.

4.3 Class 2 DAS Graphical Dictionaries

In Definition 2, we chose to define a C_2-password to have at most c components; for our DAS analysis, we use $c = 4$, since this value has support from cognitive studies (recall §3.1). We assume that users will try to minimize the amount of temporal information to recall by drawing each component with the smallest number of strokes possible; we make the simplifying assumption that this is 1 (one). Thus for DAS, we characterize a C_2-password by the number of composite strokes. This leads us to define S_2 below. We quantify the relationship between the DAS password space and the stroke-count in §5.2.

Definition 14 *(S_2).* S_2 is the DAS-related subset of Class 2 passwords, containing only those passwords having a stroke-count ≤ 4.

Our general approach is to determine how many DAS passwords are of length at most a given maximum password length L_{max}, with a maximum stroke-count of X. Counting all passwords of length at most L_{max} follows [Jermyn et al. 1999]. We modify the function *P(L,G)* [Jermyn et al. 1999] that counts the number of passwords of length $\leq L$ (where $1 \leq L \leq L_{max}$ is the password length and G is the grid dimension), to limit the stroke-count in each password to at most X.

5. SECURITY ANALYSIS OF DAS

Following the relevant parts of the attack strategy from §3.3, and using our subsets of Class 1, we expect that a DAS graphical dictionary would be prioritized according to the following ordering (see Fig. 12):

(1) $S_{1b} \cap S_2$
(2) $(S_{1a} - S_{1b}) \cap S_2$
(3) $(S_1 - S_{1a}) \cap S_2$
(4) $S_2 - S_1$
(5) $S_1 - S_2$
(6) the remainder of the DAS password space that does not fall into any of S_1 or S_2.

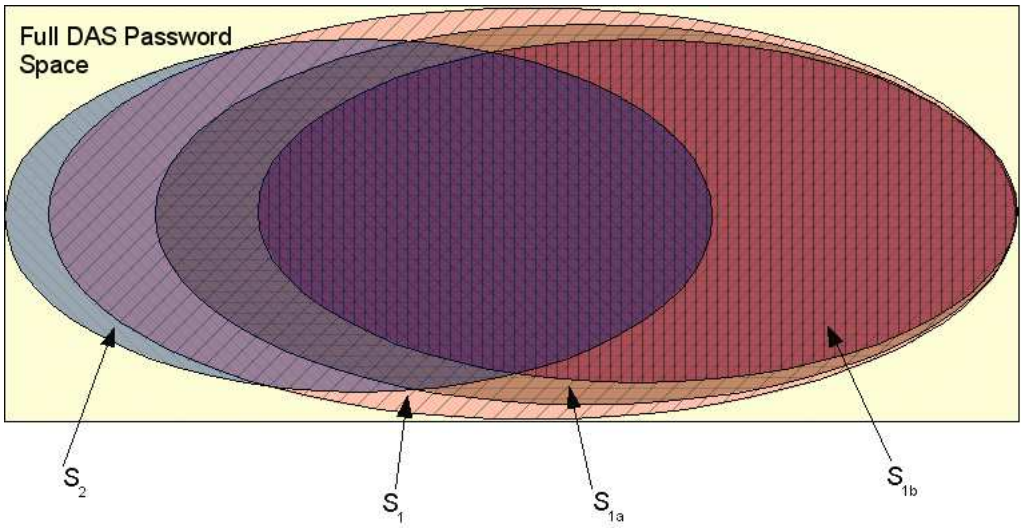


Fig. 12. Illustrative (bit-size) relationship between S_{1b}, S_{1a}, S_1, and S_2.

While we expect (as mentioned in §4.2.3) that our Class 1 graphical dictionary for DAS will include most C_1-passwords, we recognize that some C_1-passwords will not be included due to our definition of S_1. However, even if our dictionary only includes as few as e.g., $\frac{1}{8}$ of the ways users would typically draw C_1-passwords, this would imply our approximated bit-sizes are off by at most three bits – not significantly affecting our

results. Furthermore, there is strong anecdotal support that our assumptions are indeed quite realistic (see §7), and thus the above error estimate is conservative.

We approximate the size of password spaces S_{1b}, S_{1a}, S_1, and S_2, individually (§5.1 and §5.2) and combined (§5.3). For illustrative purposes, we estimate example attack times for exhausting each of these as potential graphical dictionaries in §5.4.

5.1 Approximate Size of Class 1 Graphical Dictionaries

To estimate the size of various subsets of the password space, many (equivalent) counting methods are possible. Our definitions in §3.2 and §3.3 define what DAS passwords are in the Class 1 probable password space. We detail our counting methods for generating these results in Appendix C.

Table I gives sample results for S_1 and the subsets of S_{1a} and S_{1b} (recall Definitions 11 and 13). Values given are log_2(number of passwords). S_{1a} and S_{1b} both show an exponential reduction from the full DAS space: S_{1b} grows at an exponential rate of approximately 3.6 bits per unit increase in password length and S_{1a} grows at a corresponding rate of approximately 4.0, whereas the full DAS space and S_1 grow at a corresponding rate of approximately 4.8. For example, when $L_{max} = 12$, the size of the full space is 57.7 bits, S_1 is 56.7 bits, S_{1a} is 48.1 bits, and S_{1b} is 42.7 bits. The size of the full DAS password space was cross-checked using a variation of our method (for full details, see Appendix C), closely matching the results given in [Jermyn et al. 1999].

L_{max}	1	2	3	4	5	6	7	8	9	10
Full DAS space	4.7	9.5	14.3	19.2	24.0	28.8	33.6	38.4	43.2	48.1
S_1	4.7	9.5	14.3	19.1	23.9	28.7	33.6	38.4	43.2	48.0
S_{1a}	3.3	7.7	11.6	15.7	19.8	23.8	27.9	31.9	36.0	40.0
S_{1b}	3.3	6.9	10.5	14.1	17.7	21.2	24.8	28.4	32.0	35.6
L_{max}	11	12	13	14	15	16	17	18	19	20
Full DAS space	52.9	57.7	62.5	67.3	72.2	77.0	81.8	86.6	91.4	96.2
S_1	52.8	57.6	62.4	67.2	72.0	76.8	81.7	86.5	91.3	96.1
S_{1a}	44.1	48.1	52.1	56.2	60.2	64.3	68.3	72.4	76.4	80.4
S_{1b}	39.1	42.7	46.3	49.9	53.4	57.0	60.6	64.2	67.8	71.4

Table I. Bit-size of graphical password space, for total length at most L_{max} on a 5×5 grid.

Each of the three subclasses of C_1-passwords presented in Table I allow perceptually quite distinct classes of drawings (recall Fig. 1). We initially found the size of S_1 to be surprisingly close to that of the full DAS space; however, upon reflection this is sensible, as the only requirement for a stroke to be symmetric is that it is locally symmetric about any axis in A (e.g., Fig. 1a), which includes the combinatorially large set of all permutations of dots and lines of length two.

The smaller the set of reflection axes used, the smaller the corresponding graphical sub-dictionary becomes. As discussed earlier, a reasonable attack strategy is to narrow down the graphical dictionary to a small number of axes, or prioritize a search such that globally symmetric passwords (e.g., Fig. 1c) are considered first. When any single axis (or two) are considered at a time to produce globally symmetric passwords, each result will never be larger than that for the two center axes, as the latter maximizes the symmetric area in which the passwords can reside. Thus, the maximum dictionary size of such a variation would be at most a small constant factor, proportional to the number of axes considered, of that using only the center axes.

5.2 Approximate Size of Class 2 Graphical Dictionaries

For Class 2 graphical dictionaries, following [Jermyn et al. 1999] we focus our discussion on a set of results for a 5×5 grid size, giving the bit-size of the password space for passwords of length at most L_{max} (from 1 to 20) and each possible maximum stroke-count X. The full set is provided in Table III (Appendix A).

Fig. 13 shows the effect (log_2) of increasing L_{max} for a given X: the password space's size increases exponentially, illustrating the roles of both L_{max} and X in the DAS password space. Note that the left ends of all but the line representing the full password space ($X = L_{max}$) have been omitted for simplicity – we

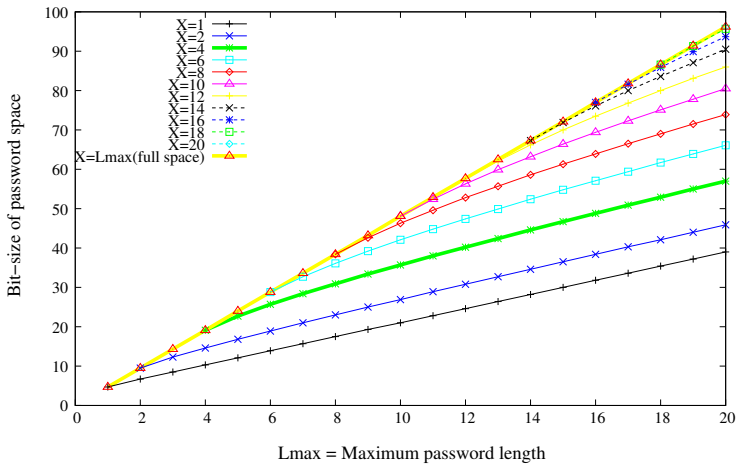


Fig. 13. Size of graphical password space for passwords of at most X strokes (for a 5×5 grid and a fixed maximum password length L_{max}). S_2 (recall that $c = 4$) is represented by the thick line where $X = 4$.

know that the maximum stroke-count for a password of length L_{max} is L_{max}, thus any line where $X > L_{max}$ will have the same value as when $X = L_{max}$.

Increasing X accounts for at least one half of the bit-size (see the difference between the $X = 1$ line and the full space line, when $L_{max} \geq 5$). The top line, where $X = L_{max}$, in Fig. 13 shows what one would likely expect from reading the original DAS paper [Jermyn et al. 1999] (i.e., "58 bits of security" against guessing attacks when $L_{max} = 12$). The other thick line, where $X = 4$ represents the size of S_2. We suggest (although we cannot "prove" this)[4] that S_2 is more representative of "effective security" of unconstrained user-selected DAS passwords, taking into account the entropy reduction due to user choice in DAS passwords, and assuming all passwords composed of 4 or fewer strokes are equi-probable. This graph highlights the impact of the number of strokes on the DAS password space; the size of the password space is significantly smaller (40 bits) if users choose a password of length at most 12, composed of 4 or less strokes. The password space still increases with longer password lengths (as shown by the rise in each curve), but at a slower rate for smaller stroke-counts (as shown by the gaps between curves). Note that for a fixed L_{max}, a smaller maximum stroke-count X implies a longer average stroke length.

Much of the strength of DAS arises from the temporal order in terms of the direction of strokes, and more importantly, the order in which these strokes are drawn. This explains why increasing the stroke-count results in large increases in the size of the password space: there are many more possible permutations of these strokes.

A high stroke count for a fixed password length implies a short average stroke length. This leads us to ask: how much of the total password space consists of passwords composed entirely from seemingly unlikely combinations of very short strokes, i.e., entirely of strokes of length 1 and/or 2? This is easily computed by discarding those strokes of length > 1 (or > 2), and the results are interesting: passwords composed entirely of strokes of length 1 comprise approximately $\frac{1}{4}$ of the total password space, and passwords composed of only strokes of length ≤ 2 comprise approximately $\frac{1}{2}$ of the password space. Thus $\frac{1}{2}$ of the password space is already accounted for by passwords that appear to be very unlikely user choices.

This might be examined from another angle: how much of the total password space consists of passwords *without* any strokes of length 1? Again this is easily counted , with the result that if users do not draw any strokes of length 1 (e.g., dots) in their DAS password, the size of the password space when $L_{max} = 12$ on a 5×5 grid is effectively reduced from 58 to 40 bits, very dramatically increasing susceptibility to dictionary attacks. This result, by itself, perhaps paints the clearest picture that the security of DAS passwords in practice is very likely to be significantly less than originally believed (unless user choice is somehow controlled; see §6). We expect that many user-chosen passwords will not contain any length-1 strokes – and note with interest that 5 of 8 examples in the original DAS paper do not contain any length-1 strokes.

[4]See further discussion at end of 5.2.

5.3 Approximate Size of Combined Class 1 and 2 Dictionary

Figure 14 shows how restricting the maximum number of strokes, while also staying within different subclasses of C_1-passwords affects the size of the password space. All data shown is for $L_{max} = 12$ on a 5×5 grid.

The triangle point on the upper-right corner is the total number of DAS passwords of length ≤ 12 on a 5×5 grid. Again, this 2^{58} value is the "security" measure one might expect from reading the original DAS paper. Notice the triangle point where $X \leq 4$ – this value of 2^{40} shows the affect of number of strokes on the full DAS password space. The S_{1b} bar directly below combines the $X \leq 4$ with S_{1b}; we believe that this value of 2^{31} is a better "ball park" measure of the amount of security DAS provides.[5]

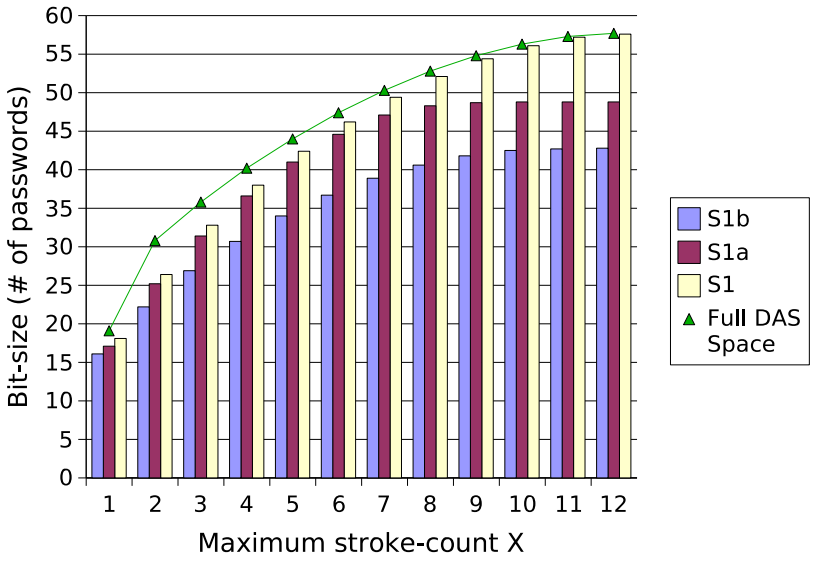


Fig. 14. Bit-size of DAS graphical password space. Values are given for each dictionary, with a fixed total password length at most 12, with at most X strokes on a 5×5 grid (see Table IV in Appendix A for actual data points).

5.4 Illustrative Attack Times

For illustrative purposes only, we estimate how long it might take to perform a dictionary attack against DAS using our graphical dictionaries. The exact time required to perform a dictionary attack depends on implementation details. For this illustration, we assume that password verification involves at least hashing the entered password using the MD5 hash algorithm, then comparing the result to a stored value.[6]

Here the attack time is at least the time to hash each candidate password. We calculate two sets of times: one for an attacker with one *Pentium 4* 3.2GHz machine, and another for an attacker with one thousand such machines. It is reasonable to consider that a determined attacker could exploit one thousand, or even one hundred thousand machines using a worm, to distribute the password-cracking load. Using an MD5 performance result of 3.66 cycles/byte for a *Pentium 3* 800MHz machine [Nakajima and Matsui 2002] (scaled to 3.2GHz), and a 512 bit block size, approximately 1.37×10^7 hashes can be performed per second per machine. Given the assumed resources, the time to generate the password hashes is calculated by (1), and given in Table II for various sets.

$$\text{Time (secs)} = \frac{\text{dictionary size}}{(1.37 \times 10^7) \times \text{no. of machines}} \tag{1}$$

[5]In fact, this estimate itself very likely over-estimates security, as (a) it does not take into account other highly-probable (but not yet known) "Classes" of passwords we have not considered; and (b) within Classes 1 and 2, our assumption of passwords being equi-probable is unrealistic, thus over-estimating password entropy.

[6]It should be clear that this attack time could be significantly increased by various implementation enhancements, e.g., see discussion in §6.

Dictionary ($L_{max} = 12$)	Time to exhaust (1 machine)	Time to exhaust (1000 machines)
Full Space	541.8 yrs	197.8 days
S_1	505.6 yrs	184.5 days
S_{1a}	255 days	6.1 hours
S_{1b}	6 days	8.7 mins
S_2	1.1 days	1.5 mins
$S_{1b} \cap S_2$	2.1 mins	0.1 secs

Table II. Illustrative times to exhaust various entire DAS graphical dictionaries (3.2GHz machines, 5×5 grid).

The tabulated time is the worst case attack time if the password(s) under attack belong to the dictionary in use. An attacker may achieve success substantially faster if dictionary entries are ordered according to their probability of occurring. Note also that if the target passwords are not in any of the dictionaries, the attack fails.

Although entirely illustrative and heavily dependent on the assumed parameters, Table II highlights the practical implications of the graphical dictionary size. Assuming that we want an attacker with 1000 computers at 3.2GHz to require an average of 10 years to exhaust these dictionaries, the dictionary size must be approximately 2^{63}. Referring to our S_2 dictionary (stroke-count ≤ 4), if we extrapolate our results, this requires $L_{max} = 23$. This implies that for this level of security (and a 5×5 grid), DAS users should choose passwords of length at least 23 to resist dictionary attacks based solely on S_2 – a somewhat negative result for DAS.

On a more positive note, some of the larger textual password dictionaries (which are effective in practice) contain approximately 4×10^7 entries [Openwall Project 2004b]. One of our smallest graphical dictionaries (S_2) exceeds this number of entries for $L_{max} \geq 9$. This implies that even if users choose passwords in S_2 (and if these are equiprobable – which admittedly is unlikely), provided the password length is at least 9, DAS may still offer greater security than textual passwords against dictionary attacks.

6. TOWARDS INCREASING GRAPHICAL PASSWORD SECURITY

There are many potential methods to increase the security of graphical passwords in practice. These include password rules (see §6.1) and mnemonic strategies to aid users in choosing stronger passwords. Passphrases, i.e., sentences that help users recall a password, are a textual password mnemonic. An analogous mnemonic proposed for graphical passwords is to create a story based on the picture(s) [Davis et al. 2004].

Implementation enhancements for textual passwords can also be applied to graphical passwords. Graphical passwords that are exactly repeatable (such as DAS), can be stored using a one-way hash. Hashing algorithms with an adaptable cost (e.g., see [Provos and Mazieres 1999]) and that use *password stretching* or repeated hashing of passwords (e.g., see [Halderman et al. 2005]) increase the computational cost of guessing attacks. "Salting" adds random data to the computation of each user's password hash, and thus if any users have the same password, the hashes will be different. Salting thus forces an attacker to compute a new hash for each password guess/user combination, increasing computational cost of guessing attacks against a set of users.

Also, minor enhancements of a graphical password implementation can increase security (albeit typically at some cost in usability), e.g., additional user-selected characteristics of drawings such as colour, backgrounds, and textures. The graphical password space could also be increased by increasing the area from which the user can select a graphical password – in DAS, this could be achieved by increasing the grid size, which is briefly mentioned, but not quantified in discussions of DAS [Thorpe and van Oorschot 2004a; Jermyn et al. 1999]. This may have a negative effect on the memorability of DAS passwords, since human recall performance decreases as a function of grid size [Ichikawa 1982].

In §6.2, we examine the impact of grid size on the DAS password space. In §6.3, we propose *grid selection* as a practical method of increasing the DAS password space through the grid size. These results can be used in determining practical design choices for implementing DAS.

6.1 Graphical Password Rules

The impact of symmetry, a small stroke-count, and/or no strokes of length 1 on the DAS password space strongly motivate the use and enforcement of graphical password rules. Given our knowledge to date, we

suggest the following as an initial set of graphical passwords rules, particularly for DAS, but which may generalize as noted below. We expect this list will grow over time.

(1) Require a stroke-count of at least $\lfloor \frac{L_{max}}{2} \rfloor$.
(2) Disallow passwords having global reflective (mirror) symmetry (e.g., Class 1b).
(3) Require at least one stroke of length 1.

The second rule can be generalized to other graphical password schemes, and the first can be generalized as simply having more than 4 components (e.g., for PassPoint schemes [Birget et al. 2003; Wiedenbeck et al. 2005], the user must click on at least 4 points).

6.2 Effect of Grid Size Increase

We examine a set of results with 3 variables: maximum stroke length L_{max} (from 1 to 20), maximum stroke-count X (from 1 to 20 or password length), and grid dimension (5, 6, 7, and 10, assumed to be square). We stop at 10×10 due to interface size limitations; if each grid cell (the level of user-input error tolerated) is $1cm \times 1cm$, this yields a $10cm \times 10cm$ input area, already exceeding the screen dimensions of many PDAs and smart phones.

Fig. 15 displays results for $L_{max} = 12$ and selected values of X. The password space increases with the grid size, but it is dependent on the value of X. The bulk of the password space growth (about 20 bits) occurs when $X = L_{max}$ (the top line), but when $X \leq \lfloor \frac{L_{max}}{2} \rfloor$, one gains only about 5-10 more bits by increasing the grid area from 25 cells (5×5 grid) to 100 cells (10×10 grid). The bulk of the password space growth also occurs between $X = \frac{L_{max}}{2}$ and $X = L_{max}$ for larger values of L_{max}. Since we know that passwords composed of short strokes make up a large proportion of the password space when $X \geq \lfloor \frac{L_{max}}{2} \rfloor$, this indicates that the bulk of the gain achieved by increasing the grid size applies to passwords with a large number of small strokes. Additionally, the password space growth achieved by increasing grid size when $X \leq \lfloor \frac{L_{max}}{2} \rfloor$ is comparable when $L_{max} = 12$ and 20, indicating that there is no combined advantage between increasing L_{max} and grid size, unless a large number of short strokes are used. Thus in practice, unless we can ensure (e.g., through password rules) that passwords contain a large number of short strokes, increasing the grid size provides relatively low security pay-back.

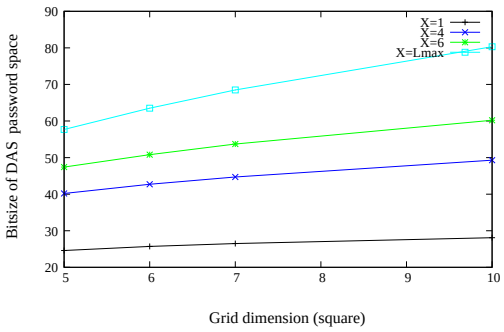


Fig. 15. Effect of grid size on bit-size of DAS password space for $L_{max} = 12$ and various values of X.

If we follow the example of a user choosing a password of length 12, composed of 4 or fewer strokes, the increase in the password space by increasing the grid size from a 5×5 grid to a 10×10 grid is comparable (10 bits) to the increase achieved by keeping a 5×5 grid and increasing the stroke-count to 7, or allowing 4 or fewer strokes and increasing L_{max} to 17. Furthermore, asking users to use only dots as composite strokes for their passwords (which is undesirable for usability, i.e., memorability, reasons) on a 5×5 grid gives 5 more bits than increasing the grid size to 10×10 and allowing a small stroke-count (≤ 4). These results suggest that increasing the grid size does not provide as much security pay-back as increasing other parameters (i.e., stroke-count or password length).

The results given here and in §5 show that the stroke-count X has a larger impact on the effective bit-size of the DAS password space, than L_{max} and grid dimension. The remaining question is which of these parameters will have the least negative impact on users ability to recall their passwords. It is possible that

users would prefer to use a larger grid rather than using passwords with a large number of short strokes and/or of longer length. In response to the dissapointing results of the grid size increase, we next propose a way of simulating a large grid size while minimizing the impact on the user.

6.3 Using Grid Selection

We have suggested that increasing the grid size to increase the DAS password space does not provide enough security pay-back to compensate for the increased difficulty to reproduce a password. Alternatively, we propose introducing a *selection grid*: an initial large, fine-grained grid from which the user selects a *drawing grid*, a rectangular region to zoom in on, in which they may enter their password. Fig. 16 illustrates this idea, where the selection grid is on the left, the drawing grid on the right. The selection grid is checkered to aid a user in locating specific cells.

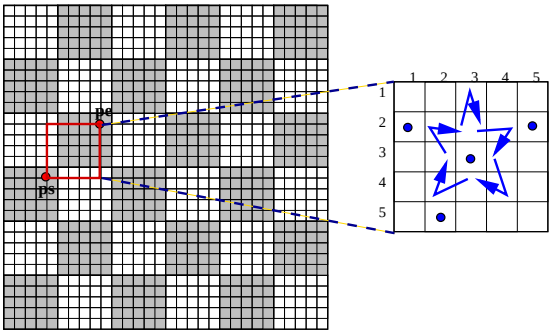


Fig. 16. Grid selection: a user selects a drawing grid in which to draw their password.

The idea of zooming in on an area is similar to that discussed by Birget et al. [Birget et al. 2003], except we are zooming in on a grid to draw in, not a picture to click a point within. This technique, which we call *grid selection*, may add as much as another 16 bits (see below) to the password space. Using grid selection, the benefits of an increased password space could be combined with input times marginally greater than without grid selection, and with what would seem to be minimal additional demand on user memory. We now discuss how many more passwords can reasonably be added to the DAS password space by a grid selection implementation.

We first consider a reasonable size for the selection grid. Assuming a $10cm \times 10cm$ input area, and that one cm^2 can be partitioned into 9 cells with approximately the same resolution as 10pt font, a 30×30 selection grid resolution would be reasonable. A user does not draw their password in this grid, but rather selects their drawing grid (with a pre-specified area range, e.g., between 25 and 100 cells) by e.g., selecting two diagonal corner points. To prevent users from choosing small drawing grids (e.g., a 2×2 grid) that would reduce the password space, a minimum drawing grid width D_W or height D_H is defined; for this example, we assume 5. On the other hand, to allow the user to be able to easily reproduce their drawing, a reasonable level of error-tolerance (which depends on the grid resolution) is necessary. Thus, a restriction on how fine the grid can be must also be defined; we assume that $1cm^2$ cells provide a desirable level of error tolerance giving a maximum D_W and D_H of 10.

In order to choose a drawing grid from the selection grid, the user selects a starting point $p_s = (x, y)$ and an ending point $p_e = (x, y)$ (thereby implying D_W, and D_H). There are 31^2 possible values for p_s (although not all equi-probable, as discussed below). Depending on p_s, there are different possible values for p_e (and thus D_W and D_H). We determine the number of possible grids for each possible p_s, using our assumed parameters of a 30×30 selection grid, a minimum drawing grid width of 5, and a maximum drawing grid width of 10. Assuming the input order of p_s and p_e is taken into account, the number of possible drawing grids is 79524, which could add up to 16 bits of password space to DAS encodings.

Of course, a grid selection implementation may be used in a predictable manner: e.g., users may be more likely to choose a drawing grid that traces the 5×5 checker squares of the selection grid. Disallowing these options eliminates $36 \times 4 = 144$ grids. Furthermore if we disallow all square grids, we lose 13324 grids, leaving $79524 - 13324 = 66200$ drawing grids. This still allows a gain of up to 16 bits.

If it is found that certain grids are such highly probable choices that it provides far less than 16 bits of additional entropy, then it may be acceptable to assign a grid to a user. If the grid squares are checkered in a coloured pattern, the task of finding and remembering an assigned drawing grid may not be too difficult. If all drawing grids are assigned on an equi-probable basis, then to perform a brute-force attack, an attacker must run the attack against each drawing grid from the selection grid.

7. ADDITIONAL OBSERVATIONS AND CONCLUDING REMARKS

Our work extends and complements existing analysis and understanding of DAS graphical passwords. A viable graphical password dictionary attack strategy follows from our results. We believe that initial estimates [Jermyn et al. 1999] of the practical security of DAS were overly optimistic.

Although the focus of our work is the application of graphical dictionaries to the DAS scheme, it is our (unverified) belief that this method of analysis could be applied to a variety of other graphical password schemes. For example, Birget et al. [Birget et al. 2003] (see also [Wiedenbeck et al. 2005]) propose that users be provided an image, and asked to choose a given number of click points. One could assume that a relatively small number of symmetric objects in an image might be significantly more likely to be chosen (than other random points) as click points (i.e., "hot spots" for click points).

One may question the likelihood of users choosing passwords within our classes of graphical passwords, based solely on cognitive studies on visual recall. It is very interesting to note that out of the 8 example passwords in the original DAS paper [Jermyn et al. 1999], all are covered by either our Class 1 or Class 2 probable passwords. 4 of these passwords fall under our definition of Class 2 (and if we used a threshold of $c = 5$ instead of 4, this increases to 7). Furthermore, 5 fall under our definition of Class 1b (globally symmetric) and 7 fall under our definition of Class 1 (locally symmetric). This provides very strong anecdotal support for our belief that Classes 1 and 2 form a significant portion of the passwords from which users are likely to choose (in the absence of password rules).

Our study of the DAS password space shows that of the complexity properties examined, stroke-count has the largest impact. Note that the space of DAS passwords when $L_{max} = 12$, restricted to at most 4 strokes (or alternatively with no strokes of length 1) is only approximately 40 bits; already in 1997, a 40 bit key space was shown to be crackable in a matter of hours [CNET News.com Staff 1999]. If users often choose passwords with a small stroke-count, or with no strokes of length one, DAS falls easily to a dictionary attack. Out of the 8 example passwords in the original DAS paper, 5 did not have any strokes of length one. An attacker could also use this knowledge to further prioritize a dictionary (e.g., within Class 1; see §5).

Greater effective security might be achieved by graphical password schemes having larger probable password spaces, even if at the expense of a smaller full (theoretical) password space. For example, encouraging users to draw passwords with more strokes might result in an increase to the effective password space, if at the same time some other means reduces the difficulty for a user to recall their password. This could be achieved by e.g., disregarding the direction of strokes. A better understanding of the breakdown of what users have the most difficulty recalling (leading to a more formal definition of graphical password complexity properties) would be beneficial to understanding how to strengthen graphical password implementations. Attneave [Attneave 1955a] provides some hints in his examination of memory and complexity factors for visual patterns. Our work highlights the need for memory and usability studies to understand more about what users recall best, and how to take advantage of the strengths of human memory to create more secure and usable graphical password implementations.

It is unclear to us that such a graphical password implementation is likely to be DAS (as originally proposed, for both the reasons discussed in this paper and other usability issues[7]). Our work shows that in order for DAS to be secure against off-line dictionary attacks, users must be able to recall asymmetric passwords with a relatively large number of short strokes. Usability might suffer with our password rules (in both memorability and repeatability), which might in turn lead to other predictable patterns in user choice. It is an open question as to whether DAS passwords will need significant reworking; a user study employing our password rules would be most beneficial towards this end.

[7]For example, issues arising from some user inputs being disallowed due to the requirement for exact repeatability.

Acknowledgements

The first author acknowledges Canada's Natural Sciences and Engineering Research Council (NSERC) for funding a NSERC Discovery Grant and his Canada Research Chair in Network and Software Security. The second author acknowledges NSERC for funding a Canada Graduate Scholarship.

REFERENCES

Attneave, F. 1955a. Complexity of Patterns. *American Journal of Psychology 68*, 209–222.

Attneave, F. 1955b. Symmetry, Information and Memory for Patterns. *American Journal of Psychology 68*, 209–222.

Birget, J.-C., Hong, D., and Memon, N. 2003. Robust Discretization, With an Application to Graphical Passwords. Cryptology ePrint Archive, Report 2003/168. `http://eprint.iacr.org/`, site accessed Jan. 12, 2004.

Bower, G. H., Karlin, M. B., and Dueck, A. 1975. Comprehension and Memory For Pictures. *Memory and Cognition 3*, 216–220.

Calkins, M. 1898. Short Studies in Memory and Association from the Wellesley College Laboratory. *Psychological Review 5*, 451–462.

CNET News.com Staff. 1999. 40-bit crypto proves no problem. CNET News.com. `http://news.com.com`, site accessed May 14, 2004.

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

Dhamija, R. and Perrig, A. 2000. Déjà Vu: A User Study Using Images for Authentication. In *9th USENIX Security Symposium*.

French, R.-S. 1954. Identification of Dot Patterns From Memory as a Function of Complexity. *Journal of Experimental Psychology 47*, 22–26.

Goldberg, J., Hagman, J., and Sazawal, V. 2002. Doodling Our Way to Better Authentication. In *Conference on Human Factors and Computing Systems* (April 20 - 25). ACM Press, 868–869. CHI '02 extended abstracts on Human Factors in Computer Systems.

Halderman, J. A., Waters, B., and Felten, E. W. 2005. A convenient method for securely managing passwords. In *Proceedings of the 14th International World Wide Web Conference*. ACM Press, 471–479.

Ichikawa, S.-I. 1982. Measurement of Visual Memory Span by Means of the Recall of Dot-in-Matrix Patterns. *Behavior Research Methods and Instrumentation 14(3)*, 309–313.

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

Kirkpatrick, E. A. 1894. An Experimental Study of Memory. *Psychological Review 1*, 602–609.

Klein, D. 1990. Foiling the Cracker: A Survey of, and Improvements to, Password Security. In *The 2nd USENIX Security Workshop*. 5–14.

Madigan, S. 1983. Picture Memory. In *Imagery, Memory and Cognition*, J. C. Yuille, Ed. Lawrence Erlbaum Associates Inc., N.J., U.S.A., 65–89.

Madigan, S. and Lawrence, V. 1980. Factors Affecting Item Recovery and Hypermnesia in Free Recall. *American Journal of Psychology 93*, 489–504.

Massey, J. 1994. Guessing and Entropy. In *ISIT: Proceedings IEEE International Symposium on Information Theory*. 204.

Monrose, F. 1999. Towards Stronger User Authentication. Ph.D. thesis, NY University.

Monrose, F. and Reiter, M. K. 2005. Graphical passwords. In *Security and Usability*, L. Cranor and S. Garfinkel, Eds. O'Reilly, Chapter 9, 147–164.

Muffett, A. 2004. Crack password cracker. `http://ciac.llnl.gov/ciac/ToolsUnixAuth.html`, site accessed Jan. 12, 2004.

Nakajima, J. and Matsui, M. 2002. Performance Analysis and Parallel Implementation of Dedicated Hash Functions. In *Advances in Cryptology – Proceedings of EUROCRYPT 2002*. 165–180.

Nali, D. and Thorpe, J. 2004. Analyzing User Choice in Graphical Passwords. Tech. Report TR-04-01, School of Computer Science, Carleton University, Canada.

Openwall Project. 2004a. John the Ripper password cracker. `http://www.openwall.com/john/`, site accessed Jan.7, 2004.

Openwall Project. 2004b. Wordlists. `http://www.openwall.com/passwords/wordlists/`, site accessed Jan.7 2004.

Perkins, F. 1932. Symmetry in Visual Recall. *American Journal of Psychology 44*, 473–490.

Perrig, A. and Song, D. 1999. Hash Visualization: A New Technique to Improve Real-World Security. In *International Workshop on Cryptographic Techniques and E-Commerce*. 131–138.

Pinkas, B. and Sander, T. 2002. Securing Passwords Against Dictionary Attacks. In *9th ACM Conference on Computer and Communications Security*. ACM Press, 161–170.

Provos, N. and Mazieres, D. 1999. A Future-Adaptable Password Scheme. In *Proceedings of the USENIX Annual Technical Conference*.

Real User Corporation. 2004. About passfaces. `http://www.realuser.com/cgi-bin/ru.exe/_/homepages/technology/passface.htm`, site accessed May 25, 2004.

Spafford, E. 1989. Crisis and Aftermath (The Internet Worm). *Comm. of the ACM 32(6)*, 678–687.

Spafford, E. H. 1992. OPUS: Preventing Weak Password Choices. *Comput. Secur. 11,* 3, 273–278.

Stubblebine, S. and van Oorschot, P. 2004. Addressing Online Dictionary Attacks with Login Histories and Humans-in-the-Loop. In *In Financial Cryptography 8th International Conference* (February 9-12). Springer-Verlag LNCS 3110.

Suo, X., Zhu, Y., and Owen, G. S. 2005. Graphical Passwords: A Survey. In *21st Annual Computer Security Applications Conference (ACSAC)* (December 5-9).

Thorpe, J. and van Oorschot, P. 2004a. Graphical Dictionaries and the Memorable Space of Graphical Passwords. In *13th USENIX Security Symposium* (August 9-13).

Thorpe, J. and van Oorschot, P. 2004b. Towards Secure Design Choices for Implementing Graphical Passwords. In *20th Annual Computer Security Applications Conference (ACSAC 2004)* (December 6-10). IEEE.

Tyler, C. 1996. Human Symmetry Perception. In *Human Symmetry Perception and its Computational Analysis*, C. Tyler, Ed. VSP, The Netherlands, 3–22.

Vogel, E. K. and Machizawa, M. G. 2004. Neural Activity Predicts Individual Differences in Visual Working Memory Capacity. *Nature 428*, 748–751.

Wagemans, J. 1996. Detection of Visual Symmetries. In *Human Symmetry Perception and its Computational Analysis*, C. Tyler, Ed. VSP, The Netherlands, 25–48.

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

Yan, J. 2001. A Note on Proactive Password Checking. ACM New Security Paradigms Workshop, New Mexico, USA.

Appendix A - Data Tables for Figures 13 and 14

X / L_{max}	1	2	3	4	5	6	7	8	9	10
1	4.7									
2	6.7	9.5								
3	8.5	12.3	14.3							
4	10.3	14.6	17.5	19.1						
5	12.1	16.8	20.3	22.7	24.0					
6	13.9	18.9	22.8	25.7	27.7	28.8				
7	15.7	21.0	25.1	28.4	30.9	32.7	33.6			
8	17.5	23.0	27.3	30.9	33.8	36.1	37.6	38.4		
9	19.3	25.0	29.5	33.4	36.6	39.2	41.2	42.6	43.2	
10	21.0	26.9	31.7	35.7	39.1	42.1	44.4	46.3	47.5	48.1
11	22.8	28.9	33.8	38.0	41.6	44.8	47.4	49.6	51.3	52.4
12	24.6	30.8	35.8	40.2	44.0	47.4	50.3	52.8	54.8	56.3
13	26.4	32.7	37.9	42.4	46.4	49.9	53.0	55.7	58.0	59.9
14	28.2	34.6	39.9	44.6	48.7	52.4	55.7	58.6	61.1	63.2
15	30.0	36.5	41.9	46.7	50.9	54.8	58.2	61.3	64.0	66.4
16	31.8	38.4	43.9	48.8	53.2	57.1	60.7	63.9	66.8	69.4
17	33.6	40.3	45.9	50.9	55.3	59.4	63.1	66.5	69.6	72.3
18	35.4	42.1	47.9	52.9	57.5	61.7	65.5	69.0	72.2	75.1
19	37.2	44.0	49.8	55.0	59.6	63.9	67.9	71.5	74.8	77.8
20	39.0	45.9	51.8	57.0	61.8	66.1	70.2	73.9	77.3	80.5

X / L_{max}	11	12	13	14	15	16	17	18	19	20
11	52.9									
12	57.3	57.7								
13	61.3	62.1	62.5							
14	64.9	66.2	67.0	67.3						
15	68.4	70.0	71.1	71.9	72.1					
16	71.6	73.5	75.0	76.1	76.7	77.0				
17	74.7	76.8	78.6	80.0	81.0	81.6	81.8			
18	77.7	80.0	82.0	83.6	84.9	85.9	86.4	86.6		
19	80.6	83.1	85.2	87.1	88.7	89.9	90.8	91.3	91.4	
20	83.4	86.0	88.4	90.5	92.2	93.7	94.9	95.7	96.1	96.2

Table III. Bit-size of DAS with different maximum lengths and stroke-counts as illustrated in Fig. 13 of §5.2. Values are given for total length at most L_{max} with at most X strokes on a 5×5 grid. S_2 is shown in the column where $X = 4$.

X / *Dictionary*	1	2	3	4	5	6	7	8	9	10	11	12
Full Space	24.6	30.8	35.8	40.2	44.0	47.4	50.3	52.8	54.8	56.3	57.3	57.7
S_1	18.2	26.4	32.8	38.0	42.4	46.2	49.4	52.1	54.4	56.1	57.2	57.6
S_{1a}	17.6	25.2	31.4	36.6	41.0	44.6	47.1	48.3	48.7	48.8	48.8	48.8
S_{1b}	16.1	22.2	26.9	30.7	34.0	36.7	38.9	40.6	41.8	42.5	42.7	42.8

Table IV. Bit-size of graphical password space as illustrated in Fig. 14 of §5.3. Values are given for each dictionary, with a fixed total password length at most 12, with at most X strokes on a 5×5 grid.

Appendix B - Supporting Data Tables

Tables V, VI, VII, and VIII give sample results computed using the method outlined in §8.2 for various values of L_{max}, X, and $G \times G$ grids for $G = 5, 6, 7, 10$ respectively. Values given are log_2(number of passwords). The size of the full DAS password space (without a limitation on the number of strokes) was cross-checked and essentially agrees with the results given in [Jermyn et al. 1999].

Tables IX and X give sample results that show how much of the password space is composed solely of strokes of length ≤ 2 and of length $= 1$ respectively. Tables XI and XII give results for the effect of having no length-1 strokes (i.e., dots) or strokes of length ≤ 2 in a DAS password. The data in Tables IX to XII are for 5×5 grids.

X / L_{max}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	4.7																			
2	6.7	9.5																		
3	8.5	12.3	14.3																	
4	10.3	14.6	17.5	19.1																
5	12.1	16.8	20.3	22.7	24.0															
6	13.9	18.9	22.8	25.7	27.7	28.8														
7	15.7	21.0	25.1	28.4	30.9	32.7	33.6													
8	17.5	23.0	27.3	30.9	33.8	36.1	37.6	38.4												
9	19.3	25.0	29.5	33.4	36.6	39.2	41.2	42.6	43.2											
10	21.0	26.9	31.7	35.7	39.1	42.1	44.4	46.3	47.5	48.1										
11	22.8	28.9	33.8	38.0	41.6	44.8	47.4	49.6	51.3	52.4	52.9									
12	24.6	30.8	35.8	40.2	44.0	47.4	50.3	52.8	54.8	56.3	57.3	57.7								
13	26.4	32.7	37.9	42.4	46.4	49.9	53.0	55.7	58.0	59.9	61.3	62.1	62.5							
14	28.2	34.6	39.9	44.6	48.7	52.4	55.7	58.6	61.1	63.2	64.9	66.2	67.0	67.3						
15	30.0	36.5	41.9	46.7	50.9	54.8	58.2	61.3	64.0	66.4	68.4	70.0	71.1	71.9	72.1					
16	31.8	38.4	43.9	48.8	53.2	57.1	60.7	63.9	66.8	69.4	71.6	73.5	75.0	76.1	76.7	77.0				
17	33.6	40.3	45.9	50.9	55.3	59.4	63.1	66.5	69.6	72.3	74.7	76.8	78.6	80.0	81.0	81.6	81.8			
18	35.4	42.1	47.9	52.9	57.5	61.7	65.5	69.0	72.2	75.1	77.7	80.0	82.0	83.6	84.9	85.9	86.4	86.6		
19	37.2	44.0	49.8	55.0	59.6	63.9	67.9	71.5	74.8	77.8	80.6	83.1	85.2	87.1	88.7	89.9	90.8	91.3	91.4	
20	39.0	45.9	51.8	57.0	61.8	66.1	70.2	73.9	77.3	80.5	83.4	86.0	88.4	90.5	92.2	93.7	94.9	95.7	96.1	96.2

Table V. Bit-size of graphical password space on a 5×5 grid. Values are given for total length at most L_{max} with at most X strokes.

X / L_{max}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	5.2																			
2	7.3	10.5																		
3	9.2	13.4	15.8																	
4	11.0	15.8	19.1	21.1																
5	12.8	18.0	21.9	24.7	26.4															
6	14.6	20.2	24.5	27.8	30.3	31.7														
7	16.5	22.3	26.9	30.6	33.6	35.8	37.0													
8	18.3	24.3	29.2	33.2	36.6	39.3	41.2	42.3												
9	20.2	26.3	31.4	35.7	39.4	42.4	44.9	46.6	47.6											
10	22.0	28.4	33.6	38.1	42.0	45.4	48.2	50.4	52.1	52.9										
11	23.9	30.3	35.7	40.4	44.5	48.2	51.3	53.9	56.0	57.4	58.2									
12	25.7	32.3	37.9	42.7	47.0	50.8	54.2	57.1	59.5	61.5	62.8	63.5								
13	27.6	34.3	40.0	44.9	49.4	53.4	57.0	60.1	62.9	65.2	67.0	68.2	68.8							
14	29.4	36.3	42.0	47.2	51.8	55.9	59.7	63.0	66.0	68.6	70.7	72.4	73.5	74.1						
15	31.3	38.2	44.1	49.3	54.1	58.4	62.3	65.8	69.0	71.8	74.2	76.3	77.8	78.9	79.4					
16	33.1	40.1	46.1	51.5	56.3	60.8	64.8	68.5	71.9	74.9	77.6	79.9	81.8	83.3	84.2	84.7				
17	35.0	42.1	48.2	53.6	58.6	63.1	67.3	71.1	74.7	77.9	80.7	83.3	85.5	87.3	88.7	89.6	90.0			
18	36.8	44.0	50.2	55.7	60.8	65.4	69.7	73.7	77.4	80.7	83.8	86.5	89.0	91.1	92.8	94.1	94.9	95.3		
19	38.7	46.0	52.2	57.8	63.0	67.7	72.1	76.2	80.0	83.5	86.7	89.7	92.3	94.6	96.6	98.3	99.5	100.3	100.6	
20	40.5	47.9	54.2	59.9	65.1	70.0	74.5	78.7	82.6	86.2	89.6	92.7	95.5	98.0	100.3	102.2	103.7	104.9	105.6	105.9

Table VI. Bit-size of graphical password space on a 6×6 grid. Values are given for total length at most L_{max} with at most X strokes.

X / L_{max}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	5.6																			
2	7.8	11.3																		
3	9.7	14.3	17.1																	
4	11.5	16.7	20.4	22.8																
5	13.4	19.0	23.3	26.5	28.5															
6	15.2	21.2	25.9	29.6	32.5	34.2														
7	17.1	23.3	28.3	32.5	35.8	38.4	39.9													
8	19.0	25.4	30.7	35.1	38.9	41.9	44.3	45.6												
9	20.9	27.5	32.9	37.6	41.7	45.2	48.0	50.1	51.3											
10	22.8	29.5	35.1	40.1	44.4	48.1	51.4	54.0	56.0	57.0										
11	24.6	31.5	37.3	42.4	47.0	51.0	54.5	57.5	60.0	61.8	62.8									
12	26.5	33.5	39.5	44.7	49.5	53.7	57.4	60.8	63.6	65.9	67.6	68.5								
13	28.4	35.5	41.6	47.0	51.9	56.3	60.3	63.8	67.0	69.6	71.8	73.4	74.2							
14	30.3	37.5	43.7	49.3	54.3	58.8	63.0	66.8	70.1	73.1	75.6	77.7	79.1	79.9						
15	32.2	39.5	45.8	51.5	56.6	61.3	65.6	69.6	73.2	76.4	79.2	81.6	83.6	84.9	85.6					
16	34.1	41.5	47.9	53.7	58.9	63.8	68.2	72.3	76.1	79.5	82.6	85.3	87.6	89.4	90.7	91.3				
17	35.9	43.5	50.0	55.8	61.2	66.1	70.7	75.0	78.9	82.5	85.8	88.8	91.3	93.5	95.3	96.5	97.0			
18	37.8	45.4	52.0	58.0	63.4	68.5	73.2	77.6	81.7	85.4	88.9	92.1	94.9	97.4	99.4	101.1	102.2	102.7		
19	39.7	47.4	54.1	60.1	65.7	70.8	75.7	80.2	84.4	88.3	91.9	95.2	98.3	101.0	103.3	105.3	106.9	108.0	108.5	
20	41.6	49.4	56.1	62.2	67.9	73.1	78.1	82.7	87.0	91.0	94.8	98.3	101.5	104.4	107.0	109.3	111.2	112.7	113.7	114.2

Table VII. Bit-size of graphical password space on a 7×7 grid. Values are given for total length at most L_{max} with at most X strokes.

X / L_{max}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	6.7																			
2	8.8	13.3																		
3	10.8	16.3	20.0																	
4	12.7	18.9	23.5	26.7																
5	14.6	21.2	26.5	30.6	33.4															
6	16.5	23.5	29.1	33.8	37.6	40.1														
7	18.5	25.6	31.6	36.7	41.0	44.5	46.8													
8	20.4	27.8	34.0	39.4	44.1	48.2	51.4	53.5												
9	22.3	29.9	36.3	42.0	47.0	51.5	55.2	58.2	60.2											
10	24.2	32.0	38.6	44.5	49.8	54.5	58.7	62.2	65.1	66.9										
11	26.2	34.1	40.9	46.9	52.4	57.4	61.9	65.8	69.2	71.9	73.6									
12	28.1	36.1	43.1	49.3	55.0	60.2	64.9	69.2	72.9	76.2	78.7	80.3								
13	30.0	38.2	45.2	51.6	57.5	62.9	67.8	72.3	76.4	80.0	83.1	85.5	87.0							
14	32.0	40.2	47.4	53.9	59.9	65.5	70.6	75.3	79.7	83.6	87.0	90.0	92.3	93.7						
15	33.9	42.2	49.5	56.2	62.3	68.0	73.3	78.2	82.8	86.9	90.7	94.0	96.8	99.0	100.4					
16	35.8	44.3	51.7	58.4	64.7	70.5	75.9	81.0	85.7	90.1	94.1	97.8	101.0	103.7	105.8	107.1				
17	37.8	46.3	53.8	60.6	67.0	72.9	78.5	83.7	88.6	93.2	97.4	101.3	104.8	107.9	110.6	112.6	113.8			
18	39.7	48.3	55.9	62.8	69.3	75.3	81.0	86.4	91.4	96.2	100.6	104.7	108.5	111.9	114.9	117.4	119.3	120.5		
19	41.6	50.3	58.0	65.0	71.6	77.7	83.5	89.0	94.2	99.1	103.7	107.9	111.9	115.6	118.9	121.8	124.2	126.1	127.2	
20	43.6	52.3	60.1	67.2	73.8	80.1	86.0	91.6	96.9	101.9	106.6	111.1	115.2	119.1	122.7	125.9	128.7	131.1	132.8	133.9

Table VIII. Bit-size of graphical password space on a 10×10 grid. Values are given for total length at most L_{max} with at most X strokes.

X / L_{max}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	4.7																			
2	6.7	9.5																		
3	6.7	12.2	14.3																	
4	6.7	13.4	17.4	19.1																
5	6.7	13.4	19.3	22.5	23.9															
6	6.7	13.4	20.2	24.9	27.5	28.7														
7	6.7	13.4	20.2	26.3	30.2	32.5	33.5													
8	6.7	13.4	20.2	26.9	32.1	35.4	37.5	38.3												
9	6.7	13.4	20.2	26.9	33.2	37.6	40.6	42.4	43.1											
10	6.7	13.4	20.2	26.9	33.6	39.1	43.0	45.7	47.3	47.9										
11	6.7	13.4	20.2	26.9	33.6	40.0	44.9	48.3	50.7	52.2	52.7									
12	6.7	13.4	20.2	26.9	33.6	40.3	46.1	50.4	53.6	55.7	57.0	57.5								
13	6.7	13.4	20.2	26.9	33.6	40.3	46.8	52.0	55.9	58.8	60.7	61.9	62.3							
14	6.7	13.4	20.2	26.9	33.6	40.3	47.0	53.0	57.7	61.3	63.9	65.7	66.7	6 7.1						
15	6.7	13.4	20.2	26.9	33.6	40.3	47.0	53.5	59.0	63.3	66.5	69.0	70.6	7 1.6	71.9					
16	6.7	13.4	20.2	26.9	33.6	40.3	47.0	53.7	59.9	64.8	68.8	71.8	74.0	7 5.6	76.4	76.7				
17	6.7	13.4	20.2	26.9	33.6	40.3	47.0	53.7	60.3	66.0	70.5	74.2	77.0	7 9.1	80.5	81.3	81.5			
18	6.7	13.4	20.2	26.9	33.6	40.3	47.0	53.7	60.4	66.7	71.9	76.1	79.5	8 2.1	84.1	85.4	86.1	86.3		
19	6.7	13.4	20.2	26.9	33.6	40.3	47.0	53.7	60.4	67.1	72.9	77.7	81.6	8 4.8	87.2	89.0	90.3	90.9	91.1	
20	6.7	13.4	20.2	26.9	33.6	40.3	47.0	53.7	60.4	67.2	73.5	78.9	83.4	8 7.1	90.0	92.3	94.0	95.1	95.7	95.9

Table IX. Bit-size of graphical password space when limited to strokes of length 1 and 2 on a 5×5 grid. Values are given for total length at most L_{max} with at most X strokes.

X / L_{max}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	4.7																			
2	4.7	9.3																		
3	4.7	9.3	14.0																	
4	4.7	9.3	14.0	18.6																
5	4.7	9.3	14.0	18.6	23.3															
6	4.7	9.3	14.0	18.6	23.3	27.9														
7	4.7	9.3	14.0	18.6	23.3	27.9	32.6													
8	4.7	9.3	14.0	18.6	23.3	27.9	32.6	37.2												
9	4.7	9.3	14.0	18.6	23.3	27.9	32.6	37.2	41.8											
10	4.7	9.3	14.0	18.6	23.3	27.9	32.6	37.2	41.8	46.5										
11	4.7	9.3	14.0	18.6	23.3	27.9	32.6	37.2	41.8	46.5	51.1									
12	4.7	9.3	14.0	18.6	23.3	27.9	32.6	37.2	41.8	46.5	51.1	55.8								
13	4.7	9.3	14.0	18.6	23.3	27.9	32.6	37.2	41.8	46.5	51.1	55.8	60.4							
14	4.7	9.3	14.0	18.6	23.3	27.9	32.6	37.2	41.8	46.5	51.1	55.8	60.4	65 .1						
15	4.7	9.3	14.0	18.6	23.3	27.9	32.6	37.2	41.8	46.5	51.1	55.8	60.4	65 .1	69.7					
16	4.7	9.3	14.0	18.6	23.3	27.9	32.6	37.2	41.8	46.5	51.1	55.8	60.4	65 .1	69.7	74.4				
17	4.7	9.3	14.0	18.6	23.3	27.9	32.6	37.2	41.8	46.5	51.1	55.8	60.4	65 .1	69.7	74.4	79.0			
18	4.7	9.3	14.0	18.6	23.3	27.9	32.6	37.2	41.8	46.5	51.1	55.8	60.4	65 .1	69.7	74.4	79.0	83.6		
19	4.7	9.3	14.0	18.6	23.3	27.9	32.6	37.2	41.8	46.5	51.1	55.8	60.4	65 .1	69.7	74.4	79.0	83.6	88.3	
20	4.7	9.3	14.0	18.6	23.3	27.9	32.6	37.2	41.8	46.5	51.1	55.8	60.4	65 .1	69.7	74.4	79.0	83.6	88.3	92.9

Table X. Bit-size of graphical password space when limited to strokes of length 1 on a 5×5 grid. Values are given for total length at most L_{max} with at most X strokes.

X / L_{max}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0.0																			
2	0.0	0.0																		
3	8.1	8.1	8.1																	
4	10.2	10.2	10.2	10.2																
5	12.1	12.1	12.1	12.1	12.1															
6	13.9	16.4	16.4	16.4	16.4	16.4														
7	15.7	19.2	19.2	19.2	19.2	19.2	19.2													
8	17.5	21.6	21.6	21.6	21.6	21.6	21.6	21.6												
9	19.3	23.9	25.0	25.0	25.0	25.0	25.0	25.0	25.0											
10	21.0	26.0	28.1	28.1	28.1	28.1	28.1	28.1	28.1	28.1										
11	22.8	28.0	30.8	30.8	30.8	30.8	30.8	30.8	30.8	30.8	30.8									
12	24.6	30.1	33.3	33.8	33.8	33.8	33.8	33.8	33.8	33.8	33.8	33.8								
13	26.4	32.0	35.6	36.9	36.9	36.9	36.9	36.9	36.9	36.9	36.9	36.9	36.9							
14	28.2	34.0	37.9	39.8	39.8	39.8	39.8	39.8	39.8	39.8	39.8	39.8	39.8	39.8						
15	30.0	36.0	40.1	42.4	42.7	42.7	42.7	42.7	42.7	42.7	42.7	42.7	42.7	42.7	42.7					
16	31.8	37.9	42.3	45.0	45.8	45.8	45.8	45.8	45.8	45.8	45.8	45.8	45.8	45.8	45.8	45.8				
17	33.6	39.8	44.4	47.4	48.7	48.7	48.7	48.7	48.7	48.7	48.7	48.7	48.7	48.7	48.7	48.7	48.7			
18	35.4	41.7	46.4	49.8	51.5	51.6	51.6	51.6	51.6	51.6	51.6	51.6	51.6	51.6	51.6	51.6	51.6	51.6		
19	37.2	43.6	48.5	52.1	54.2	54.6	54.6	54.6	54.6	54.6	54.6	54.6	54.6	54.6	54.6	54.6	54.6	54.6	54.6	
20	39.0	45.5	50.5	54.3	56.7	57.6	57.6	57.6	57.6	57.6	57.6	57.6	57.6	57.6	57.6	57.6	57.6	57.6	57.6	57.6

Table XI. Bit-size of graphical password space when limited to strokes greater than length 2 on a 5×5 grid. Values are given for total length at most L_{max} with at most X strokes.

X / L_{max}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0.0																			
2	6.3	6.3																		
3	8.4	8.4	8.4																	
4	10.3	12.9	12.9	12.9																
5	12.1	15.7	15.7	15.7	15.7															
6	13.9	18.1	19.6	19.6	19.6	19.6														
7	15.7	20.3	22.7	22.7	22.7	22.7	22.7													
8	17.5	22.4	25.5	26.4	26.4	26.4	26.4	26.4												
9	19.3	24.5	28.0	29.7	29.7	29.7	29.7	29.7	29.7											
10	21.0	26.5	30.3	32.6	33.2	33.2	33.2	33.2	33.2	33.2										
11	22.8	28.5	32.6	35.3	36.6	36.6	36.6	36.6	36.6	36.6	36.6									
12	24.6	30.4	34.8	37.9	39.6	40.0	40.0	40.0	40.0	40.0	40.0	40.0								
13	26.4	32.4	36.9	40.3	42.5	43.4	43.4	43.4	43.4	43.4	43.4	43.4	43.4							
14	28.2	34.3	39.0	42.7	45.2	46.6	46.9	46.9	46.9	46.9	46.9	46.9	46.9	46.9						
15	30.0	36.2	41.1	45.0	47.8	49.6	50.3	50.3	50.3	50.3	50.3	50.3	50.3	50.3	50.3					
16	31.8	38.1	43.2	47.2	50.3	52.5	53.6	53.7	53.7	53.7	53.7	53.7	53.7	53.7	53.7	53.7				
17	33.6	40.0	45.2	49.4	52.7	55.2	56.7	57.2	57.2	57.2	57.2	57.2	57.2	57.2	57.2	57.2	57.2			
18	35.4	41.9	47.2	51.6	55.1	57.8	59.6	60.5	60.6	60.6	60.6	60.6	60.6	60.6	60.6	60.6	60.6	60.6		
19	37.2	43.8	49.2	53.7	57.4	60.3	62.5	63.7	64.0	64.0	64.0	64.0	64.0	64.0	64.0	64.0	64.0	64.0	64.0	
20	39.0	45.7	51.2	55.8	59.7	62.8	65.2	66.7	67.4	67.5	67.5	67.5	67.5	67.5	67.5	67.5	67.5	67.5	67.5	67.5

Table XII. Bit-size of graphical password space when limited to strokes greater than length 1 on a 5×5 grid. Values are given for total length at most L_{max} with at most X strokes.

8. APPENDIX C - METHOD TO QUANTIFY S_1 AND S_2

To derive our results, many (equivalent) counting methods are possible. Our definitions in §3.2 and §3.3 define what DAS passwords are in the Class 1 and 2 probable password spaces. For completeness, §8.1 and §8.2 detail our methods to approximate the size of S_1 and S_2 respectively.

8.1 Overview of our Method to Quantify S_1

Our general approach to quantify $|S_1|$ (recall Definition 11) is to determine how many DAS passwords in S_1 are of length at most a given maximum password length L_{max}. The composite strokes of each password in S_1 have defining strokes that connect a given virtual start and end point in the symmetric area. Counting all passwords of length at most L_{max} and defining passwords in terms of strokes follows Jermyn et al. [Jermyn et al. 1999]; however, our method for defining the set of strokes of a given length is entirely different, and only symmetric strokes are included in the set.

The key points of our method of quantifying $|S_1|$ are discussed in this section (for more details see §8.1.1, §8.1.2, and §8.1.3). Generally, the base formula for defining the set of strokes does the following: for every possible virtual start point $s = (x, y)$, and end point $e = (x, y)$ in a given $W \times H$ grid, we determine the number of ways to draw a symmetric stroke (symmetric about any valid axis in A) of length ℓ modeled by a defining stroke that joins s to e. The reason for specifying s and e is so we know explicitly whether s and/or e are potentially continuous (recall §4.2.2) in order to enumerate the continuous and enclosed cases.

The number of defining strokes from s to e is enumerated by examining the number of permutations of up, down, left, and right movements that join s to e while remaining within the bounds of the symmetric area, for all valid axes in A. The primary considerations in this method are: path *diversions*, and the amount of *room* between the current position and the bounds in every direction within a given symmetric area.

The number of possible diversions for a given s, e, ℓ, and axis $a \in A$ is based on the difference between the desired defining stroke length $\frac{\ell}{2}$ and the minimum length path (stroke with the least number of cells) that joins s to e. The difference between $\frac{\ell}{2}$ and the minimum length path required to join s to e is the number of extra cells that should exist in the stroke from s to e that divert from the minimum length path. In order for the defining stroke with diversions to connect s to e, each diversion must be paired with a cell crossing in the opposite direction to reconnect with the minimum length path. An example of a diversion is provided in Fig. 17.

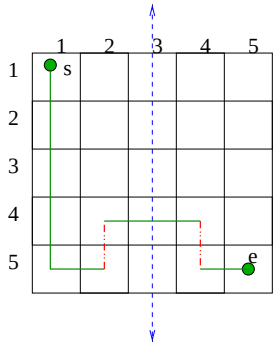


Fig. 17. Defining stroke with a diversion (diversion is marked in a dashed line).

Room is the number of cell crossings in a given direction that can occur from s, before the defining stroke goes out of the symmetric area bounds in question. If at any point in the defining stroke, the number of left cell crossings exceeds the number of right cell crossings by more than the amount of left room, the defining stroke is invalid. The use of room in other directions is analogously defined. Given a starting point s and the symmetric area, we know the amount of available room in each direction. For example, in Fig. 18, right room = 2, left room = 1, top room = 1, and bottom room = 3.

When ℓ is even, the symmetric strokes enumerated for a given s and e are a combination of the disjoint, continuous, and enclosed cases. When ℓ is odd, the symmetric strokes enumerated for a given s and e are a combination of the continuous and enclosed cases. These sets intersect due to the nature of our counting

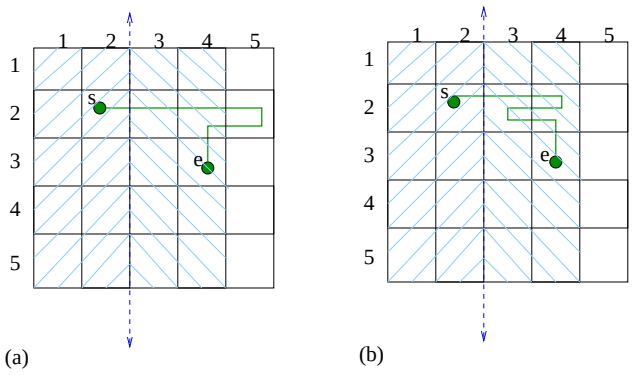


Fig. 18. (a) Defining stroke that goes outside of the symmetric area when right room = 2, number of right cell crossings = 3. (b) Defining stroke with the same amount of right room and cell crossings that remains within the symmetric area.

method. In determining the size of an overall memorable password space, overlaps must be accounted for to avoid double-counting. Fig. 19 gives a representative illustration of how the strokes intersect with one another when ℓ is even (more specifically, $\ell = 12$); when ℓ is odd, the disjoint case is void (recall §4.2.1).

If a symmetric stroke z has a defining stroke z^D that is independently symmetric, and a reflected stroke z^R that is independently symmetric, z is the same as two independent symmetric strokes, which can be independently included in S_1 (i.e., they are either continuous symmetric strokes or enclosed symmetric strokes). Thus, we must ensure that all disjoint case symmetric strokes that have symmetric defining strokes are subtracted from the count. Our method to subtract these symmetric defining strokes is detailed in §8.1.3.

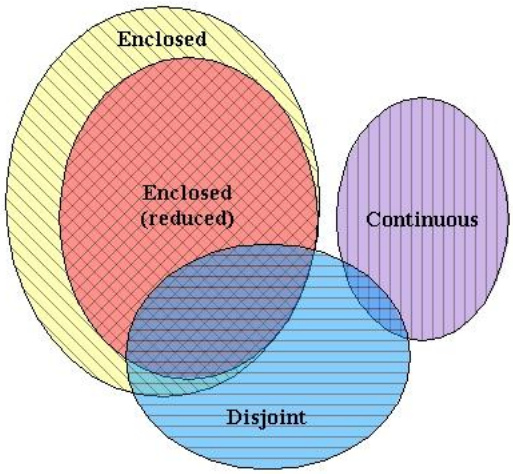


Fig. 19. Relationship between different cases of symmetric strokes of length 12 on a 5×5 grid. Enclosed (reduced) refers to the enclosed case, after removing double-counting. (Note: a stroke of length 12 implies a password of length at least 12.)

Some enclosed shapes will be double-counted by this method since an enclosed stroke may be symmetric about both a horizontal axis $a \in A_h$ and a vertical axis $a \in A_v$ (e.g., Fig. 20). The double counting is due to counting all possible start/end points of an enclosed shape case. The enclosed shapes that are symmetric about an $a \in A_h$ and an $a \in A_v$ can be identified as those whose defining strokes are symmetric. We identify and subtract those defining strokes that are symmetric continuous cases (including those whose start point is potentially continuous). This involves determining the candidate axis a_c and all candidate mid-points m that will produce a continuous symmetric stroke from s to e. These defining strokes must be identified only

once; we identify them when $a \in A_h$. In Fig. 20, the symmetric defining stroke (about the horizontal axis) is indicated as the circular dashed line.

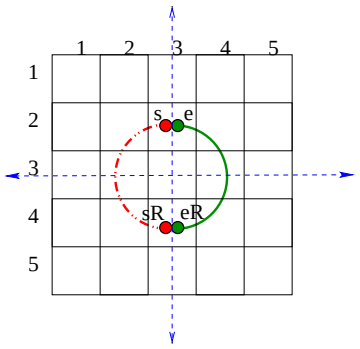


Fig. 20. An enclosed shape symmetric about a horizontal and vertical axis; it would be double-counted by our approach, without explicit subtraction.

We are aware of other smaller cases of overlap that we have experimentally determined as insignificant to the overall results. One such case is when the reflected stroke is identical regardless of whether it is drawn from s^R to e^R or from e^R to s^R, but is not an enclosed case (e.g., lines that repeat over each other more than once). Additionally, there is a smaller set of defining strokes that result in the same symmetric stroke when reflected about one horizontal and another vertical axis, which occurs when the second half of the defining stroke is a 180 degree rotation of the first half of the stroke. There may be other small cases of overlap that we are presently unaware of, but we believe that the set we used will account for any overlap of significant impact on the overall result.

8.1.1 *Basic Structure.* Here we provide additional details regarding quantifying S_1. The analysis method begins as in [Jermyn et al. 1999] with $\pi(L_{max})$ and *P(L)*:

$$\pi(L_{max}) = \sum_{L=0}^{L_{max}} P(L) \tag{2}$$

$$P(L) = \begin{cases} \sum_{\ell=1}^{L} N(\ell) \times P(L-\ell) & \text{if } 1 \le L \le L_{max} \\ 1 & \text{if } L = 0 \end{cases} \tag{3}$$

P is the function that defines the cardinality of the set of passwords in S_1 of length less than or equal to L. P is defined recursively in terms of N, which defines the cardinality of the set of symmetric strokes of length ℓ. Thus, P, the number of passwords of length less than or equal to L, is defined as the number of passwords consisting solely of symmetric strokes of length L-ℓ multiplied by the number of symmetric strokes of length ℓ, for each value of ℓ. L_{max}, W, and H are assumed constant once set.

$$N(\ell) = \sum_{s,e \in W \times H} strokes(\ell, s, e) \tag{4}$$

N says that for every possible start point $s = (x, y)$, and end point $e = (x, y)$ in the given $W \times H$ grid, determine the number of ways to draw a symmetric stroke of length ℓ based on a defining stroke that joins s to e. The number of ways to draw a symmetric stroke that is based on a defining stroke from s to e, where the reflection is about any valid axis in A (i.e., an axis defining a symmetric area that both s and e lie within) is given in *strokes(ℓ, s, e)*.

The cardinality of the set of valid symmetric strokes modeled by a fixed s and e is determined by *strokes*(ℓ, s, e), $\forall a \in A$.

$$strokes(\ell, s, e) = \begin{cases} \sum_{a \in A} 2 \times (paths(\ell, s, e, a_{center}, a) - overlap(\ell, s, e, a)) + cont(\ell, s, e, a_{center}, a) & \text{if case 1} \\ \sum_{a \in A} (paths(\ell, s, e, a_{center}, a) - overlap(\ell, s, e, a)) + cont(\ell, s, e, a_{center}, a) & \text{if case 2} \\ \sum_{a \in A} cont(\ell, s, e, a_{center}, a) + encl(\ell, s, e, a_{center}, a) & \text{if case 3} \end{cases} \tag{5}$$

where a_{center} is the centermost axis (this axis defines the effective boundaries of the grid – its purpose is explained in §8.1.3). Case 1 occurs when ℓ is even and the reflected stroke is not a stroke of length 1. Case 2 occurs when ℓ is even and the reflected stroke is of length 1 (meaning $e^R = s^R$ and $\ell = 2$). Case 3 occurs when ℓ is odd. The *disjoint case* is covered by cases 1 and 2, the *enclosed case* is covered by case 3, and the *continuous case* is covered by all 3 cases. *paths*, see (6), provides the cardinality of the set of defining strokes from s to e of length $\frac{\ell}{2}$ that remain within the symmetric area defined by a. In case 1, the factor of two is for the number of ways to draw a reflected stroke that is an exact reflection, which is from s^R, the reflection of s, to e^R, the reflection of e, or from e^R to s^R. Note the factor of two is not used in case 2, when the reflected stroke is of length 1 as a stroke from e^R to s^R is the same as a stroke from s^R to e^R. To avoid double-counting from the disjoint case when ℓ is even, the result is reduced by *overlap*, defined in §8.1.3. The relationship between the different symmetric stroke cases that *strokes* counts is displayed in Fig. 19.

By the nature of the disjoint case, the above counting method overlooks some permutations of strokes to form a password. A disjoint case is considered as one symmetric stroke composed of two halves drawn consecutively. We consider permutations in this context, thus any permutations of other symmetric strokes between the two halves of a disjoint case are overlooked. Regardless, we believe the effect that this will have on the overall count will be small, because we have experimentally found that the enclosed and continuous cases are the dominating factors in this method to compute the mirror symmetric password space.

paths counts the set of defining strokes from s to e of length $\frac{\ell}{2}$ that remain within the symmetric area defined by a:

$$paths(\ell, s, e, a_b, a) = \begin{cases} \sum_{i=0}^{\frac{D}{2}} divPaths(d(i, d_x, d_x \leq 0), d(\frac{D}{2} - i, d_y, d_y > 0), d(i, d_x, d_x > 0), d(\frac{D}{2} - i, d_y, d_y \leq 0), a_b, a) & \text{if case 1} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where a_b is passed to define the boundaries of the grid (purpose described in §8.1.3). Case 1 occurs when $D = [\frac{\ell}{2} - (|d_x| + |d_y| + 1)] \geq 0$ is even, explained further below, and s, e, and their reflections are within the symmetric area defined by a. The function d, defined further below, calculates the number of cell crossings for a given direction. The variables in *paths* are defined as:

- $d_x = s_x - e_x$, where $s = (s_x, s_y)$, and $e = (e_x, e_y)$. In words, d_x is the minimum number of horizontal cell crossings that must occur to get from s to e. Note $d_x \leq 0$ represents right cell crossings.
- $d_y = s_y - e_y$ is defined analogously to d_x. Note $d_y \leq 0$ represents down cell crossings.
- D is the difference between the desired defining stroke length $\frac{\ell}{2}$ and the shortest defining stroke length joining s to e. Recall that there must be at least d_x horizontal cell crossings, and d_y vertical cell crossings to join s to e. The shortest defining stroke length joining s to e is $|d_x| + |d_y| + 1$, since the length of a stroke is the number of cell crossings plus 1 (for the start cell). $\frac{\ell}{2}$ is the required length of the defining stroke from s to e to create a symmetric stroke of length ℓ when reflected about a. This means that the length ℓ of the symmetric stroke must be even. Odd length symmetric strokes are handled by the continuous case in *cont* and the enclosed case in *encl*. The difference between the required defining stroke length and the minimum defining stroke length required to obtain a path from s to e is the number of extra cells that should exist in the path from s to e that divert from the minimal path. In order for the path with diversions to connect s to e, each diversion must be

paired with a cell crossing in the opposite direction to reconnect with the minimal path. Thus, D must be even. An example of a diversion is provided in Fig. 17.

For a given s and e that lie within the symmetric area defined by the axis a, *paths* counts all possible ways to split up the diversions between horizontal and vertical. The number of diversion pairs is $\frac{D}{2}$; note that under case 1, D is even. Each iteration of the summation calls *divPaths* with a different combination of horizontal and vertical diversion pairs. For a path from s to e that has i horizontal and $\frac{D}{2} - i$ vertical diversion pairs, there must be i horizontal cell crossings in the direction opposite to the horizontal direction of the minimal path, and $\frac{D}{2} - i$ vertical cell crossings in the direction opposite to the vertical direction of the minimal path. There must be $|d_x| + i$ horizontal and $|d_y| + \frac{D}{2} - i$ vertical cell crossings in the direction of the minimal path. The number of right, left, up and down cell crossings to get from s to e in a path of length $\frac{\ell}{2}$ are defined by passing the appropriate parameters to the function that determines the number of cell crossings for a given direction, *d(n, d_c, b)*:

$$d(n, d_c, b) \;=\; n + b \times |d_c|, \text{ so that } d_c \text{ is only added if condition b holds.}$$

where the variable n represents the number of diversions going in the direction, $|d_c|$ represents the number of cell crossings required in the direction of the minimal path, and b is intended to be the condition that places d_c in the direction.

divPaths counts the number of valid defining strokes of length $\frac{\ell}{2}$ between s and e that lie within the symmetric area defined by the boundary axis a_b, the reflection axis a, and the grid boundaries (given the number of cell crossings to occur in each direction). To precede, we define the method of determining the boundaries of the symmetric area. The right, left, top and bottom boundaries can be determined by passing the appropriate parameters to *bound(a, G, oppBound)*. The variable a is the axis to determine the bound for, G is the maximum grid boundary for the coordinate in question (either W or H), and *oppBound* is the opposite of the current grid boundary for the coordinate in question (either 1 for the right or bottom boundaries, W for the left boundary, or H for the top boundary). For example, on Fig. 18, the right bound = bound(2.5, 5, 1) = 4, left bound = bound(2.5, 5, 5) = 1, top bound = bound(2.5, 5, 5) = 1, bottom bound = bound(2.5, 5, 1) = 5, which bound the shaded symmetric area.

$$bound(a, G, oppBound) = \begin{cases} 2a - oppBound & \text{if } (a \text{ affects this coordinate, and (case 1 or case 2)}) \\ G & \text{if case 3 or case 4} \\ 1 & \text{if case 5 or case 6} \end{cases} \tag{7}$$

where case 1 is when $a \leq \lceil G/2 \rceil$ and $oppBound = 1$, case 2 is when $a > \lceil G/2 \rceil$ and $oppBound = G$, case 3 is when a affects this coordinate and $a > \lceil G/2 \rceil$ and $oppBound = 1$, case 4 is when a does not affect this coordinate and $oppBound = 1$, case 5 is when a does not affect this coordinate and $oppBound = G$, and case 6 is when a affects this coordinate and $a \leq \lceil G/2 \rceil$ and $oppBound = G$.

Next we define the concept of *room*, which is the number of cell crossings in a given direction that can occur from s, before the defining stroke goes out of the symmetric area in question. If at any point in the path, the number of left cell crossings exceeds the number of right cell crossings by more than the amount of left room, the defining stroke is invalid. The use of room in other directions is analogously defined. The definition of room for symmetric area boundaries is:

$$room(a, s_c, oppBound, G) \;=\; \begin{cases} |bound(a, G, oppBound) - s_c| & \text{if oppBound = 1} \\ |s_c - bound(a, G, oppBound)| & \text{if oppBound = G} \end{cases}$$

where the variable a is the axis, s_c is the start point coordinate of the coordinate in question, and G and *oppBound* are defined as in *bound(a, G, oppBound)*. For example, in Fig. 18 (recall §8.1), right room = $room(2.5, 2, 1, 5) = 2$, left room = $room(2.5, 2, 5, 5) = 1$, top room = $room(2.5, 2, 5, 5) = 1$, and bottom room = $room(2.5, 2, 1, 5) = 3$.

A defining stroke remains within the symmetric area if the displacement in the direction of the boundary is never more than the amount of room between the start point and that boundary. Otherwise, the defining

stroke will go out of bounds as in Fig. 18a (recall §8.1).

$$divPaths(r,u,f,d,s,a_b,a) = \begin{cases} 1 & \text{if case 1} \\ valid(r, room(a_i, si_x, 1, W_i), f, room(a_i, si_x, W_i, W_i)) & \text{if case 2} \\ valid(u, room(a_i, si_y, H_i, H_i), d, room(a_i, si_y, 1, H_i)) & \text{if case 3} \\ & \\ valid(r, room(a_i, si_x, 1, W_i), f, room(a_i, si_x, W_i, W_i)) \ \times & \\ valid(u, room(a_i, si_y, H_i, H_i), d, room(a_i, si_y, 1, H_i)) \ \times & \\ C((r+f+u+d),(r+f)) & \text{otherwise} \end{cases} \tag{8}$$

where W_i, the width of the "virtual grid" (the symmetric area defined by a_b), is the difference (plus 1) between the right and left bounds of the symmetric area provided by a_b. H_i, the height of the "virtual grid", is the difference (plus 1) calculated between the upper and lower bounds of the symmetric area provided by a_b. a_i and si are the results of shifting a and s such that their coordinates are positioned within the "virtual grid" bounded by a_b. This is intended for when overlap is calculated (see §8.1.3). It is to ensure that the strokes counted are valid within the symmetric areas of both a_b and a.
Also referring to *divPaths*, case 1 is when r, f, u, and d are all 0 (stroke of length 1), case 2 is when u and d are 0 (only horizontal movement), and case 3 is when r and f are 0 (only vertical movement), and C is the number of combinations.

Given the number of cell crossings in each direction, the sum of which should equal $\frac{\ell}{2} - 1$, *divPaths* determines the number of valid orderings of these $\frac{\ell}{2} - 1$ cell crossings. There are four cell crossing directions that may occur: left, right, up or down. The number of each of these cell crossing directions is passed into *divPaths* from *paths*. The function makes two calls to *valid* described in (9). The first call returns the number of sets of horizontal cell crossings that stay within the horizontal boundaries, and the second call returns the number of sets of vertical cell crossings that stay within the vertical boundaries. Each set does not affect the validity of the other, so they may be merged with each other with no restrictions. Since the orderings of the sets is determined within *valid*, the ordering of the horizontal and vertical sets should not be altered. Each horizontal set should be paired with each vertical set, and then the number of combinations of ways to merge the two are determined by C(r+f+u+d , r+f).

$$valid(dir, dirRoom, oppDir, oppDirRoom) = \begin{cases} 0 & \text{if (dir=0 and oppDir=0)} \\ 1 & \text{if (dir=0 xor oppDir=0)} \\ c_d \times valid(dir-1, dirRoom-1, oppDir, oppDirRoom+1) + & \\ c_o \times valid(dir, dirRoom+1, oppDir-1, oppDirRoom-1) & \text{otherwise} \end{cases} \tag{9}$$

Where:

$$c_d = \begin{cases} 1 & \text{if } (dirRoom > 0) \\ 0 & \text{otherwise} \end{cases}$$

$$c_o = \begin{cases} 1 & \text{if } (oppDirRoom > 0) \\ 0 & \text{otherwise} \end{cases}$$

Valid recursively counts the number of valid sets of either horizontal or vertical cell crossings depending on which are passed in the parameters. The amount of cell crossings and room are considered in both the direction and the opposing direction. If there are no cell crossings in either direction, there are no ways to order them. If there are no cell crossings in one direction, there is clearly only one ordering of the two cell crossing directions. Otherwise, if the amount of room in one direction is greater than zero, a cell crossing in that direction may occur. Once the cell crossing is performed, the number of remaining cell crossings and room in that direction reduce by one, and the amount of room in the opposing direction increases by one. The changing of the amount of room and cell crossings in a given direction is represented in the recursive call to valid to find the number of ways that sub-stroke can be drawn given the new set of constraints. This method represents the changing of the set of constraints and the changing of the number of required cell crossings left as the stroke continues. An example is provided in Fig. 21.

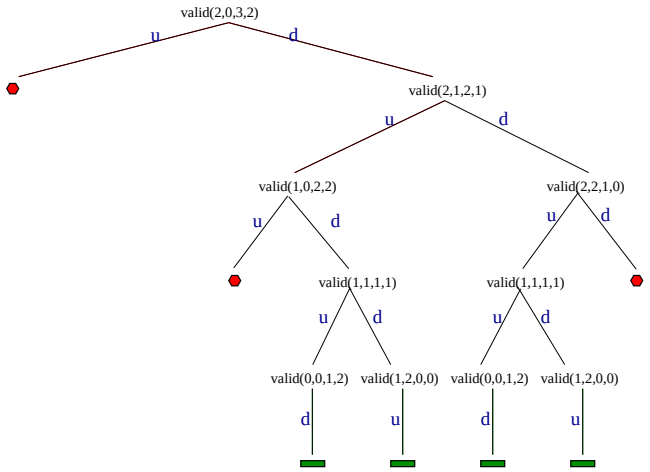


Fig. 21. The tree for a 3×3 grid, vertical axis= 1, $s = (1, 1)$, $e = (1, 2)$, $u = 2$, $d = 3$, $r = 0$, $f = 0$, upper room = 0, down room = 2. The result indicates that there are 4 valid paths under these conditions, namely: dudud, ddudu, duddu, and dduud.

8.1.2 *Quantifying Continuous and Enclosed Cases.* Here we provide details regarding the calculation of the continuous and enclosed cases from (5). Until this point, it has been assumed that the defining stroke is exactly reflected to obtain a symmetric stroke. Recall the additional cases discussed in §4.2.2 of the continuous case and the enclosed case. Note that for simplicity of the formulas, just the equations using A_v are shown. The equation using A_h is defined analogously.

Recall from §4.2.2 the definition of the continuous and enclosed cases. When a cuts a set of cells, $e = e^R$, so is not reflected in the reflected stroke, causing the length of the resulting symmetric stroke to be $\ell - 1$, e.g., Fig. 9b (recall §4.2.1). When a is on a grid line, e is a neighbour of e^R, so the length of the resulting symmetric stroke is ℓ, e.g., Fig. 9c.

$$cont(\ell, s, e, a_b, a) = \begin{cases} paths(\ell, s, e, a_b, a) & \text{if case 1} \\ paths(\ell + 1, s, e, a_b, a) & \text{if case 2} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where case 1 occurs when ℓ is even, a is on a grid line, e is in a cell adjacent to a. Case 2 occurs when ℓ is odd, a cuts a set of cells, e is in a cell that is cut by a, and s is not in a cell that is cut by a since this is handled by the enclosed shape case (defined in (11)).

The number of continuous symmetric strokes is simply the number of valid defining strokes from s to e, of length $\frac{\ell}{2}$ when a is on a grid line and ℓ is even, or of length $\frac{\ell+1}{2}$ when a is through cells and ℓ is odd. All other cases do not model the continuous case.

The enclosed shape case (recall §4.2.2) occurs when the cells in which s and e reside either contain, or have neighbours that contain s^R and e^R in the reflected stroke. We define *encl* from (5) as follows:

$$encl(\ell, s, e, a_b, a) = \begin{cases} enclCalc(\ell - 1, s, e, a_b, a) & \text{if case 1} \\ enclCalc(\ell + 1, s, e, a_b, a) & \text{if case 2} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where case 1 occurs when ℓ is odd, a is on a grid line, both s and e are adjacent to a, and $\ell > 1$. Case 2 occurs when ℓ is odd, a cuts a set of cells, and both s and e are in cells cut by a.

As with the continuous case, the enclosed shape case is different depending on whether the axis a cuts a set of cells, or is on a grid line. If a cuts a set of cells, since $e = e^R$, with no pen-up between, e^R is not included in the encoding, and s and s^R lie in the same cell, enclosing the shape as shown in Fig. 10a. To achieve an enclosed shape of length ℓ for case 1, a defining stroke length of $\frac{\ell-1}{2}$ is required. To achieve an enclosed shape of length ℓ for case 2, a defining stroke of length $\frac{\ell+1}{2}$ is required. If a is on a grid line, both s and e would be reflected and s would be repeated to enclose the shape as in Fig. 10b. The difference in the

required defining stroke length is the reasoning for the first two cases of *encl*. All other cases do not model the enclosed case.

enclCalc says that for all possible enclosed symmetric strokes of length ℓ_m, the modified ℓ passed from *encl*, there are a number of possible starting points for it to be drawn from. Since it is guaranteed that at least the start point has been repeated for the shape to be enclosed, there are at most ℓ_m (when a cuts cells) or $\ell_m - 1$ (when a is on a grid line) unique cells that the path could start and end from; however, note that a defining stroke's reflected stroke going in the same direction will be independently counted as a separate defining stroke. Thus, only the values in the defining stroke (with the exception of s in the case where a cuts cells) are counted, which is accounted for by *validStPts*. The reason for *validStPts*' exception of s when a cuts cells is because $s = s^R$, thus s will be counted when the defining stroke is from e^R to s^R. If there is more than one repeated cell in the symmetric stroke, this factor will be an overestimate; however, we expect this simplification to have negligible effect on the overall analysis.[8] This yields:

$$enclCalc(\ell_m, s, e, a_b, a) = \{\ (paths(\ell_m, s, e, a_b, a) - enclOverlap(\tfrac{\ell_m}{2}, s, e, a)) \times validStPts(\ell_m, a) \tag{12}$$

where:

$$validStPts(\ell_m, a) = \begin{cases} \frac{\ell_m}{2} & \text{if } a \text{ on a grid line and } \ell_m \geq 2 \\ (\frac{\ell_m}{2} - 1) & \text{if } a \text{ cuts cells and } \ell_m \geq 4 \\ 1 & \text{otherwise} \end{cases} \tag{13}$$

and where *enclOverlap* is defined further below. The use of ℓ_m is to emphasize that it is not the same as the ℓ passed to *encl* (it has been modified – note that the modification ensures ℓ_m is always an even number).

There are some enclosed shapes that will be double counted by $paths(\ell, s, e, a) \times validStPts$, since $A = A_h \cup A_v$, an enclosed stroke may be symmetric about one $a \in A_h$ and one $a \in A_v$ as shown in Fig. 20 (recall §8.1). The enclosed shapes that are symmetric about an $a \in A_h$ and an $a \in A_v$ can be identified as those whose defining strokes are symmetric. Thus, the problem is identifying those defining strokes that are symmetric continuous cases including those whose starting point is on the axis a. This involves determining the candidate axis a_c and all candidate mid-points m that will produce a continuous symmetric stroke from s to e. The symmetric continuous cases from s to e are identified by $enclOverlap(\ell_{md}, s, e, a)$. These defining strokes must be identified only once, so we identify them only when $a \in A_h$. In Fig. 20, the symmetric defining stroke (about the horizontal axis) is indicated as the circular dashed line.

$$enclOverlap(\ell_{md}, s, e, a) = \begin{cases} \sum_{a_c \in A_v} \sum_{m_y \in bound(a,H,H),\ldots,bound(a,H,1)} sDefStrokes(\ell_{md}, s, m_y, a, a_c) & \text{if case 1} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

where case 1 occurs when $a \in A_h$ and $s^R = e$ about a_c, *sDefStrokes* is defined below, and m_y is the y component of a candidate mid-point for the continuous defining stroke from s to e. Recall that *bound* was defined in (7). Note that ℓ_{md} is used to emphasize that this value is $\frac{\ell_m}{2}$.

In words, *enclOverlap* determines all defining strokes connecting s and e that are continuous symmetric strokes about a candidate axis $a_c \in A_v$.

$$sDefStrokes(\ell_{md}, s, m_y, a, a_c) = \begin{cases} \sum_{a_m = \lfloor a_c \rfloor}^{\lceil a_c \rceil} paths(\ell_{md}, s, m = (a_m, m_y), a, a_c)) & \text{if case 1} \\ 0 & \text{if case 2} \\ paths(\ell_{md} + 1, s, m = (a_c, m_y), a, a_c)) & \text{if case 3} \\ 0 & \text{if case 4} \end{cases} \tag{15}$$

where case 1 occurs when a_c on a grid line and ℓ_{md} even, case 2 occurs when a_c on a grid line and ℓ_{md} odd, case 3 occurs when a_c cuts cells and ℓ_{md} odd, and case 4 occurs when a_c cuts cells and ℓ_{md} even.

[8] Without tracking the cells that join s to e that overlap for each case, there is no simple way to determine the exact number of repeated cells for this factor. Experimentally, setting *validStPts* to one had a negligible effect.

In words, *sDefStrokes* determines the number of defining strokes from s and e that are continuous symmetric strokes about a candidate axis a_c. This is done by determining all possible paths between s and all possible mid-points m that would produce a continuous defining stroke that is symmetric about a_c from s to e. A possible m is a point in the symmetric area provided by a, that is in a cell cut by a_c when a_c cuts cells, or adjacent to a_c when a_c is on a grid line. Note that the second case where a_c is on a grid line and ℓ_{md} is odd cannot produce a continuous defining stroke, as if the axis of reflection is on a grid line, its defining stroke is of even length since it is exactly reflected as shown in Fig. 9c (recall §4.2.1). Enclosed defining strokes can have odd length about a_c where a_c is on a grid line, but this defining stroke would not produce the same symmetric stroke when reflected about both a and a_c. The stroke will not be symmetric .Also note that the fourth case cannot produce a continuous defining stroke as in order for a continuous stroke to be of even length, the axis must be on a grid line.

sDefStrokes is different than the overlap counted in §8.1.3 in that it does not count all enclosed defining strokes, but only those enclosed cases when a_c cuts cells. We do not include those enclosed defining strokes where a_c is on a grid line since s and e are on one side of a_c, thus the line connecting e to e^R will only be on one side of a, producing a different shape than when reflected about a (e.g., Fig. 22).

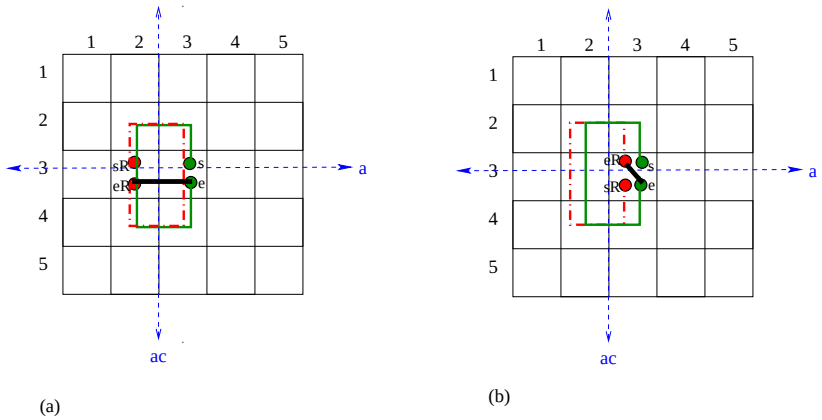


Fig. 22. Enclosed defining strokes (e.g., the thinner solid line here) produce a different enclosed shape when reflected about (a) a_c or (b) a. The thicker solid line shows how e connects to e^R in each case.

8.1.3 *Removing Double-Counting.* Now we discuss the overlap from (5). Overlap between the sets in the case where ℓ is even can occur due to the nature of our counting method. If a symmetric stroke z has a symmetric defining stroke z^D, and thus a symmetric reflected stroke z^R, z is the same as two independent symmetric strokes, which can be independently included in S_1 by *cont* or *encl* with length $\frac{\ell}{2}$. Thus, we must ensure that all symmetric strokes found in *paths* that have symmetric defining strokes, i.e., they are either continuous symmetric strokes or enclosed symmetric strokes, are subtracted from the set. The set of symmetric defining strokes is defined in *overlap*. Note that we describe overlap in terms of the culprit axis $a_c \in A_v$ for simplicity, as the calculation is performed analogously when $a_c \in A_h$.

$$overlap(\ell, s, e, a) = \begin{cases} \sum_{a_c \in A_v} \sum_{m_y \in H} enclDStrokes(\ell, s, m_y, a_c) & \text{if case 1} \\ \sum_{m_y \in H} contDStrokes(\ell, s, m_y, culpritAxis(s, e)) & \text{if case 2} \end{cases} \tag{16}$$

where case 1 occurs when $s = e$ and a_c and its reflection about a are both in the symmetric area defined by a. Case 2 is similarly defined: it occurs when $s_x \neq e_x$ and a_c (as returned from *culpritAxis*) and its reflection about a are in the symmetric area defined by a. *culpritAxis*, *contDStrokes*, and *enclDStrokes* are defined below. Verbally, *overlap* defines the set of defining strokes for disjoint symmetric strokes that are either continuous or enclosed in their own right.

$$culpritAxis(s,e) = \tfrac{s_x+e_x}{2} \tag{17}$$

$$contDStrokes(\ell, s, m_y, a, a_c) = \begin{cases} cont(\frac{\ell}{2}, s, m = (a_c, m_y), a, a_c) & \text{if } a_c \text{ cuts cells} \\ cont(\frac{\ell}{2}, s, m = (\lceil a_c \rceil, m_y), a, a_c) + cont(\frac{\ell}{2}, s, m = (\lfloor a_c \rfloor, m_y), a, a_c) & \text{if } a_c \text{ on grid line} \end{cases} \tag{18}$$

$$enclDStrokes(\ell, s, m_y, a, a_c) = \begin{cases} encl(\frac{\ell}{2}, s, m = (a_c, m_y), a, a_c) & \text{if } a_c \text{ cuts cells} \\ encl(\frac{\ell}{2}, s, m = (\lceil a_c \rceil, m_y), a, a_c) + encl(\frac{\ell}{2}, s, m = (\lfloor a_c \rfloor, m_y), a, a_c) & \text{if } a_c \text{ on grid line} \end{cases} \tag{19}$$

contDStrokes and *enclDStrokes* call *cont* and *encl* respectively with the possible values of m that may produce a continuous or enclosed stroke about a_c. When a_c is on a grid line, m must be in a cell adjacent to the axis, meaning that it could have a horizontal coordinate on either side of the axis. When a_c cuts cells, m must be in a cell that is cut by a_c. Note that when *enclDStrokes* uses *encl*, *validStPts* will always be 1 (as we only want to consider when the enclosed case has the specified s value). Also note that it is this use of a culprit axis (that still must be within the symmetric area of a) that requires the extra boundary adjustment in *divPaths*. This boundary adjustment ensures that the resulting stroke is within the symmetric area of both a_c and a. This is an issue since the "culprit" axis that the defining stroke is symmetric about it its own right may provide symmetric areas whose boundaries lie outside of the symmetric area of the axis we are counting disjoint cases about. If not considered, it would result in an over-count of the real amount of overlap. This is why the boundary axis is introduced, which defines a "virtual grid". If there is no axis whose symmetric area should bound the stroke (as in the call from *strokes*), a_{center} is used as it is the same as using the grid boundaries, thus the "virtual grid" is the same as the actual grid.

We are aware of other smaller cases of overlap that we have experimentally determined as insignificant to the overall results (for details see discussion in §8.1). There may be other small cases of overlap that we are unaware of at this point, but we believe that the set defined in *overlap* will account for any overlap that will have a significant impact on the overall result.

8.2 Overview of our Method to Quantify S_2

Our general approach to quantify the relationship between DAS and the stroke-count is to determine how many DAS passwords are of length at most a given maximum password length L_{max}, with a maximum stroke-count of X. Counting all passwords of length at most L_{max} follows [Jermyn et al. 1999]; we add the restriction of the maximum stroke-count X within the function P (see below).

We modify the function *P(L)* (recall (3) in 8.1.1) that counts the number of passwords of length $\leq L$ (where $1 \leq L \leq L_{max}$ is the password length), to limit the stroke-count in each password to at most X.

$$P(L,X) = \begin{cases} 0 & \text{if } X = 0 \text{ and } L > 0 \\ 1 & \text{if } X \geq 0 \text{ and } L = 0 \\ \sum_{\ell=1}^{L} N(\ell) \cdot P(L-\ell, X-1) & \text{otherwise} \end{cases} \tag{20}$$

P defines the cardinality of the set of passwords with X or fewer strokes, of total password length at most L. P is defined recursively in terms of *N(ℓ)* (recall (4) in §8.1.1), which gives the number of *strokes* of length ℓ.