The Design and Analysis of Graphical Passwords

Ian Jermyn* Alain Mayer† Fabian Monrose‡ Michael K. Reiter§ Avi Rubin¶

March 8, 1999

Abstract

In this paper we propose and evaluate new *graphical* password schemes that exploit features of graphical input displays to achieve better security than text-based passwords. Graphical input devices enable the user to decouple the *position* of inputs from the *temporal order* in which those inputs occur, and we show that this decoupling can be used to generate password schemes with substantially larger (memorable) password spaces. In order to evaluate the security of one of our schemes, we devise a novel way to capture a subset of the "memorable" passwords that, we believe, is itself a contribution. In this work we are primarily motivated by devices such as personal digital assistants (PDAs) that offer graphical input capabilities via a stylus, and we describe our prototype implementation of one of our password schemes on such a PDA, namely the Palm Pilot™.

1 Introduction

For the vast majority of computer systems, passwords are the method of choice for authenticating users. It is well-known, however, that passwords are susceptible to attack: users tend to choose passwords that are easy to remember, and often this means that they are also easy for an attacker to obtain by searching for candidate passwords. In one case study of 14,000 Unix passwords, almost 25% of the passwords were found by searching for words from a carefully formed "dictionary" of only 3×10^6 words [Kle90]. This relatively high success rate is not unusual despite the fact that there are roughly 2×10^{14} 8-character passwords consisting of digits and upper and lower case letters alone.

In this paper we explore an approach to user authentication that generalizes the notion of a textual password and that, in many cases, improves the security of user authentication over that provided by textual passwords. We design and analyze *graphical passwords*, which can be input by the user to any device with a graphical input interface. A graphical password serves the same purpose as a textual password, but can consist, for example, of handwritten designs (drawings), possibly in addition to text. The devices by which we are primarily motivated

*Department of Computer Science, New York University, New York, NY, USA; jermyn@cs.nyu.edu
†Bell Laboratories, Lucent Technologies, Murray Hill, NJ, USA; alain@research.bell-labs.com
‡Department of Computer Science, New York University, New York, NY, USA; fabian@cs.nyu.edu
§Bell Laboratories, Lucent Technologies, Murray Hill, NJ, USA; reiter@research.bell-labs.com
¶AT&T Labs—Research, Florham Park, NJ, USA; rubin@research.att.com

Submission to the *8th USENIX Security Symposium.*

are "personal digital assistants" (PDAs) such as the Palm Pilot™, Apple Newton™, Casio Cassiopeia E-10™, and others, which allow users to provide graphics input to the device via a stylus. More generally, graphical passwords can be used whenever a graphical input device, such as a mouse, is available.

To the best of our knowledge, the notion of a "graphical password" is due to Blonder [Blo96]. That work proposed a password scheme in which the user is presented with a predetermined image on a visual display and required to select one or more predetermined positions ("tap regions") on the displayed image in a particular order to indicate his or her authorization to access the resource. Beyond this proposal, however, [Blo96] did not further explore the power of graphical passwords or argue security for its particular proposal.

In this paper we considerably advance the theory and practice of graphical passwords. We take as a main criterion the need to evaluate graphical passwords' security relative to that of textual passwords. We design two graphical password schemes that we believe to be more secure than textual passwords (and more secure than the scheme of [Blo96]), and we employ novel analysis techniques to make this argument. Moreover, we describe our implementation of one of our graphical password schemes on the Palm Pilot.

The graphical password schemes that we propose derive their strength from the following observation: A graphical interface for providing input enables the user to decouple the *positions* of the inputs from their *temporal order*. This is in contrast to textual passwords input via a keyboard: here, the temporal order in which the user types characters uniquely determines their position in the password. However, in a graphical password, e.g., consisting of several drawn lines, the final position of each line can be determined independently of the temporal order in which the lines are drawn. We show that this independence between input position and order can be used to build interesting new password schemes, and in some cases obtain authentication that is convincingly stronger than textual passwords but not significantly harder to remember.

The first graphical password scheme builds directly on textual password schemes, by enhancing the input of textual passwords using graphical techniques. In this case, if we assume the same underlying distribution on the choice of the password, the graphical password is at least as strong as the textual password that underlies it, and even a conservative estimate of the variations introduced by the graphical input yields a substantial improvement in strength over the purely textual version. We propose and implement a second scheme, called "draw a secret" (DAS), which is purely graphical; the user draws a secret design (the password) on a grid. Here, to argue an improvement over textual passwords, we define a class of DAS passwords that, we believe, captures a small subset of the memorable ones. This class consists of those passwords that can be generated by a short program in a simple grid-based language. We do not argue that every memorable password has a short program to describe it, but that passwords describable by short programs are memorable. We show that even this subset of memorable DAS passwords is larger than the dictionaries of textual passwords to which a high percentage of passwords typically belong.

Throughout this paper we focus on graphical passwords that are exactly repeatable by the

user. This distinguishes our work from all works on graphical pattern recognition of which we are aware (see Section 4), where it suffices for the device to recognize an input as being "sufficiently similar" to—but not necessarily the same as—a previously stored input. Because pattern recognition schemes require the storage of (some representation of) the plaintext password on the device, the password is vulnerable to an attacker who captures and probes the device. In contrast, because graphical passwords are repeatable, our schemes can derive a secret key, e.g., to encrypt and decrypt files, without need to store the password on the device. This protects both the password and the encrypted content from the attacker if the device falls into the attacker's hands.

The rest of this paper is outlined as follows: In Section 2, we present textual passwords with graphical assistance. In Section 3, we proceed to purely graphical passwords with a scheme called "draw-a-secret" (DAS). Section 3.2 shows our design and implementation of a memo pad encryption scheme based on DAS. Section 3.3 proposes novel ways to analyze and estimate the security of DAS and graphical passwords in general. In Section 4 we overview other password schemes, unrelated to graphical passwords, but putting our work in context. Finally, Section 5 concludes.

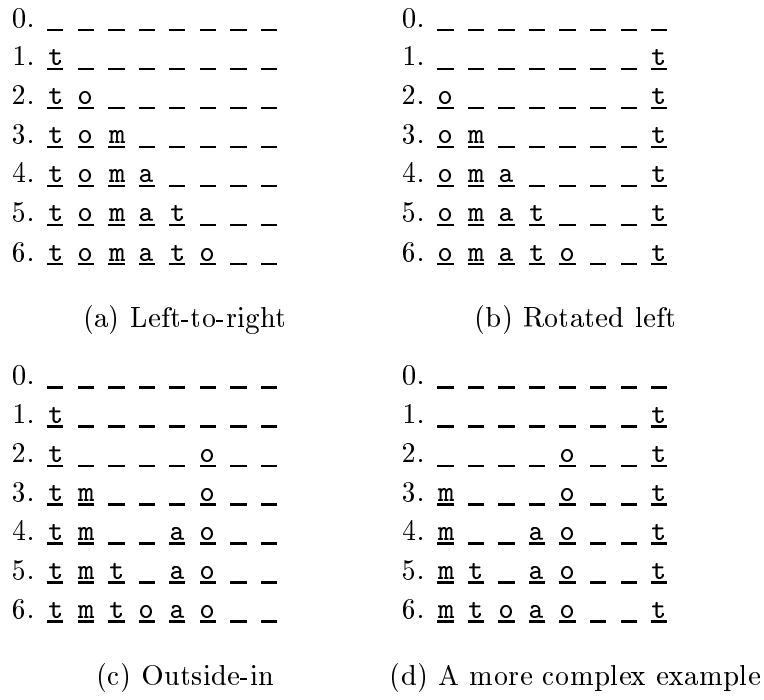
2 Textual Passwords with Graphical Assistance

In this section we present a password selection and input scheme which uses textual passwords augmented by some minimal graphical capabilities that enable the decoupling of temporal order of input and the position in which characters are input. This scheme is interesting because it simply demonstrates the power of graphical input abilities while yielding a scheme that is convincingly stronger than textual passwords are today.

We start by defining a normal, k-character textual password as a total function $\pi : \{1, \ldots, k\} \to \mathcal{A}$, where $\mathcal{A}$ is the set of allowed characters for the textual password. Intuitively, the domain of π denotes the temporal order of inputs, so that the user first enters $\pi(1)$, then $\pi(2)$, and so on. That is, for a password "`tomato`", we have $\pi(1) = \texttt{t}$, $\pi(2) = \texttt{o}$, $\pi(3) = \texttt{m}$, $\pi(4) = \texttt{a}$, $\pi(5) = \texttt{t}$, and $\pi(6) = \texttt{o}$.

Now suppose that the user is presented with a simple graphical input display consisting of, say, eight positions into which to enter a textual password, as illustrated in Figure 1. In this figure, step 0 is the initial row of blanks, and steps 1–6 indicate the temporal order in which the user fills in the blanks; i.e., $\pi(i)$ is entered in row i. The password can be placed in the "normal", left-to-right positions as shown in Figure 1a. Due to the graphical nature of the input interface, however, the user could enter the password in other positions, as well. For example, Figure 1b shows a modification in which the user enters the password in a left-to-right manner, but starting from a different initial position than the leftmost. Figure 1c shows entering the password in an "outside-in" strategy. And, of course, these variations can be combined in the obvious way, as shown in Figure 1d.

Formally, a k-character graphical password in this scheme can be defined by a total function $\pi' : \{1, \ldots, k\} \to \mathcal{A} \times \{1, \ldots, m\}$, where $m \geq k$ is the number of positions into which characters



(a) Left-to-right (b) Rotated left

(c) Outside-in (d) A more complex example

Figure 1: Variations on inputting `tomato`. The word `tomato` can be input in the "normal" left to right manner as shown in (a). Step 0 is the initial row of blanks, and steps 1–6 indicate the temporal order in which the user fills in the blanks. In addition, however, the user can vary the position of the letters in `tomato`. Figure (b) demonstrates shifting the input left by one, (c) represents an outside-in input strategy, and (d) is the combination of these.

can be entered ($m = 8$ in Figure 1). If $\pi'(i) = (c, j)$, then this means that the i-th entry (temporally) is the character c in position j. A conventional textual password π, entered in the standard left-to-right way, can be expressed in this scheme as a graphical password π' where $\pi'(i) = (\pi(i), i)$. But as shown in Figure 1, more generally we can have variations π' in which $\pi'(i) = (\pi(i), j)$ and $i \neq j$. In fact, it is easy to see that each k-character conventional password π yields $m!/(m-k)!$ graphical passwords π', and indeed this is the factor by which the size of the graphical password space exceeds the k-character conventional password space. This can be a relatively large number: e.g., for $k = 8$ and $m = 10$, this factor is approximately 2×10^6.

Of course, there are far fewer than 2×10^6 variations of each 8-character password that are memorable for human users. However, it is easy to derive a convincing lower bound on the improvement this achieves over a conventional password scheme. It is conservative to assume that the m positional rotations of a password, plus perhaps a handful of others (e.g., reversal, outside-in, inside-out, evens-then-odds, odds-then-evens), and combinations thereof, are memorable, because the choices of position involved in these cases can be derived from simple algorithms that are more memorable than the positions themselves. (We will return to this characteristic of memorability in the next section.) The attacker's work load will thus be increased by a factor of at least m. An important feature of this scheme is that it is at least as strong as the initial textual password that was chosen by the user, assuming that users do not reduce the size of the space of character sequences that they choose in response to the need to

remember a positional order.

There are a number of steps that we can take to make this scheme more usable. First, to maximize the ease of inputting passwords with varied position, each character should be echoed once the user places it in a position, at least with a nondescript character (e.g., "$*$") but preferably with the letter itself. This is a departure from most password-input interfaces, which echo at most a nondescript character in order to protect the password from onlooking persons. However, for the platforms by which we are primarily motivated, i.e., hand-held PDAs such as the Palm Pilot, it is much easier to shield the screen from onlookers entirely. Going further, the interface might allow the user to first enter the password "normally" (left-to-right), and then drag each character to its final position in the desired temporal order.

Inevitably, there are numerous variations on the scheme presented here. One direction includes arranging the k input positions in some other way than a straight line (e.g., a grid), to promote other variations in position. Rather than pursuing these options here, we instead explore a purely graphical approach.

3 The Draw-a-Secret (DAS) Scheme

In this section we present a purely graphical password selection and input scheme, which we call "draw a secret" (DAS). In this scheme, the password is a simple picture drawn on a grid. This approach is alphabet independent, thus making it easily accessible for users speaking Chinese, Hebrew, etc. Users are freed from having to remember any kind of alphanumeric string.

The most compelling reason for exploring the use of a picture-based password scheme is that humans seem to possess a remarkable ability for recalling pictures (i.e., line drawings and real objects). The "picture effect", that is, the effect of pictorial and object representations on a variety of measures of learning and memory has been studied for decades [Cal98, She67, PRS68, Sta73, BKD75]. Cognitive scientists and psychologists have shown that there is a substantial improvement of performance in recall and recognition with pictorial representations of to-be-remembered material than for verbal representations.

Superiority in recall of objects over words in immediate recall and over short retention intervals has been demonstrated through a number of experiments. Empirical evidence of the power of pictures over words dates back to the early 1800s; experiments performed by Calkins [Cal98] showed the recall of words declining by 50% or more over a 72 hour retention interval, and recall of objects dropping by less than 20% over the same period. Studies exhibiting strikingly high differences in memory recall of pictures over words have since been replicated on numerous occasions [She67, Sta73, NRW76, BSH77]. Some theories that have been proposed to explain these experimental results are outlined in Appendix A.

3.1 Password Selection and Input

Consider an interface consisting of a rectangular grid of size $G \times G$. Each cell in this grid is denoted by discrete rectangular coordinates $(x, y) \in [1..G] \times [1..G]$. Suppose that the the user is given a stylus with which she can draw a design on this grid. The drawing is then mapped to a sequence of coordinate pairs by listing the cells through which the drawing passes in the order in which it passes through them, with a distinguished coordinate pair inserted in the sequence for each "pen up" event, i.e., whenever the user lifts the stylus from the drawing surface. For example, consider the drawing in Figure 2. Here, the coordinate sequence generated by this drawing is

$$(2,2),(3,2),(3,3),(2,3),(2,2),(2,1),(5,5)$$

where $(5,5)$ is the distinguished "pen up" indicator. If there were a second stroke in this example, then its sequence would be appended to the end of the sequence above, and similarly for subsequent strokes. In this way, we divide the space of possible drawings into equivalence classes, two drawings being equivalent if they have the same encoding, or in other words if they cross the same sequence of grid cells, with the breaks between strokes occurring in the same places.

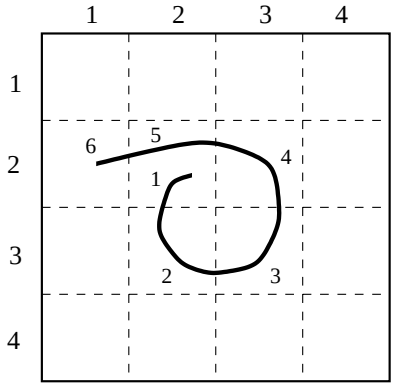


Figure 2: Input of a graphical password on a 4×4 grid. The drawing is mapped to a sequence of coordinate pairs by listing the cells in the order which the stylus passes through them, with a distinguished coordinate pair inserted in the sequence whenever the stylus is lifted from the drawing surface.

First we give some terminology. We define the neighbors, $\mathcal{N}_{(x,y)}$, of a cell (x, y) to be the subset of the set of cells $\{(x-1, y), (x+1, y), (x, y-1), (x, y+1)\}$ whose elements exist in the grid. We then define a stroke to be a sequence of cells $\{c_i\}$, in which $c_i \in \mathcal{N}_{c_{i-1}}$, and which does not contain a "pen up" event. A password is then defined to be a sequence of strokes separated by "pen up" events. The length of a stroke is the number of coordinate pairs it contains, while the total length of a password is the sum of the lengths of its component strokes (excluding the "pen up" characters).

As with the scheme of Section 2, this scheme is most viable if the user's strokes are echoed as curves while they are drawn. Again we appeal to the maneuverability of the devices we are targeting (i.e., PDAs) to support the restriction that the user must shield the input display from onlookers.

Our requirement of repeatability constrains the parameters of this scheme. As long as the user's current drawing lies in the same equivalence class as the original drawing, she has successfully repeated a chosen password. In general, this gives the user sufficient tolerance when (involuntarily) varying the drawing, provided that the cells of the grid are not too small. Indeed, this was the purpose of separating the drawings into equivalence classes to begin with. Difficulties might arise however, when the user chooses a drawing that contains strokes that pass too close to a grid-line. In those cases, the user might vary the drawing in such a way as to change the resulting sequence of coordinates. We consider the following two possibilities to address this problem: (1) The user is offered to view the internal representation, depicting the path of cells, when she chooses a password so that she can confirm which cells were actually touched by the drawing. (2) The system does not accept a drawing which contains strokes that are located "too close" to a grid line. In the implementation, described in Section 3.2, we offer both alternatives.

3.2 Application of DAS: An Encryption Tool for a PDA

Our graphical password schemes are motivated primarily by PDAs that offer graphical input capabilities. We now describe our implementation of a memo pad encryption tool for the Palm Pilot that uses a user-input graphical password to derive the encryption key. Either of the schemes of Sections 2 and 3 could be used to enter the password. Here we illustrate our tool using the DAS scheme, which we have implemented and use regularly.

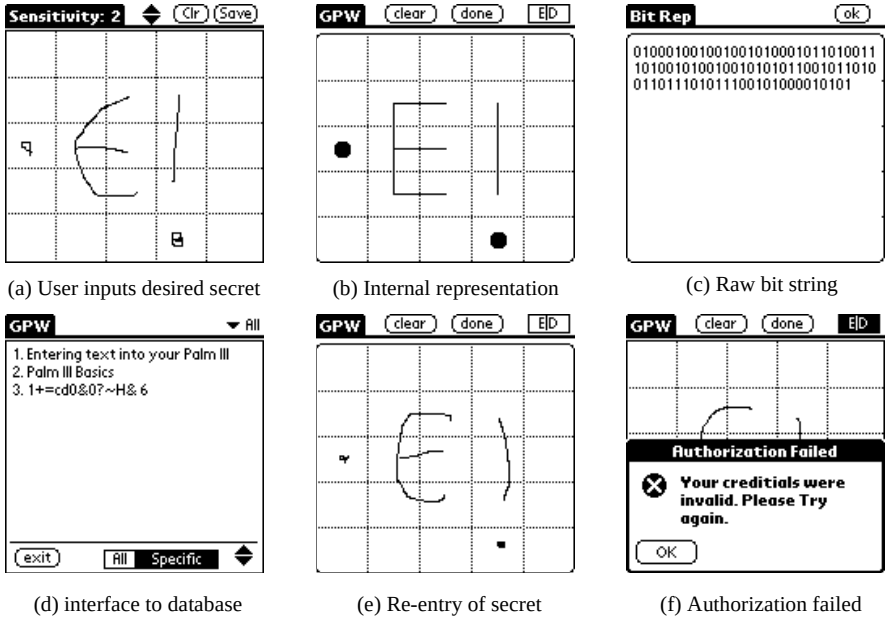
In our tool, an encryption/decryption key is derived from a DAS password (i.e., its sequence of coordinates) as follows: Let $\mathcal{B}$ be a bit string that represents the sequence of coordinates (including the unique "pen up" indicator). Let h denote a cryptographic hash function, such as MD5 or SHA. The key, k, is defined as $h(\mathcal{B}||P)_{128}$, where P is unambiguous padding, resulting from first adding a single 1-bit and then all 0-bits so that the result is a full input block for the hash function h. k results from, e.g., taking the first 128 bits of the output of h. Our key derivation assures that two distinct coordinate sequences are transformed (with high probability) into two distinct, fixed-length keys. A standard symmetric encryption scheme E with k as its symmetric key is used to encrypt and decrypt data records stored on the PDA.

Key selection is as follows: the user is prompted with an empty grid to input the password design. Once the password is entered, a symmetric key k is derived and a pre-defined phrase p is encrypted (as $E_k(p)$) and stored on the PDA. On repeat access, the user is prompted again with the empty grid, upon which she draws the same design. A symmetric key k' is derived and an attempt is made to decrypt $E_k(p)$. If it results in p, then $k' = k$ and the password (and key) is accepted. The user then can proceed to encrypt/decrypt data records. k is deleted from the PDA at the latest when the PDA is powered off.

An adversary who captures the PDA can presumably obtain all of the ciphertext encrypted under k, and since p is either public or stored in plaintext on the device, the adversary has at least one known plaintext/ciphertext pair with which to attack E. For a strong encryption scheme E, however, the best bet for the attacker remains to guess the original password, which, as we will show in Section 3.3, on average is likely to be much harder than if the attacker were

faced with attacking a textual password.

We implemented the DAS scheme on the Palm Pilot and use it regularly to encrypt/decrypted information on our PDAs. The Pilot is based on the Palm operating system that is integrated with the `Graffiti` writing technology. The Palm OS supports a very natural form of data input, and as such, provides an ideal platform for implementing the DAS scheme.



(a) User inputs desired secret (b) Internal representation (c) Raw bit string

(d) interface to database (e) Re-entry of secret (f) Authorization failed

Figure 3: A password is created by drawing the secret on the display as shown in (a). Both the internal representation of the input password showing the cells covered by the user's drawing and the derived key are depicted in (b) and (c) respectively. To apply a symmetric cryptographic function to records in the database (shown in (d)), the user selects the records and then re-inputs the DAS password. If the encryption of a known clear-text with the input password matches the stored ciphertext created during initialization, then the symmetric cryptographic routine, $E_k(x)$, is applied to the selected records. Otherwise, the user is prompted to re-enter the DAS secret.

The interface for our DAS implementation is shown in Figure 3. Our application shares the database of the `memopad` application, and allows a user to encrypt/decrypt records in the database based on a user specified drawing. Our implementation conforms to the methodology outlined in Section 3.2, with SHA-1 as the cryptographic hash function and 3DES[1] as the symmetric encryption scheme.

3.3 Security of the DAS Password Scheme

We define the information content of a password space as the entropy of the probability distribution over that space given by the relative frequencies of the passwords that users actually choose. Information content is the correct measure for describing difficulty of attack, since it determines the optimal choices to be made when trying different possibilities for a password.

[1]Based on Ian Goldberg's crypto library for his port of SSLeay for the Pilot (see `http://www.isaac.cs.berkeley.edu/pilot`).

High information content renders a password scheme more or less invulnerable. For example, if users did in fact choose passwords uniformly from the space of all textual passwords, successful attacks would be extremely unlikely. What is it that renders such attacks successful in practice? There are two factors. The first is that in reality users do not choose their passwords uniformly. If we assume that the data collected in Klein's study [Kle90] is representative of the general population, then users in fact use only 10^{-8} of the possible passwords 25% of the time. Such a distribution is highly peaked, and the information content of the textual password space is correspondingly reduced.

However, the fact that users do not pick passwords uniformly is in itself not sufficient to make password guessing attacks successful. The second factor that renders textual passwords vulnerable is that the attacker has significant knowledge of the distribution of user passwords, and can use that knowledge to her advantage. In the case of textual passwords, this knowledge includes information about specific peaks in the distribution (users often choose passwords based on their own name), and information about gross properties (words in the English dictionary are likely to be chosen). Without information about the distribution, an attacker would be no better off than if users were in fact choosing uniformly.

Due to the dependence of the security of a scheme on the passwords that users choose in practice, a new password scheme can not be *proven* better than an old scheme. Performing trials on subjects in order to learn the distribution of user passwords for a new scheme is impractical for such large sample spaces. In the case of textual passwords, learning the knowledge that attackers routinely use would correspond to trying to learn the English dictionary (among others) given no prior knowledge of the types of letter combinations used in English, by having subjects type in 8-character passwords. In the absence of such objective proof, we present three plausibility arguments that suggest that the DAS scheme is considerably harder to crack than the conventional textual scheme. Two of these are estimates of the information content of the DAS password space, and hence address why textual passwords are vulnerable to attack in practice. The third argument discusses the effect that lack of knowledge of the distribution of user choices has on an attacker and the likelihood that such lack of information can be used in a deliberate and constructive manner to attack a password scheme.

3.3.1 Argument 1: The Size of the Password Space

First we consider the raw size of the password space, or in other words, its information content assuming users are equally likely to pick any element as their password. The raw size is an upper bound on the information content of the distribution that users choose in practice. We need some way to delimit the password space in order to obtain a finite answer, or in probabilistic terms, a way to ascribe probability zero to an infinite subset of passwords, leaving a finite subset which we will count. We will assume that all passwords of total length (as defined in Section 3.1) greater than some fixed value have probability zero. We compute the size $\Pi(L_{\max}, G)$ of the space of passwords of total length less than or equal to $L_{\max}$ on a grid of size $G \times G$. Π is defined in terms of the number of passwords with total length equal to L,

$P(L, G)$ by:

$$\Pi(L_{\max}, G) = \sum_{L=1}^{L\max} P(L, G) \tag{1}$$

In turn, $P(L, G)$ can be defined in terms of $N(l, G)$, the number of strokes of length equal to l by:

$$P(L, G) = \sum_{l=1}^{l=L} P(L - l, G)N(l, G) \tag{2}$$

In words, the above equation says that a new stroke of length l may be added to any shorter password of length $L - l$ to make a password of total length L. By defining $P(0, G) = 1$, we complete the definition of the recurrence, once we have given an expression for $N(l, G)$.

The following recurrence relation defines $N(l, G)$. Let $n(x, y, l, G)$ be the number of strokes of length l ending at the cell (x, y) in a grid of size $G \times G$. Then N can be defined in terms of n by

$$N(l, G) = \sum_{(x,y)\in[1..G]\times[1..G]} n(x, y, l, G) \tag{3}$$

Clearly, $\forall(x, y) \in [1..G] \times [1..G], n(x, y, 1, G) = 1$. Moreover, it is convenient to define n at the "boundaries" of the grid as follows:

$$n(0, y, l, G) \;=\; n(x, 0, l, G) \;=\; n(G + 1, y, l, G) \;=\; n(x, G + 1, l, G) \;=\; 0$$

The function n can then be evaluated using the following recurrence:

$$\begin{aligned} n(x, y, l, G) \;=\; & n(x - 1, y, l - 1, G) + n(x + 1, y, l - 1, G) \\ & + n(x, y - 1, l - 1, G) + n(x, y + 1, l - 1, G) \end{aligned}$$

Putting the pieces together, we can calculate the size of the password space. The results for different upper bounds on total password length on a 5×5 grid are given in Table 1.

$L_{\max}$	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
$\log_2$(# passwords)	5	10	14	19	24	29	33	38	43	48
$L_{\max}$	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*
$\log_2$(# passwords)	53	58	63	67	72	77	82	87	91	96

Table 1: Number of passwords of total length less than or equal to $L_{\max}$ on a 5×5 grid.

The data in Table 1 shows that the raw size of the graphical password space surpasses that of textual passwords for reasonable password configurations. While these numbers are encouraging, in practice not all graphical passwords are equally likely to be chosen by users, rendering a uniform distribution overly optimistic. For example, although the number of passwords of length greater than or equal to 12 is already greater than the number of textual passwords of 8 characters or less constructed from the printable ASCII codes ($95^8 \approx 2^{53}$), this includes all possible combinations of twelve isolated dots.

In order to obtain a more realistic estimate of the information content, in the following section we suggest a model in which we characterize passwords as being "memorable" in terms of the programs which generate them.

3.3.2 Argument 2: Modeling User Choice

We assume that the reason that users choose from such a small subset of textual passwords is that the passwords in that set are more memorable than those outside it. That lack of imagination on the part of the user is not the cause for the lack of variety is supported by the fact that system-generated passwords have been so unsuccessful [Bis91]. By making the same assumption about DAS passwords, we can "reduce" our task to that of modeling the set of "memorable" graphical passwords. If we can show that this set, or some subset of it, has cardinality larger than the dictionary of textual passwords from which users typically choose, we can plausibly claim that as far as information content goes, DAS is more secure than conventional textual password schemes. Here, we identify two such subsets using different criteria of memorability, and show that the cardinalities of these sets do indeed satisfy the above criterion.

What constitutes a memorable password? In the textual case, one obvious component is semantic content. If the sequence of characters has a meaning for the user, the password is more likely to be memorable [Mil56, She67, BSH77]. This semantic definition is extremely hard, if not impossible, to characterize in the abstract. It is only because the semantic content of many character combinations has been established by the common use of a written language that we can talk about such content at all. In the DAS scheme, there are obvious password components that have meaning, but it is impossible *a priori* to identify exactly which passwords will have semantic content, and to how many users, precisely because it is not a representation with meanings established by common use.

Memorability based on simple shapes The first set of "memorable" passwords that we define is a subset of those passwords that might reasonably be expected to carry meaning. We look at all strokes in the form of rectangles, and show that by combining two such strokes, we already reach the size of the dictionaries used to crack textual schemes. To be more precise, consider the set of rectangles within a $G \times G$ grid. Since a rectangle can be defined by two rows (the top and bottom edges of the rectangle) and two columns (the left and right edges), it is clear that the number $R(G)$ of rectangles on a $G \times G$ grid is

$$R(G) = \binom{G}{2}^2 = \frac{1}{4}G^2(G-1)^2 \tag{4}$$

Each of these rectangles can be generated in many ways. For example, the starting point of a stroke can be at any of the corners, and the stroke direction can be clockwise or counter-clockwise. This yields 8 possibilities for each rectangle. In addition, one can choose whether to close the rectangle by returning to the starting cell or not, again doubling the possibilities.

On a 5×5 grid, this amounts to 1600 possible strokes. Two such strokes in succession gives 2.56×10^6 passwords, already roughly the size of the textual dictionary that contained the passwords of 25% of users in Klein's study [Kle90]. Clearly we can generate a much larger set of passwords by considering variations on the theme of rectangles, or by considering other Gestalt forms [Wer38].

Memorability based on short algorithms The second set of passwords that we describe is suggested by the discussion of text-based graphical passwords in Section 2, which pointed toward a different definition of memorability. There, a memorable sequence of positions seemed characterized by the fact that there existed a short algorithm to describe the sequence. It is this definition of memorable that we wish to apply here, since it can be characterized in precise terms. We do not argue that every memorable password has a short algorithm to describe it, but that passwords describable by short algorithms are memorable. We will show that the cardinality of this subset of memorable passwords is already larger than the dictionary of character sequences from which users most often draw their passwords, and that therefore, following the argument above, the DAS password scheme should be harder to crack in practice than the conventional textual scheme.

In order to characterize the 'complexity' of the algorithm required to generate a DAS password, we define a very simple language suited to the task of describing DAS passwords. Then, we generate all programs in this language whose complexity is at most a chosen maximum. In order to avoid counting different programs that produce the same password twice, we then execute the generated programs to output the passwords, which are then bucketed, and distinct passwords counted. The result is the number of DAS passwords generated by programs of complexity at most the chosen maximum.

Before describing the results of this endeavor, we give some details of the language in which we generated the programs. The grammar of the language is as follows:[2]

program	$\rightarrow$	digit digit block
block	$\rightarrow$	statement block
statement	$\rightarrow$	instr | **repeat** digit block **end**
instr	$\rightarrow$	**up** | **down** | **right** | **left** | **penup** | **pendown**
digit	$\rightarrow$	**1** | **2** | **3** | **4** | **5**

The first two digits represent a starting position. The instructions **up**, **down**, **left**, and **right** move the pen one square in the indicated direction. If the pen is currently in the down position, then moving in the specific direction will draw a line. Otherwise, the direction statement will merely move the pen location. The pen begins in the up position. The **repeat** statement is our iterator. We allow digit values up to the number of grid squares on each axis (i.e., 5 on a 5×5 grid) to indicate the number of repetitions, although in principle a password

[2]Those readers old enough to remember the APPLE II will recognize that our language bears a striking resemblance to Turtle Graphics [SP76], the kid's language based on LOGO (see, e.g., [ABGP75]).

consisting of more than 5 repetitions of something on a 5×5 grid are possible (e.g., ten dots in the same position).

To calculate the complexity for a given program, we assign a complexity to each literal in our language. We assign every statement and digit complexity one, except for the **end** marker, which has complexity zero. This means that **repeat** loops have a complexity of two (one for the **repeat** statement, and one for the integer indicating the number of repetitions) plus the complexity of the repeated block. In addition, the last **penUp** statement of a program is assigned a complexity of zero (lifting one's pen from the surface at the end of entering a password is difficult to forget). So, for example, there are no programs of complexity only two, since the integers describing the starting position of the program already consume a complexity of two without allowing any **penDown** statements. The first complexity of which there are any programs is three—the two digits describing the initial starting position, followed by a **penDown**—and the passwords generated by programs of complexity three are simply those consisting of a single tap on one of the grid squares. Note that our complexity calculations for programs are very conservative, in the sense that even pen movements *between* strokes (i.e., while the pen is raised) are counted in the complexity of a program.

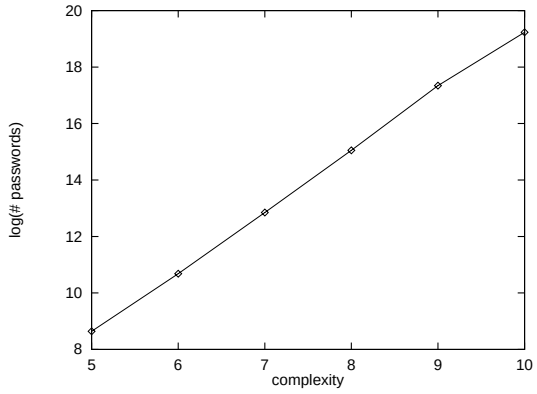
The results of using the above described procedure for counting the number of DAS passwords of a given complexity on a 5×5 grid are shown in Figure 4. As expected, this data shows that the number of DAS passwords grows exponentially as a function of the maximum complexity of the program. What is more interesting, however, is that by extrapolation[3] we see that the number of DAS passwords generated by programs of only complexity 12 far surpasses the dictionary size of approximately 3×10^6 used in Klein's password-cracking studies [Kle90]. As a point of comparison, even just tracing the outermost cells of a 5×5 grid to make a square already requires a program of complexity at least fifteen in our simple language. And, obviously this design and many other, more complex ones will fall in the realm of memorable for most users. We believe that this is compelling evidence that DAS passwords, of which those generated by programs of complexity at most twelve are but a very small subset, will be significantly harder to crack in practice than textual passwords. Example DAS passwords and the shortest programs that generate them are given in Appendix B.

3.3.3 Argument 3: Lack of Knowledge of the Distribution

Given the size of typical password spaces, knowledge of the distribution of user passwords is essential to an adversary. Without such knowledge the adversary has no way of directing her search toward more probable passwords, and is no better off than if users really did pick their passwords uniformly from the set of possibilities [Cov91].

Where did the knowledge of the distribution come from in the case of textual passwords? For the most part, dictionaries have been compiled by using reasonable assumptions about likely choices. The assumptions stem from the use of a shared language, and shared knowledge

[3]Calculating the exact number of 'memorable' graphical passwords, as defined by our language, for complexities greater than 10 requires significantly more computational resources (and time) than we have available to us. An attacker wishing to build any such database will face similar difficulties.



Complexity	Passwords
3	25
4	105
5	398
6	1,645
7	7,370
8	34,026
9	165,760
10	614,660

Figure 4: Number of DAS passwords generated by programs of short complexity on a 5×5 grid.

of the semantic content of words. For example, in the work of Klein [Kle90] the sources for likely passwords included the St. James Bible, the `Unix` dictionary, and many other sources of English words that were available to the author precisely because they are a part of our language. If these assumptions had turned out to be incorrect, textual password schemes would be extremely difficult to break in practice.

The assumptions made about likely password choices are strongly confirmed by Klein's work, and by successful attacks on textual passwords, but confirmation of pre-existing dictionaries is not the same as deriving a dictionary in the first place by learning from example without prior knowledge. In the case of textual passwords, this would mean learning the English dictionary (or some equivalent corpus of words) by collecting user passwords. This would involve acquiring millions of verified passwords, and, as such, represents a significant challenge for a would-be adversary.

In the case of the DAS scheme, similar reasonable assumptions about user choice do not exist. Furthermore, the learning task is made even more difficult by two factors. First, arguments 1 and 2 suggest that both the space of passwords and the space of likely user choices are considerably larger than for textual passwords. Second, the platform that we are targeting, PDAs, renders the task of data collection much harder than on, e.g., networked computers.

3.4 Summary

The above arguments do not *prove* that graphical password schemes are more secure than traditional textual schemes. In fact, as we have argued, such a proof is impossible. Nevertheless, taken together they provide convincing evidence that this would indeed be the case.

4 Prior Work

There is a considerable amount of prior work on authenticating users via graphical inputs to a device, particularly handwritten signatures (see, e.g., [LP90, LP94, Nal97]). None of these

works strive for exact repeatability by the user, and therefore, a model of the user's graphical input is stored on the device and used to ascertain whether a new input is sufficiently similar to the previously-stored one to grant access. This renders it essential to protect the device's (PDA's) storage from probes by an attacker. In contrast, repeatability is achieved in our schemes, thereby enabling designs in which the device, if captured, is of little help to the attacker (see Section 1).

The security of textual passwords has been examined by numerous researchers, notably [MT79, Kle90, FK90, Spa92, Wu99]. Without exception, these studies reiterate the fact that people choose passwords that are easy to find by automated search. In order to improve the security of passwords, it is common practice for system administrators to invoke reactive password checkers to identify weak passwords [RU88, Muf92], or to use proactive checkers to filter out certain classes of weak passwords when the user inputs her password for the first time [Bis95, Spa91].

A technique to improve the security of even a poorly chosen password is to *salt* the password by prepending it with a random number, R, before hashing [MT79]. R is typically stored with the hash value so that the hash input can be reconstructed from the plaintext password. When a user tries to authenticate and enters a password, R is retrieved from the password table, prepended to the password and hashed. The result is compared to the stored hash. The net effect is that the search space of the attacker is increased by a factor of $2^{|R|}$ if the attacker does not have access to the stored salts. A variation on this is to not store R [Man96]. During a login sequence the password is input and the system searches for R by trying all possible values for the salt. This variation increases the search space for the attacker by a factor of $2^{|R|}$ even if the attacker captures all stored information related to password authentication. However, this results in significant additional overhead on each login if R is large.

The techniques in this paper can be combined in natural ways with the techniques discussed above for strengthening textual passwords—i.e., proactive and reactive password checking, and salting—to improve the security of graphical passwords, as well.

More distantly related is work on one-time passwords (e.g., [Hal93]). One-time password schemes are relevant primarly for network settings, to defend against the threat of a network eavesdropper capturing password information in transit between the user and a secure authentication server. To render such eavesdropping harmless, a one-time password scheme varies the user's password from each login to the next in a way that only the user and the server can predict, based on state shared between the server and user. In the main setting we consider, however, there is no network communication that is vulnerable to eavesdropping, and consequently the attacks with which we are concerned is the capture and analysis of all stored state relevant to authentication (the PDA in our setting, or equivalently the server's and client's states in the one-time password setting). One-time password schemes of which we are aware offer no benefit against this attacker over traditional password schemes.

5 Conclusions and Future Work

We have presented graphical password schemes that achieve better security than conventional textual based passwords alternatives. Our approaches exploit the input capabilities of graphical devices that allow us to decouple the position of inputs from the temporal order in which they occur. We presented arguments for the security of our schemes in which we analyzed the information content of the resulting password spaces. We also presented a novel approach for capturing the 'memorability' of graphical passwords by examining the class of DAS passwords generated by short programs in a simple grid-based language, and showed that even this relatively small subset of graphical passwords (for some fixed program complexity) constitutes a much larger password space than the dictionaries of textual passwords to which a high percentage of passwords typically belong.

We have been using our DAS-based memo pad encryption on the Palm Pilot for a few months and we are quite happy with its ease-of-use. We hope to initiate some user studies to collect further feedback on (1) user acceptance and (2) their choices of passwords.

For future work we are exploring alternative schemes for modeling the memorability of DAS passwords that we hope will capture their high-level structure more intuitively than our current models. The goal is to capture the concept of organized drawings, in which the view of the whole is more than just the sum of the individual parts that constitute it. For example, one can view a square as an object in itself and not simply as an arrangement of the individual lines from which it is composed. In this way, we can define a set of primitive structures from which all 'memorable' drawings can be derived using meta-level compositions of these primitives. We hope to show that even this reduced set of DAS passwords (for some reasonable number of primitives) constitutes a larger space than that of textual-based passwords, and as such will be significantly harder to crack in practice.

References

[ABGP75] H. Abelson, J. Bamberger, I. Goldstein, and S. Papert. Logo Progress Report 1973-1975. *MIT, AI memo 356*, September 1975.

[Alv90] A. Alvare. How crackers crack passwords or what passwords to avoid. In *Proceedings of the 2nd USENIX Security Workshop*, August 1990.

[Bis91] M. Bishop. Password management. In *Proceedings of COMPCON '91*, pages 167–169, February 1991.

[Bis95] M. Bishop. Improving system security via proactive password checking. *Computers and Security* 14(3):233-249, April 1995.

[Blo96] G. Blonder. *Graphical passwords.* United States Patent 5559961, 1996.

[BKD75] G. H. Bower, M. B. Karlin, and A. Dueck. Comprehension and memory for pictures. *Memory and Cognition* 2:216–220, 1975.

[BSH77] M. A. Borges, M. A. Stepnowsky, and L. H. Holt. Recall and recognition of words and pictures by adults and children. *Bulletin of the Psychonomic Society* 9:113–114, 1977.

[Cal98] M. W. Calkins. Short studies in memory and association from the Wellesley College Laboratory. *Psychological Review* 5:451–462, 1898.

[Cov91] T. M. Cover,and J. A. Thomas. *Elements of Information Theory*, John Wiley and Sons, 1991.

[FK90] D. Feldmeier and P. Karn. UNIX password security – Ten years later. In *Advances in Cryptology—CRYPTO '89 Proceedings* (Lecture Notes in Computer Science 435), 1990.

[GS96] S. Garfinkel and E. Spafford. *Practical Unix & Internet Security*. O'Reilly & Associates, Inc., 1996.

[Hal93] N. Haller. The s/key(tm) one-time password system. In *Proceedings of the 1994 Symposium on Network and Distributed System Security*, pages 151–157, February 1994.

[Kle90] D. Klein. Foiling the cracker: A survey of, and improvements to, password security. In *Proceedings of the 2nd USENIX Security Workshop*, August 1990.

[LP94] F. Leclerc and R. Plamondon. Automatic signature verification: The state of the art—1989–1993. *International Journal on Pattern Recognition and Artificial Intelligence* 8(3):643–660, June 1994.

[LP90] G. Lorette and R. Plamondon. Dynamic approaches to handwritten signature verification. In *Computer Processing of Handwriting*, pages 21–47, World Scientific, 1990.

[Mad83] S. Madigan. Picture memory. In *Imagery, Memory, and Cognition*, pages 65–86, Lawrence Erlbaum Associates, 1983.

[Man96] U. Manber. A simple scheme to make passwords based on one-way functions much harder to crack. *Computers & Security*, 15(2):171–176, 1996.

[Man91] G. Mandler. Your face looks familiar but I can't remember your name: A review of dual process theory. *Relating Theory and Data* 207–225, 1991.

[Mil56] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review* 63:81–97, 1956.

[MT79] R. Morris and K. Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, November 1979.

[Muf92] A. Muffet. Crack: A sensible password checker for Unix. Available via anonymous *ftp* from *cert.org*.

[Nal97] V. S. Nalwa. Automatic on-line signature verification. *Proceedings of the IEEE*, pages 215–239, February 1997.

[NRW76] D. L. Nelson, U. S. Reed, and J. R. Walling. Picture superiority effect. *Journal of Experimental Psychology: Human Learning and Memory* 3:485–497, 1977.

[Pai71] A. Paivio. *Imagery and Verbal Processes*. Holt, Rinehard, and Winston, New York, 1971.

[Pai76] A. Paivio. Imagery in recall and recognition. *Recall and Recognition*, John Wiley, New York, 1976.

[PRS68] A. Paivio, T. B. Rogers, and P. C. Smythe. Why are pictures easier to recall than words? *Psychonomic Science* 11:137–138, 1968.

[RU88] T. Raleigh and R. Underwood. CRACK: A distributed password advisor. In *Proceedings of the 1st USENIX Security Workshop*, pages 12–13, August, 1988.

[She67] R. N. Shepard. Recognition memory for words, sentences, and pictures. *Journal of Verbal Learnings and Verbal Behavior* 6: 156–163, 1967.

[Spa91] E. Spafford. Preventing weak password choices. In *Proceedings of the 14th National Computer Security Conference*, pages 446–455, October 1991.

[Spa92] E. Spafford. Observations on reusable password choices. In *Proceedings of the 3rd USENIX Security Symposium*, September 1992.

[Sta73] L. Standing. Learning 10,000 pictures. *Quarterly Journal of Experimental Psychology* 25:207–222, 1973.

[SP76] Cynthia J. Solomon and Seymour Papert. A case study of a young child doing Turtle Graphics in LOGO. *MIT AI memo 375*, July 1976.

[Wel72] J. E. Wells. Encoding and memory for verbal and pictorial stimuli. *Journal of Experimental Psychology* 24:242–252, 1972.

[Wer38] Max Wertheimer. *Laws of organization in perceptual forms.* In W. Ellis, W (Ed. & Trans.), A source book of Gestalt psychology (pp. 71-88). London: Routledge & Kegan Paul. 1938. (Original work published in 1923 as Untersuchungen zur Lehre von der Gestalt II, in Psycologische Forschung, 4, 301-350.)

[Wu99] T. Wu. A real-world analysis of Kerberos password security. In *Proceedings of the ISOC Symposium on Network and Distributed System Security*, 1999.

A A Picture is Worth a Thousand Words

Our "draw-a-secret" scheme is motivated by the experimentally-proven fact that pictures are easier to remember than words. Why are pictures easier to recall? Four hypotheses have been offered as explanations of picture-word differences in recall:

- Common-code theory: this view of memory and recall theorizes that pictures and words access semantic information in a single conceptual system that is neither word-like or picture-like. This theory hypothesizes that pictures and words both require analogous processing before accessing semantic information, but pictures require less time than words for accessing the common conceptual system. Common-code theorists attribute better picture recall to differences in the encoding of pictures and words: pictures share fewer common perceptual features among themselves and therefore need to be discriminated from a smaller set of possible alternatives than words. The greater number of dictionary meanings or the greater lexical complexity of words create uncertainty and confusion, and hence poorer recall.

- Dual-code theory: unlike the common-code approach, this theory postulates that language and knowledge of worlds are represented in functionally distinct verbal and nonverbal memory systems. The verbal system is specialized for dealing with linguistic information whereas the non-verbal stores perceptual information. The most evident examples of dual process theory can be found in experiences that we have all had at some time or the other: we meet someone, know them to be familiar but do not know who they are; we recognize a melody, but fail to remember its name or when or where we heard it before; we read a line of a poem, know it, but do not know where we have read it before, much less the title or author of the poem. In all these cases, we experience a sense of familiarity, but have — at least at first — no access to any contextual or conceptual information [Man91].

  Dual code theory suggests that there are qualitative differences between the ways words and pictures are processed during memory and hypothesizes that the reason for superior picture memory is that pictures automatically engage multiple representations and associations with other knowledge about the world, thus encouraging a more elaborate encoding than occurs with words [PRS68, Pai71].

- Abstract-propositional theory: in contrast to the dual-code approach, this theory rejects any notion of sophisticated distinctions between verbal and nonverbal modes of representation, but instead describes representations of experience or knowledge in terms of

an abstract set of relations and states, called propositions. This theory postulates that better free recall with pictures may be due to even more elaborative encoding effects than those suggested by dual-code theorists. Propositional theorists view the involvement of *abstractive and interpretive* processes in picture memory as the explanation for the picture effect [Mad83]. Therefore, a series of line drawings will be poorly remembered if a subject is unable to interpret the drawings in a meaningful way, whereas memory for the same drawings, presented in the same way will be much better if a conceptual interpretation is provided, and it this interpretive process which is responsible for better picture memory recall.

While the strongest evidence thus far for the picture effect can be best explained by dual-code theory (see [Man91]), an understanding of picture memory and the means by which we acquire and maintain information about the visual environment is still an ongoing challenge. Nonetheless, the research to date provides strong arguments in terms of the memorability of drawings over words in recognition tasks and hence its applicability to computer security.

B Examples

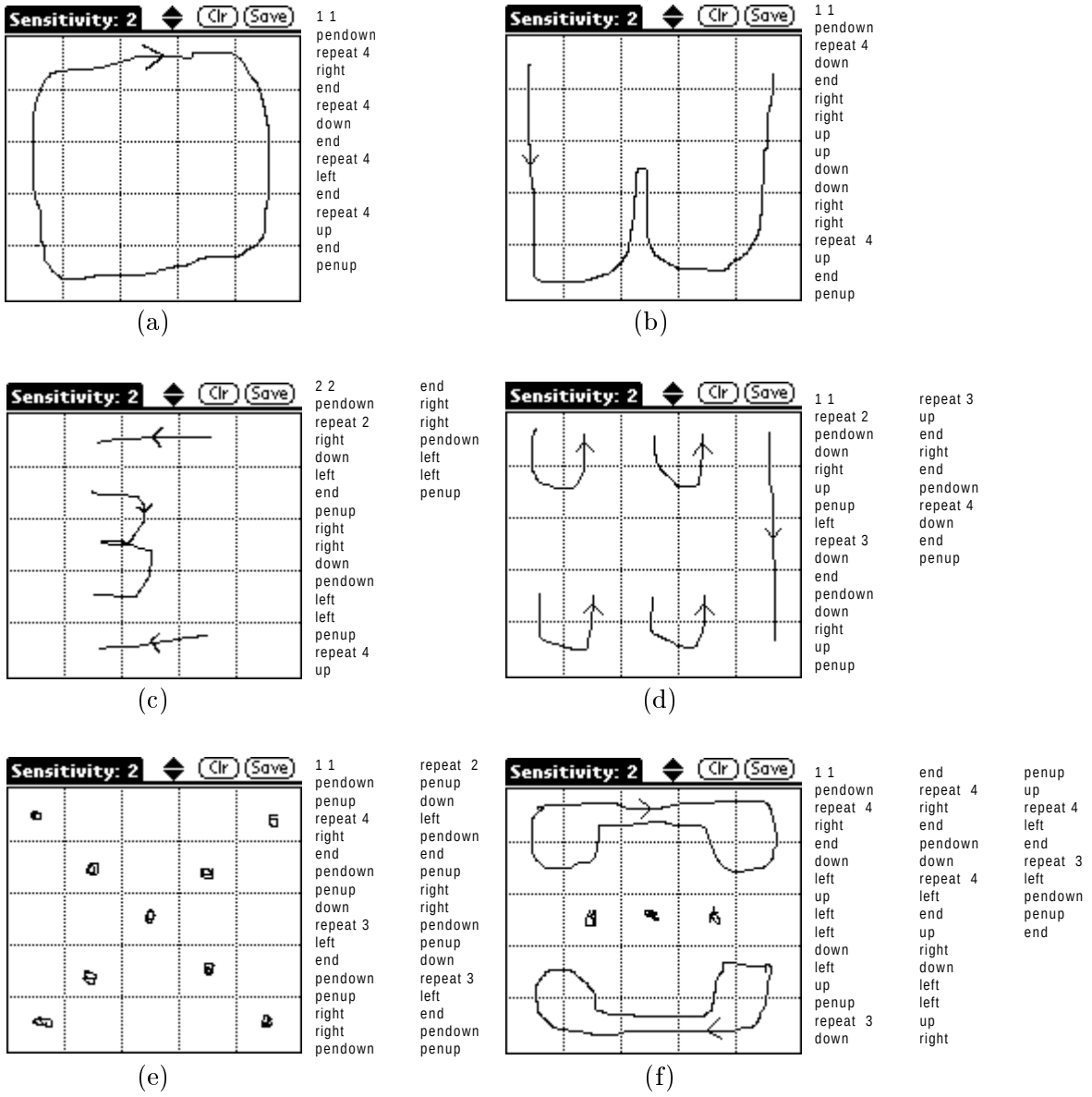


Figure 5: The drawings above have complexities 15, 17, 24, 26, 39, and 42, respectively (recall that final pen-ups have zero cost).