ON THE PREDICTABILITY AND SECURITY OF USER CHOICE IN PASSWORDS

by

Julie Thorpe

A thesis submitted
in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

School of Computer Science

at

CARLETON UNIVERSITY

Ottawa, Ontario
January, 2008

Table of Contents

List of Tables

List of Figures

Abstract

It is well-known that traditional text-based passwords are weak due to predictable patterns in user choice. In response, other knowledge-based authentication schemes such as *graphical passwords* have been proposed, motivated in part by our remarkably better memory for pictures over words. We hypothesize that these schemes, when allowing users free choice of their password, will suffer from similar weaknesses.

We examine and generalize existing methods for attacking knowledge-based schemes, and apply these methods to two representative, previously unanalyzed, graphical password schemes: the "Draw-A-Secret" scheme of Jermyn et al. (1999), and the "PassPoints" scheme of Weidenbeck et al. (2005). We validate our hypothesis using data collected from our own user studies, and from user studies by others, providing the first attacks and effective security analyses for these two schemes. Our results naturally lead to a set of recommendations that may help improve the effective security of these particular schemes.

In light of these attacks, we propose a novel idea for user authentication we call "pass-thoughts", which may prove to have some unique properties to defend against various attacks (but we do not prove this herein). We end with a discussion of other considerations for user authentication schemes, a comparative analysis of existing schemes, and a discussion to put the results of this thesis in context.

Acknowledgements

First and foremost, I would like to express my gratitude to my thesis supervisor, Prof. Paul Van Oorschot. He introduced me to research in user authentication and encouraged me to pursue this degree. I cannot possibly quantify the importance of his academic guidance and valuable advice these past five years.

I would also like to thank my committee for helping improve this thesis: Prof. Anil Somayaji for his encouragement regarding pass-thoughts and for helping bring out “the big picture”, Prof. Robert Biddle for his insightful questions regarding predictive models, Prof. Carlisle Adams for his incredible eye for detail, and Prof. Fabian Monrose for agreeing to serve as the external examiner and for his valuable comments.

I am grateful to my undergraduate advisor, Prof. Michael McAllister, for encouraging me to apply for a Natural Science and Engineering Research Council (NSERC) award; without his encouragement, I may have never pursued graduate studies.

Thanks to all members of Carleton’s Computer Security Lab, for your feedback and discussions that helped improve my research and for the many laughs in the lab. Many thanks also to Prof. Robert Biddle and Sonia Chiasson for their collaborative effort with us in running the PassPoints user studies; to Prof. Adrian Chan and his students for welcoming me into the Biomedical Sensors and Signals Laboratory and providing advice for my pass-thoughts experiments; and to Prof. Doron Nussbaum, for providing access to his lab resources that helped me in developing PassPoints image processing software.

Finally, I thank all of my wonderful friends and family for their support and celebrations for various milestones. I’m especially thankful to my parents for their support and words of wisdom, and to Laura, who I could always count on for a dose of humour.

I am thankful to NSERC for funding a PGS A and CGS D, and to have received the following financial awards administered by Carleton University: the Edward Bower Carty Graduate Scholarship, the J. James Mackie Endowment, and the Wylda Blanche McDermid Holbein Memorial Scholarship.

Chapter 1

Introduction

1.1 Motivation

Traditional text-based passwords have a well known weakness; people tend to choose passwords with predictable patterns that, unsurprisingly, correlate with what is easiest to remember. These patterns in user choice produce a marked reduction in entropy, and thus the effective security that passwords provide.

In response to this bleak status quo of text-based passwords, many new knowledge-based (or “what you know”) user authentication techniques have emerged that *in theory* produce higher entropy user authentication. It is relatively straightforward to determine the theoretical entropy where user choice is assumed to be uniform; however, measuring the entropy when considering the predictability of user choice is a more challenging task.

Accurately estimating the entropy of new user authentication schemes is an important step in measuring the security they provide. If a new scheme has low entropy due to patterns in user choice, it may fall to attack. To avoid the compromise of user accounts, it is important that any such security problems are understood prior to widespread deployment.

1.2 Threat Model

When an authentication scheme has low entropy, it is susceptible to guessing attacks. We focus on the threat of offline dictionary-based guessing attacks,[1] where the attacker is only limited by his or her time and computational resources. We assume that the attacker has access to one or more hashed passwords or verifiable texts [56], which may have been obtained by one of many ways discussed further below.

[1]However, one of the attacks we present for PassPoints (see Section 6.6) would be a threat in both offline and online environments.

Of course, knowledge-based user authentication schemes are also vulnerable to other attacks, such as shoulder surfing, phishing, and social engineering attacks. Although these other threats are real and themselves worthy of study, we focus on the threat of dictionary-based guessing attacks, since many new schemes are argued to be resistant. The threat of dictionary-based guessing attacks also has been shown to remain for text passwords, despite proper implementation of the password policies designed to thwart their success [167].

Guessing attacks are an increasing threat in today's mobile environment. Mobile devices can be lost or stolen, and once a device is in an attacker's hands, he/she might manually make guesses or override the regular input method for an online attack, or copy the hashed password or encrypted check-word (and possibly other relevant information) for an offline attack. If a device's files are password-encrypted, a guessed password equates with a compromise of private data. Guessing attacks are also a threat to servers and desktop PCs, which normally store hashed passwords in a file using a one-way hash (e.g., MD5 or SHA-1). One-way hashes are not reversible, and thus offline recovery of the original password(s) requires hashing candidate guesses for comparison against the file's values. Insiders, or anyone with physical access to computers hosting password files or backup tapes, can easily steal copies of the password files required to launch an offline attack. Furthermore, operating system vulnerabilities and related access control failures can lead to the inadvertent disclosure of entire password databases.

1.3 Thesis Statement

We hypothesize other (non-text-based) types of knowledge-based user authentication will also suffer from lowered entropy when based on user-chosen secrets. Lowered entropy has already been demonstrated for a set of knowledge-based schemes: regular text passwords [77, 167, 82], passphrases [82], and recognition-based graphical passwords [34]. We believe lowered entropy can also be shown for other previously unanalyzed schemes by generalizing established attack methods, and applying this generalization using information related to the particular scheme in question. This hypothesis motivates the following questions:

- *Question 1:* Can established attack strategies for some knowledge-based user authentication methods (e.g., for text passwords, pass-phrases, and recognition-based graphical passwords) be generalized?
- *Question 2:* How well do these strategies work against specific, previously unanalyzed, representative schemes?
- *Question 3:* Can our knowledge lead us to other directions in user authentication that might be more immune to such attacks?

1.4 Thesis Overview and Organization

This thesis aims to answer Question 1 in Chapter 4 by generalizing methods of identifying and generating "weak password subspaces". These weak password subspaces roughly equate with guessing dictionaries. We call this generalization "predictive modelling" [113], which gathers information related to the user task from other sources (e.g., psychological studies, visual attention, and observing small sets of other users in another context).

To answer Question 2, we create "candidate weak password subspaces" for two representative, previously unanalyzed graphical password schemes: a pure-recall scheme called "Draw-A-Secret" (DAS) by Jermyn et al. [72] in Chapter 5, and a cued recall scheme called "PassPoints" by Weidenbeck et al. [164, 163, 162] in Chapter 6. Data from both small-scale and large-scale user studies (both our own user studies and one by Tao [146]) are used to determine whether these subspaces are indeed weak. We showed that these subspaces are weak, thus adding two more schemes to the body of literature supporting our hypothesis that other (non-text-based) knowledge-based user authentication schemes are weak when based on user-chosen secrets. This result, along with results by others [77, 167, 34, 82], suggests the emergence of a general pattern that calls the effective security of any candidate knowledge-based authentication scheme into question, when based solely on user-chosen secrets in environments where offline attack is possible, and for some schemes, even where online attack is possible (see Section 6.6).

In response to Question 3, in Chapter 7 we discuss some other user authentication schemes that are more immune to guessing attacks. Most of these other schemes unfortunately have limitations, leading us to propose a new type of non-static biometric that we call “pass-thoughts” [153], which may prove to be more immune to the attack methods known to date (but this is not proven herein).

Finally, to put the results of this thesis into context, we survey existing user authentication schemes and their resistance to a set of attacks in Chapter 8, review a set of other considerations for user authentication schemes, and discuss different scenarios whereby some of these schemes may still be “secure enough”. Concluding remarks are made in Section 8.5.2.

1.5 Main Contributions

Our main contributions include a generalized method for analyzing the security of knowledge-based user authentication methods, and an illustration of how to apply this method using two different graphical password schemes. We provide the first attacks and effective security analyses of two representative graphical password schemes: a pure-recall scheme called “Draw-A-Secret” (DAS) by Jermyn et al. [72], and a cued-recall scheme called “PassPoints” by Weidenbeck et al. [164, 162, 163]. These effective security analyses lead us to a set of specific recommendations to help improve the security of these two schemes, and to outline existing and new ideas that might prove to reduce the problem of predictable patterns in user choice. In our view, a lesson that we can take away from these results is that graphical passwords are still immature, and are just beginning to undergo serious analysis. We survey existing graphical password schemes and their resistance to a set of attacks, within the context of the larger body of user authentication methods. Finally, we outline a new proposal for user authentication that may prove to have some interesting characteristics (but these are not proven herein) that we call “pass-thoughts”.[2]

[2]Most of the work presented in this thesis has been published [113, 149, 150, 151, 153].

Chapter 2

Background

2.1 Introduction

User authentication is one of the first lines of defense for a computer system, the method through which users prove that they are who they claim to be. As humans, confirming the identity of a person we have met before is easy; we simultaneously recognize many parts of a person, such as their physical appearance, voice, gait, style of speech, clothing, and shared knowledge. However, it is not as easy for a computer system to confirm such subtle factors. The most common form of user authentication is shared knowledge in the form of traditional text-based passwords, whereby a user enters a text string also known to the computer system. This form of knowledge-based user authentication is popular due to its low cost, ease to implement, and general familiarity amongst users. Unfortunately, text passwords have many well-known limitations: they often have low entropy in practice (making them susceptible to dictionary attacks; see Section 2.2.2), can be difficult to remember, and are vulnerable to shoulder-surfing [136], social engineering (through verbal or written communication) [95], phishing [39], and broader guessing attacks.

We focus on the threat of dictionary-based guessing attacks, further discussed in Section 2.2. These attacks have been actively fought with password policies, proactive checking, salting, stronger storage methods, and Automated Turing Tests (a.k.a. CAPTCHAS), further discussed in Section 2.3. Despite these defensive measures, guessing attacks remain a serious threat. Thus, many new schemes such as graphical passwords [98] have been proposed as replacements for text passwords. We discuss two graphical password schemes that are central to this thesis in Section 2.4.

2.2 Guessing Attacks

Password guessing attacks can be launched by human attackers and malicious software (e.g., the Morris Worm [143]). Such attacks can easily be launched by anyone due to the widespread availability of free software[1] to automate the process (e.g., John the Ripper [115], Crack [103], and RainbowCrack [141]).

For security reasons, systems normally store their passwords as the output of a one-way hash function (e.g., MD5 or SHA-1), and perform the same hash on a user-supplied password for a direct comparison; if the hashes match, the password is declared correct. Sometimes a *salt* is prepended to the password prior to hashing (see Section 2.3.2). Storing hashed passwords prevents someone with access from simply reading the password file. Since the one-way hash cannot be reversed, the only known way to obtain the original password is to compare hashed password guesses.

The efficacy of a guessing attack is tightly related to the entropy offered by a scheme. If users choose passwords that are either short enough to be guessed by brute-force (see Section 2.2.1) or are predictable enough to be in an attacker's dictionary (see Section 2.2.2), they may be guessed by an attacker under certain, often achievable, conditions.

These attacks are possible when an attacker is free to make as many guesses as he wishes against a target password. When login servers do not lock the account after a small number of tries (normally between three and ten), these attacks can be performed *online* by making repeated login attempts. Alternatively, these attacks could be performed *offline*, on an attacker's machine given the hashed target password. Hashed target passwords can be obtained through analysis of captured network traffic (e.g., for Kerberos [167]), although this can be prevented by using schemes such as Encrypted Key Exchange (EKE) [8] and Strong Password Exponentiated Key Exchange (SPEKE) [68]. However, operating system vulnerabilities [22] and related access control failures can lead to the inadvertent disclosure of entire password databases. Furthermore, password databases can be stolen by insiders with physical access to the device or login server by using a live CD[2] (e.g., Knoppix

[1]Although in some cases, optimizations such as larger dictionaries are available for a small fee (e.g., \$28 USD for John the Ripper's wordlists [115]).

[2]A live CD contains an operating system that will load directly from the CD, allowing someone

[78]), or anyone with physical access to backup media (e.g., cleaning staff). In such cases, the theft of hashed target passwords, and their subsequent compromise by a correct guess through an offline guessing attack is difficult to detect, as the attacker is providing perfectly valid proof of being a legitimate user.

The speed of offline guessing attacks is only limited by the attacker's available computing speed and disk space. Although these attacks have been known for years, they are still quite successful. We detail how these attacks work, and various published works reporting their success, in Sections 2.2.1 and 2.2.2.

2.2.1 Brute-Force Attacks

A brute-force attack attempts guessing *all* possible passwords until a successful match is found. These attacks often order the search by first guessing passwords with simple characteristics, such as those of a certain length or those composed of only lowercase characters. The classic brute-force attack is only limited by the attacker's computer speed; however, rainbow tables and acoustic enhancements provide faster search times as outlined below.

Rainbow Tables: A Time-Space Tradeoff

The time-space tradeoff in guessing attacks describes the time savings that can be achieved by pre-computing and storing guesses with their hashed values: more pre-computation saves time, but requires more space. The time-space tradeoff describes finding an optimal amount of pre-computation such that an attack is feasible according to the attacker's resources. "Rainbow tables" [109] are the most recent optimization to the time-space tradeoff.

Rainbow tables are based on the probabilistic idea of hashing chains, first described by Hellman [62]. The basic idea of hashing chains is that a reduction function f is used to make a pseudo-random walk through the entire (finite) space of passwords P. The function f is applied to an initial hashed password guess $H(x_0)$, which then

with physical access to boot into an operating system other than that installed on the machine. This effectively disables the protection for sensitive files that is particular to the installed operating system, and an attacker with physical access to a machine can thus boot into the live CD to access its files.

converts it to another key (password); the hash and reduction functions are applied to each key (repeated t times) to create a chain of keys and hashed keys (see Figure 2.1). Tables of these chains are stored (only as the first and last key in each chain) to allow fast password lookups. Only storing the first and last key in each chain significantly reduces the required storage space. To find a password given its hash, the attacker only needs to repeatedly apply the hash function, and then f, and compare each key to the last key stored for each chain. Once a match is found, the entire chain can be recreated from the first key stored, and the password should exist in the chain (with high probability). Rainbow tables are an optimization to make chain merges within the same table unlikely, by using a successive reduction function for each point in the chain (see Figure 2.2). This optimization fixes the chain lengths (which are variable in other hash chaining methods) and thus can reduce the search time by half (or more). There is a time-memory tradeoff when defining the length of the hash chains t, discussed in detail by Oechslin [109]. Previous hashing chain based methods used cycle detection [62] or distinguished points [36], leading to potentially longer hash chains.

$$\begin{tikzcd} x_0 \arrow[r] & H(x_0) \arrow[r] & x_1 = f(H(x_0)) \arrow[r] & H(x_1) \arrow[r] & x_2 = f(H(x_1)) \arrow[r] & \dots \arrow[r] & x_t = f(H(x_{t-1})) \end{tikzcd}$$

Figure 2.1: Hash chain: a chain of t passwords (and hashed passwords) are created using the reduction function f. Only x_0 and the final hash are stored in the final table.

$$\begin{tikzcd} x_0 \arrow[r] & H(x_0) \arrow[r] & x_1 = f_1(H(x_0)) \arrow[r] & H(x_1) \arrow[r] & x_2 = f_2(H(x_1)) \arrow[r] & \dots \arrow[r] & x_t = f_t(H(x_{t-1})) \end{tikzcd}$$

Figure 2.2: Rainbow table chain: a chain of t passwords (and hashed passwords) are created using the reduction functions f_1 to f_{t-1}. Only x_0 and x_t are stored in the final table.

Rainbow tables have been implemented in a variety of freely available cracking programs including *RainbowCrack* [141], *Cain and Abel* [101], and *Ophcrack* [108]. Precomputed tables are available online (currently only for Windows NTLM, and MD5 passwords), most often for purchase (e.g., [129, 130]), but some are also free (e.g., through bittorrent from the Shmoo group [147]). Additionally, an MD5 Rainbow table password cracking implementation is available online [119]; in only 40 minutes, it can crack any password composed of 8 (or fewer) lower case characters and numbers. A 2007 large-scale study of user passwords [47] shows that the vast majority of passwords contain only lower-case letters, 20% are only digits, 10-20% are alphanumeric, and only a very small proportion contain special characters, indicating that this table would have high success rates in practice.

Introduction of this search optimization (and its widespread availability) has drastically reduced the amount of security we can expect from text passwords. Although salting (see Section 2.3.2) prevents attacks from a single pre-computed rainbow table, if the salt is too small (e.g., on the order of 12 bits), it is feasible for an attacker to generate a separate table for each salt value.

Acoustic Enhancements

Acoustic emanations from keyboards can be recorded and analyzed to recover the user's typed text, including passwords. Zhuang et al. [170] use unlabeled data to recover approximately 96% of individual characters, a substantial improvement over previous work by Asonov et al. [2] that only achieved a success rate of 80% using labeled data. The high success rates can create an ordering for a sequence of characters that are likely to be a password (e.g., after entering a URL and feasible username), for use in a guessing attack. They found that the ordering correctly guessed 90% of 5-character passwords in fewer than 20 guesses, and 80% of 10-character passwords in fewer than 75 guesses.

Their attack only requires 10 minutes of recording a user on a specific keyboard, and 30 minutes of computation (on a single Pentium IV 3.0GHz CPU with 1G memory), to obtain these high success rates. The attack groups different keystrokes according to their sound, and then further classifies those sounds according to their (language-specific) statistical frequencies and relationship to other letters. The shift, control, backspace, and caps lock keys are not accounted for in their work.

Zhuang et al.'s results are alarming, as they reduce the number of password guesses to be within the reach of both offline and online guessing attacks that do not lock-out accounts (see Section 2.3). Defenses against this attack include using a noise-maker (e.g., playing music) while logging in, using a quiet keyboard, and/or increased physical security to prevent acoustic recordings, which unfortunately are not possible in all environments.

2.2.2 Dictionary Attacks

A *dictionary attack* is a guessing attack involving candidate guesses from a prioritized list of "likely passwords". The difference between a brute-force and dictionary attack is that brute-force attacks exhaust the entire space, optionally with a crude ordering where smaller subspaces are guessed first (e.g., shorter passwords), whereas a dictionary attack only guesses passwords from a small subspace containing more probable passwords. Freely available password guessing programs (e.g., John the Ripper [114]) often start with a dictionary attack, and if unsuccessful, they continue with a brute-force attack. In Klein's case study [77], 25% of 14000 user passwords were found in a dictionary of only 3×10^6 words; the Morris Worm [143] used a dictionary of only 432 words in addition to the 1988 UNIX online dictionary (about 25000 words [144]) with remarkable success: some sites reported that 50% of passwords were correctly guessed. More recent studies by Yan et al. [169] found that 33-35% of passwords were guessed using a dictionary attack[3] when no password rules are suggested or implemented. This suggests that a password scheme's security is linked more closely to the size of its weak password subspaces 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}).

Some freely available programs also implement a "permutations" feature, whereby the passwords in the dictionary are also varied (and then guessed) according to a set of rules that can be modified by the attacker. Examples of permutations include substituting certain letters with numbers (e.g., "e" with "3"), changing lowercase letters to uppercase, and appending a number or special character.

[3]No details are provided regarding which passwords were in the dictionary, its size, and the computing resources required for completing the attack.

Fast Dictionary Attacks: Rainbow Table Hybrids

The rainbow table attacks discussed in Section 2.2.1 did not take advantage of patterns in user choice, but precomputed all possible passwords within a particular space (e.g., *all* passwords composed of 8 or fewer lowercase characters). Narayanan et al. [106] present a method to combine a dictionary with a rainbow table such that all elements in the key space are dictionary entries.

Their attack uses Markov models relating to letter distributions in the English language, and they posit that phonetic similarity with words in the user's native language contributes to memorability. They also model use of non-alphabetic characters with finite automata to apply the equivalent of permutation rules in most password cracking programs (e.g., try a special character only at the end of the password). Their ability to create a rainbow table from a dictionary depends on an algorithm to efficiently enumerate the remaining password space that satisfies their Markov models and finite automata.

Using a dictionary with 2 billion entries, they cracked 67.6% of passwords (from a database of 150). A regular Rainbow table of the same size cracked 27.5% of passwords from the same database. Use of their method produces faster and more effective attacks, demonstrating the importance of using large salts to prevent such pre-computation attacks.

2.3 Text Password Attack Defenses

A variety of defenses have been proposed in response to the attacks presented in Section 2.2. Such defenses include ensuring that users choose "stronger" passwords (see Section 2.3.1), and increasing the cost of each guess to an attacker (see Section 2.3.2). Defenses that are particular to online password guessing are discussed in Section 2.3.3, such as delaying the server's response time, locking accounts after a number of incorrect logins, and inhibiting automated guessing programs with Automated Turing Tests (ATTs).

2.3.1 Password Rules and Proactive Checking

Password Rules

Password rules are commonly guided by known password dictionaries. Such guidelines are normally communicated to users when creating or resetting a password. A common password rule is "Your password must be at least 8 characters and include at least one number and one uppercase character".

Proactive Password Checking

Proactive password checking is the act of checking a password's strength before it is accepted by the system (as opposed to checking password's strength after it has been accepted). Proactive password checking programs verify that users are complying with password rules, and that they are not choosing common "weak" passwords that are contained in selected password dictionaries. Proposed implementations include the use of Bloom filters [144], Markov models [33], decision trees [9], and algorithms to check entropy-related password properties [168].

2.3.2 Password Salting and Adaptive-Cost Hashing

Salting

Password salting is a commonly used method that helps protect against pre-computation of an offline attack dictionary. The concept of salting was first introduced for UNIX passwords in 1979 [92, Note 10.2]. Salting stores a random value (e.g., 12 bits in the case of UNIX passwords circa 1979) in the password file for each user, which is appended to the user's entered password prior to hashing. Thus, if an attacker is to guess Alice's password, he must first look up Alice's salt value from the stolen password file, and append that value to each guess before hashing. For pre-computation (e.g., rainbow tables) to work against a salted password, a set of tables must be created for each possible salt value, increasing the time to compute the tables, and the space required. Thus, it is important that a large enough salt value is used, such that it is infeasible for most attackers to perform such precomputation.

Adaptive-Cost

Adaptive-cost hashing algorithms allow system administrators to define how much time it should take to hash a password for comparison, to slow down an attacker making repeated guesses. Bcrypt [128] is one such adaptive-cost method, which is implemented in FreeBSD. Bcrypt works by allowing system administrators to parameterize the computational complexity of the hashing algorithm itself, and thus of each guess in an attack, so that its protection may adapt over time to increasing computer speeds. Another adaptive-cost hashing algorithm for web passwords is presented by Halderman et al. [59] that parameterizes the number of times a password is repeatedly hashed. The idea of "key-stretching" by repeatedly hashing a password is attributed to Kelsey et al. [75]. Halderman et al. use this concept in a two-step process for use in a password manager for web passwords. The process is as follows: (1) compute and cache an intermediate value $V = f^{k_1}(username : password)$, then (2) compute $sitePassword = f^{k_2}(siteName : password : V)$, where f is a hash function, and k_1 and k_2 represent the number of repeated hashes in each step. Having these two steps allows caching of the intermediate hash V upon first use of their browser plugin, so k_1 can be much larger than k_2, and the k_1 delay in login time is not noticeable for the legitimate user on their regular computing device(s).

A more recent proposal in this area is to introduce a "Halting Password Puzzle" [18], which allows the user to specify the number of times that a password is hashed, by specifying the amount of time they are willing to wait while the password is repeatedly hashed. This method is particularly nice (although its usability remains to be studied) as it defines the number of hashes as user-perceived elapsed time, and thus it naturally scales with CPU speeds. This allows users to choose a weaker password that is easier to remember for an infrequently used account, and add extra security by accepting a longer login time. The algorithm will never finish hashing when an incorrect password is entered, so the attacker must guess what is a reasonable upper bound of hashing time, making this upper bound the new time for each guess. The usability of halting password puzzles should be studied, as it is not obvious that users would understand the relationship between the strength of their password and the amount of time they wait before stopping the hashing. It seems that any implementation would require an interface that communicates this concept of a time-security tradeoff to users, but any such interface would require user studies to ensure its usability and efficacy.

2.3.3 Online Defenses

Systems have slightly more control over the handling of online attacks; administrators can introduce delays between incorrect login attempts to increase the time to run a guessing attack, or completely lock an account after a number of incorrect login attempts. However, account locking is not a viable solution in certain environments such as online auctions where denial-of-service (DOS) attacks are a concern, or when the hosting company cannot afford customer service costs for resetting locked accounts. Also, inducing a delayed response from the server cannot protect against global attacks, where the attacker is trying all (or many) accounts on a system in parallel, which means the attacker does not suffer the latency penalty. These problems motivate the proposal of using Automated Turing Tests to increase the difficulty of an online automated guessing attack (further discussed below).

Automated Turing Tests (ATTs)

Pinkas and Sander [125] first introduced the idea of using ATTs (a.k.a. CAPTCHAs [158]) to increase the cost of online guessing attacks. ATTs are computations that are easy for humans, difficult for computers to solve, and should be difficult to guess correctly. Commonly, ATTs are images of words distorted such that it is difficult for optical character recognition (and other specialized) software to process correctly, but easy for most human users to read. The Pinkas-Sander protocol requires that users answer an ATT after entering a correct username/password at least once on a particular machine (after which a cookie is set, allowing a number of logins without an ATT). When the user (or attacker) enters an incorrect username/password pair, he will be asked to answer an ATT with probability $0 < p \leq 1$, and denied access regardless of the answer. With probability $1 - p$, access is denied immediately. The decision to ask an ATT for a given username/password pair must be deterministic to ensure a certain number of ATTs will be asked to complete an online dictionary attack, thus forcing human involvement.

Van Oorschot and Stubblebine [111] propose further enhancements to the Pinkas-Sander protocol to increase its security and usability. They make use of failed-login counts (parameterized for a particular user) such that legitimate users will rarely need to answer an ATT when no cookie is present. Thus, a travelling user in an Internet café, who has rarely provided an incorrect password, does not need to answer an ATT, with negligible cost to security. They also make use of failed-login counts to increase the cost of an attack; when a certain failed login threshold has been exceeded, all incorrect guesses require an ATT.

2.4 Graphical Passwords

Graphical passwords require that users remember pictures, and/or information related to a picture, in lieu of a word. Motivated in part by people's well-known ability to remember images better than words, they have been hypothesized to have higher entropy than text passwords, and thus provide better security. Many variations of graphical password exist; those that will be referenced extensively in Chapters 5 and 6 are described below. Other graphical password schemes are described further in Chapter 3, and are compared in Chapter 8.

2.4.1 Draw-A-Secret: A Pure-Recall Scheme

"Draw-A-Secret" (DAS) by Jermyn et al. [72] requires that a user recall and reproduce a drawing (using a stylus our mouse) on a presented grid, as illustrated in Figure 2.3. DAS encompasses both a general idea – user drawings as passwords – and a specific grid-based method to implement that idea (i.e., the encoding that maps a user drawing into an exactly repeatable password) [72, 97]. To distinguish these concepts, we will refer to the specific encoding scheme of Jermyn et al. as DAS_J, and hereafter reserve the term DAS for the general idea. DAS$_J$ decouples the position of password input from the temporal order, producing a larger password space than text-based password schemes constrained by keyboard input (where the order in which characters are typed predetermines their position).

A DAS password is simply a 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$_J$, an

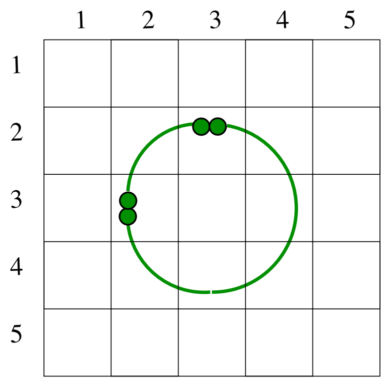


Figure 2.3: DAS: users draw a password on a presented grid.

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.[4] Drawings are divided into equivalence classes in this manner.

We reuse the following terminology.

- The *neighbors* $N_{(x,y)}$ 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$_J$ *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$_J$ password* is the sum of the lengths of its strokes (excluding pen-ups).

Jermyn et al. [72] recursively compute the (full) password space size, i.e., the number of distinct encoded graphical passwords in DAS$_J$. This gives an upper bound

[4]This implies a many-to-one mapping of user drawings to encoded DAS$_J$ passwords.

on the size of the 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, the size of the full password space is 2^{58}, exceeding the number of text-based passwords of 8 characters or fewer constructed from the printable ASCII codes ($\sum_{i=1}^{8} 95^i < 2^{53}$). It is this large full password space that makes DAS$_J$ particularly interesting for further analysis.

2.4.2 PassPoints: A Cued-Recall Scheme

PassPoints is a click-based graphical password scheme, based on a single background image (see Figure 2.4). The user chooses a password by clicking on an ordered sequence of 5 points on the background image, and logs in by clicking that same sequence of points again (where each click must be within some predefined amount of error from its counterpart within the original set of click-points). The user's recall is cued by presenting the background image on which they enter their click-points. Birget et al. [13] originally proposed a method called "robust discretization" to enable secure storage of PassPoints passwords as a one-way hash, while permitting some amount of error. Allowing some amount of error is necessary for PassPoints, as it would be extremely difficult (if not impossible) for a user to click on the exact same pixels during subsequent logins.

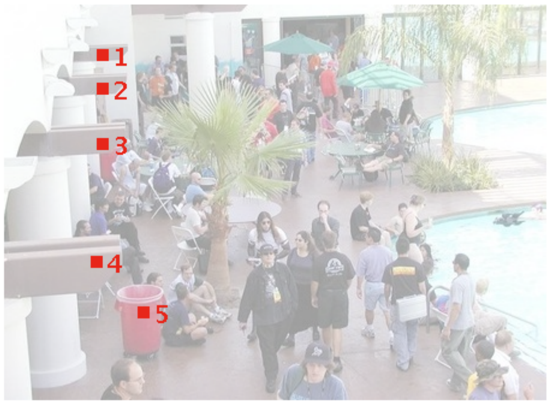


Figure 2.4: PassPoints: A user chooses an ordered sequence of 5 points on a single background image.

We use the following terminology. Assume a user chooses a given click-point c as part of their password. The *tolerable error* or *tolerance* t is the error allowed for a click-point entered on a subsequent login to be accepted as c. This defines a *tolerance region (T-region)* centered on c, which for example using $t = 9$ pixels (in both vertical and horizontal directions) is a 19×19 pixel square. A *cluster* is a set of one or more click-points that lie within a T-region. The number of click-points belonging to a cluster is its *size*. A *hot-spot* is defined as an area that users choose with higher probability (than if all points were equi-probable), and is thus measured by user choice, as opposed to a hypothesized location on the image. A hot-spot is thus indicated by a cluster observed (from a number of user password selections) to be larger than would be expected given random selection with the same sample size. An *alphabet* is a set of distinct T-regions. Both our own (in this thesis) and earlier PassPoints studies [164, 163, 162] use 451×331 pixel background images; using 19×19 pixel T-regions produces an alphabet of size $m = 414$. Using passwords composed of 5-clicks, on an alphabet of size 414, provides the system with only a 43-bit full password space (however this could be increased by increasing the image size); we discuss the implications of this in Chapter 6. In practice however, as we will show, the effective password space is smaller as users choose hot-spots as click-points and/or have predictable dependencies between their click-points.

The theoretical security of PassPoints depends on its alphabet size, and the number of click-points. The alphabet size depends on the image size and error tolerance used. Weidenbeck et al. [163] found that users could successfully re-enter their points with a 20×20 pixel error tolerance, and a 14×14 error tolerance, but not as well with a 10×10 error tolerance. Chiasson et al. [24] later found that most users re-entered their passwords within a 9×9 pixel T-region. The security effect (in practice when considering user choice) of using a larger image is unknown.

PassPoints' usability was originally studied by Weidenbeck et al. [164, 163, 162], and further by Chiasson et al. [24]. These findings show that PassPoints have usability competitive with regular text passwords, making it of interest for further analysis.

2.5 Summary

User authentication is an important part of computer security. Unfortunately, ubiquitous text passwords are susceptible to guessing attacks. Countermeasures have been proposed, but have a limited impact when users choose weak passwords. Graphical passwords have been presented as an alternative; the goal of these methods to date has been to increase the size of the effective password space, while retaining usability. The two particular schemes discussed in this Chapter (DAS and PassPoints) are the subjects of the attacks and security analyses performed in the body of this thesis. Chapter 3 discusses other related knowledge-based schemes and attack methods that are not required background for this thesis.

Chapter 3

Related Work

3.1 Introduction

Many alternatives to text-based passwords have been proposed to date; Section 3.2 reviews other knowledge-based schemes that were not covered in Chapter 2. Some of these methods have been shown to be vulnerable to attack, as discussed in Section 3.3. Other schemes that do not require the user to remember anything have also been proposed (e.g., biometrics and physical tokens); however, we do not focus on these schemes in this chapter.

3.2 Other Knowledge-based Schemes

Generally, the common goal of alternative knowledge-based schemes is to increase memorability, and thus usability and effective security. We group these schemes into mnemonic text password strategies in Section 3.2.1, and graphical passwords in Section 3.2.2.

3.2.1 Mnemonic Text Password Strategies

Passphrases

Passphrases are a strategy suggested to users to aid in the creation of strong passwords. A typical passphrase is created by asking the user to think of a phrase such as “My dog Spot is 7 years old!”, and using that phrase to create a password by using the first character of each word, including any numbers or special characters. The aforementioned example phrase would thus create the password “MdSi7yo!”. This method is suggested as it creates passwords that are not typically in conventional password dictionaries (as they are not based on a single word), yet are easy for people to remember.

Yan et al. [169] performed a study to verify that using passphrases did result in passwords that are both secure (as measured by resistance to their attack methods) and memorable. They performed a study involving 288 first year science students from Cambridge University, by dividing them into three groups: control, random, and passphrase. After one month, Yan et al. [169] took a snapshot of the password file, and ran a dictionary attack with permutations (recall Section 2.2.2) and user-specific information, and a brute-force attack on all 1-6 character passwords. They ran these attacks against each experimental group, plus another comparison group of 100 students who did not receive any instructions. Using this attack, they were able to guess the passwords of 33% of the control group, 35% of the comparison group, 8% of the random group, and 6% of the passphrase group. This finding supports that using passphrases produces passwords that are less susceptible to dictionary attack (although Kuo et al. [82] later found otherwise, see Section 3.3.2). They also found that passphrases had similar memorability to regular passwords; on average, students in the passphrase and control groups felt their passwords were easier to remember, and kept written copies of their passwords for less than one week, as opposed to almost five weeks for the random password group.

Picture Cues

Stubblefield et al. [145] propose *inkblot authentication*, whereby the user is shown "inkblots" to cue their memory. Inkblots are random-looking images, created by blotting pen ink on a piece of paper and folding it (e.g., see Figure 3.1). The use of inkblots is motivated by the Rorschach inkblot tests, which asks the subject to tell the examiner what he or she "sees" in the inkblot; their responses are evaluated to characterize the personality of the subject. Passwords are created using inkblots by asking the user to create an association for each inkblot, and then type the first and last letters of the association. Thus, a set of 10 inkblot cues produces a 20-character long password.

A preliminary user study [145], involving both Rorschach and computer-generated inkblots, found that memorability for these associations was good after one day and again after one week. A frequency analysis was performed on the data, finding that

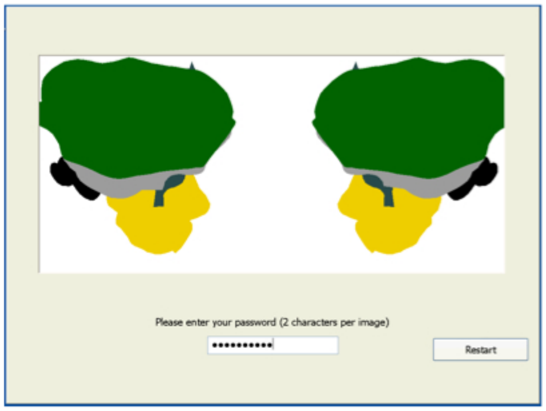


Figure 3.1: Inkblot authentication [145]: user is cued by a sequence of inkblots to remember the characters of their password.

each character provided just under 3 bits of entropy; thus, a set of 10 inkblot cues produces a password with about 59 bits of entropy (since each inkblot has two associated characters). They also performed a frequency analysis of the characters, finding them quite similar to the English language. Finally, they found that by guessing associations from most to least probable, it took 5 million guesses before the first blot was guessed. Unfortunately, they did not provide more information about the success rate from continuing the attack for a longer time.

3.2.2 Other Graphical Password Schemes

Recognition-Based Schemes

In this section we review several recognition-based schemes. As mentioned by Davis et al. [34], all of the schemes below have small full password spaces, and are thus vulnerable to offline attacks. Thus, these schemes are only secure for systems implemented in environments that are presumably safe from offline attack (e.g., ATM machines).

Déjà Vu Déjà vu [38] is based on a user recognizing 5 pictures out of a single challenge set of 25 presented in a panel (see Figure 3.2). The system challenges the user with a single panel during a login session. The 100 images from which users choose their 5 during enrollment are generated by random art, which is seeded with an 8-byte number. The 100 images are then hand-picked such that they are distinguishable from one another (about 70% of all generated were considered to have sufficient quality for the scheme) .

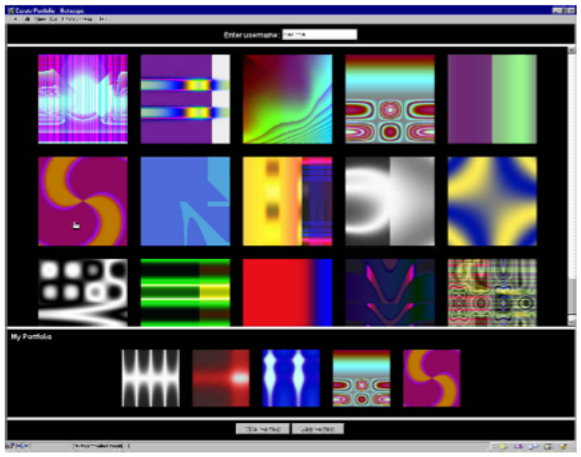


Figure 3.2: Déjà vu [38]: user selects five images.

Dhamija et al. [38] did a user study involving 20 participants to determine the difference between 4 digit PINs, 6-character passwords (never used before), and image portfolio schemes (one based on photos, the other based on random art). Few users of the 20 chose the same random art image, although 9 out of 20 chose the Golden Gate bridge out of the photograph scheme. This implies attraction to random art is much more randomly distributed, and thus shows more promise for security than other recognition schemes; however, a larger sample size would be necessary to strengthen this conclusion.

Passfaces *Passfaces* [131] requires that a user select a set of human faces to remember. The scheme is motivated by the assumption that humans have an exceptional ability to recall human faces [132]. To login, the user is presented a panel of nine images, containing eight decoy faces, and one face from the user's password. The user must correctly select their password face from the presented panel; a total of four panels are presented for a single login session. User studies performed on Passfaces found that login success rates were higher than text passwords [20], and that a modified version (called “Faces”) had higher login success rates than the Story scheme [34].

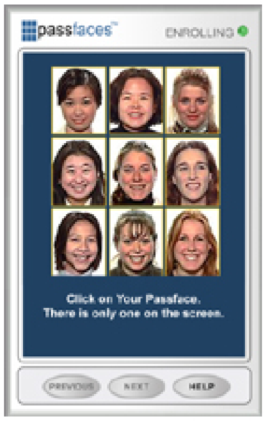


Figure 3.3: Passfaces [131].

Story Examined by Davis et al. [34], Story presents the user a panel of nine images on a 3×3 grid (see Figure 3.4), and asks the user to select one image on each of four panels presented. The images presented belong to different categories, such as men, women, children, food, animals, etc. A mnemonic strategy is suggested to create a story from the pictures chosen. However, the authors found that this scheme was not as memorable as their version of Passfaces, possibly since many subjects did not actually base their password on a story (as revealed by a closing questionnaire).

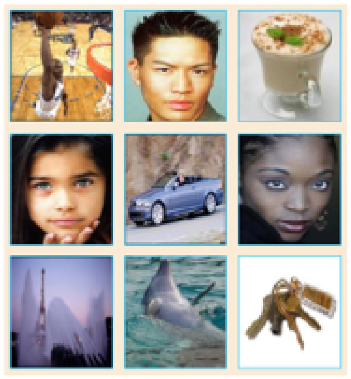


Figure 3.4: Story [34].

Other Pure Recall Schemes

Pass-Go In terms of the user drawing, Pass-Go [146] may be viewed as an implementation of DAS, wherein the feedback to the user is normalized to better reflect the underlying encoding. To compensate for the rigidity of this normalization, Pass-Go allows for strokes to also connect to diagonal neighbors. The user drawing is normalized to points on cell corners, as opposed to DAS_J's normalizing to points on grid cell centers. This implementation design arguably addresses the potential repeatability issue in DAS_J when users draw passwords that are too close to grid lines and corners.

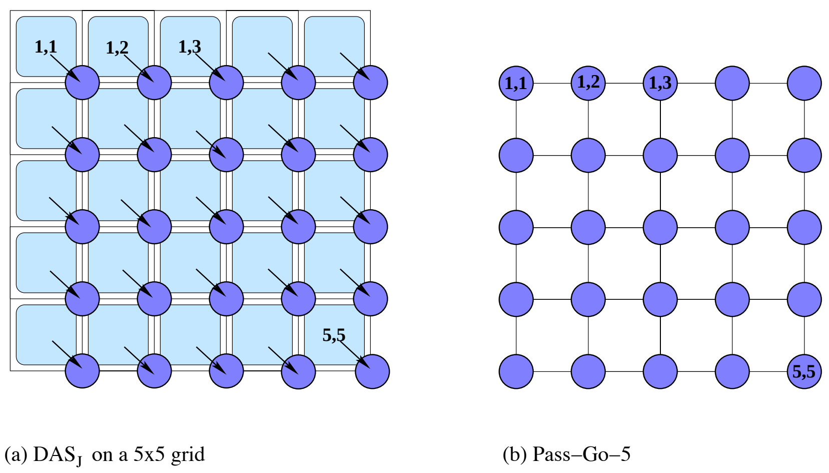


Figure 3.5: Mapping from (a) DAS_J to (b) Pass-Go.

There is a one-to-one mapping between the corners of a Pass-Go-5 grid and the cells of a 5×5 DAS grid as shown in Figure 3.5. The difference is in what strokes are valid; DAS_J permits strokes between horizontal and vertical neighbors, whereas Pass-Go also permits strokes between diagonal neighbors. Thus in Pass-Go every point has up to 8 neighbors versus at most 4 in DAS_J. This results in a full password space increase of approximately one bit when the maximum password length is 12. The reason for this surprisingly small increase is that the password space is dominated by all permutations of length-1 strokes (which do not connect to any neighbors).[1]

[1]Through enumerations of the Pass-Go space under different stroke-length conditions, we found this difference is more noticeable in the space of passwords without any length-1 strokes; here there is an increase of 6.2 bits for a maximum password length of 12 over original 5×5 DAS_J.

Other Cued Recall Schemes

Blonder's Scheme Blonder [15] patented a "graphical password" that requires users to click an ordered sequence of predefined image areas (tap regions; see Figure 3.6). The password space of this scheme doesn't leverage all of the available areas of the background image; only a small number of tap regions are available for a user to choose from.

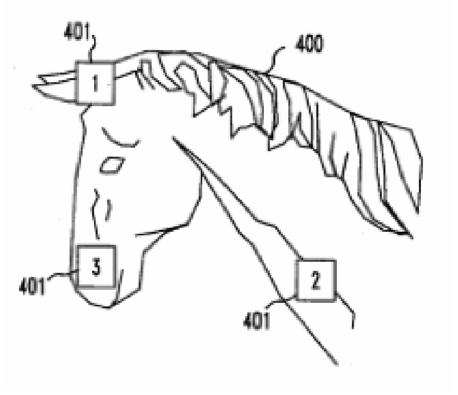


Figure 3.6: Blonder's graphical password scheme [15].

Picture Password Jansen et al. [71] proposed a graphical password scheme intended for PDAs, that requires users to click an ordered sequence of visible squares on a background image (see Figure 3.7). The visible squares are the result of placing a grid over the entire image, which provides the user with an understanding of how to repeat their click-points in subsequent logins. To increase the alphabet size (and thus the full password space), they propose allowing two different actions: selecting a grid square, and selecting a pair of grid squares (e.g., by a drag and drop operation). The grouping of a pair of grid squares does increase the alphabet size and thus the theoretical security, but at a cost to usability, as there are more operations for the user to enter, which would require more time to enter (and there are more pieces of information for the user to recall).

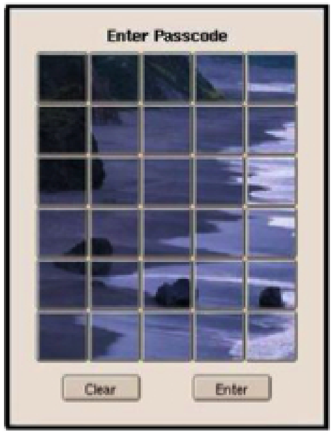


Figure 3.7: Picture Password [71].

V-Go Created by Passlogix [120], this scheme allows users to create passwords by clicking on objects in a picture in the correct sequence, such as entering the time on a clock, drawing cards from a card deck, or selecting ingredients for a meal or drink. This scheme is unlikely to have a large password space, as there are a limited number of things the user can select, and it encourages patterns in user choice (e.g. all aces in a deck). It appears this scheme may have been abandoned by Passlogix; the only place their website now discusses graphical passwords are in old press releases.

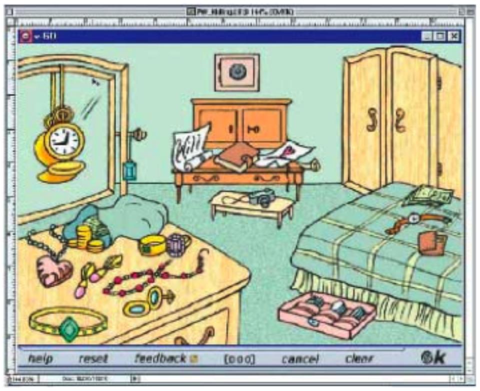


Figure 3.8: V-Go by Passlogix [120].

VisKey VisKey is a commercial graphical password product, marketed for the Pocket PC [139]. It appears quite similar to PassPoints, as users are permitted to click anywhere, and can set their own error tolerance. The images can be defined by the user, but unfortunately no advice is given to users on choosing a good image or password (for example, see Figure 3.9).

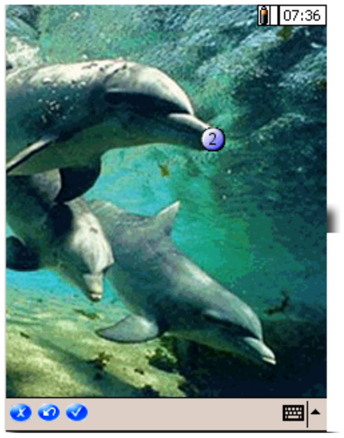


Figure 3.9: VisKey: the demo password is (clockwise, starting at the lower left) the nose of the three dolphins. [139].

Cued Click Points (CCP) Chiasson et al. [25] propose this variant, where instead of clicking on five points on a single image as in PassPoints, the user clicks on a single point on each of five images (see Figure 3.10). The general idea is that each image (after the first) is dependent on the previous click-point. The attacker can obtain the first image in the sequence "for free" by entering the username, but not the following four images. This scheme has a similar full password space to PassPoints, but may prove more difficult to attack as it increases the work for an attacker to obtain the background images for analysis (a pre-requisite for the attacks we present herein for PassPoints in Chapter 6). The effect of hot-spotting and dependencies between click-points may lead to an attacker guessing all 5 points, but the efficacy of such a directed attack has not yet been examined. A preliminary analysis of this scheme [23] has shown that images have hot-spotting similar to that found for PassPoints in a comparable lab setting. In this same technical report, Chiasson et al. [23] show that using a persuasive method helps reduce this hot-spotting in a single-session lab setting.

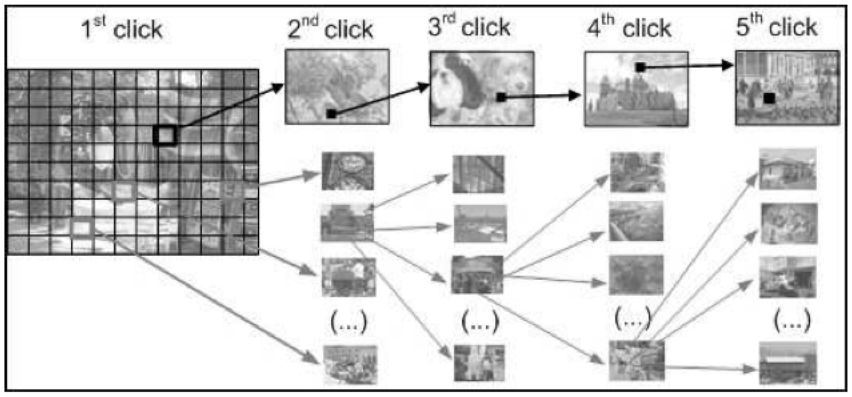


Figure 3.10: Cued Click Points (CCP) [25].

Challenge Response Schemes

Some schemes are a combination of the user remembering a shared secret, and being required to compute something based on a challenge provided by the login server (or device). The motivation for such schemes is that they do not reveal the shared secret in an untrusted environment (e.g., where shoulder surfing might be possible). The user's computed response to the challenge provided by the login server is a proof that they remember the shared secret. One such method called *cognitive trapdoor games* was proposed by Roth et al. [136] for PIN numbers. Other (less usable) examples include those proposed by Hopper et al. [63] and Li et al. [85].

Here we outline one such challenge-response scheme involving images.

Weinshall's Protocols Weinshall [161] proposes a "high-complexity" and a "low-complexity" protocol, based on a shared secret set of pictures, and human cognitive abilities. In general, N pictures comprise the set of possible shared secrets. In a given login session, B pictures are shown. The secret picture set is $F \subset B$, of size $M < N$. Login consists of a set of challenge-response rounds, where in each round, the user must answer a question based on knowledge of the subset F.

In the high-complexity protocol, the user is shown a grid of images (see Figure 3.11) and computes a path from the top-left to the lower-right, based on moving downward at the images in F, and to the right at the images not in F. The user enters the number at the exit point of the panel (either on bottom or right edge). A user is presented with $n < N$ pictures randomly selected from B, in the form of a rectangular panel. Weinshall suggests using 11 rounds, with $N = 80$, $M = 30$, and $n = N$.

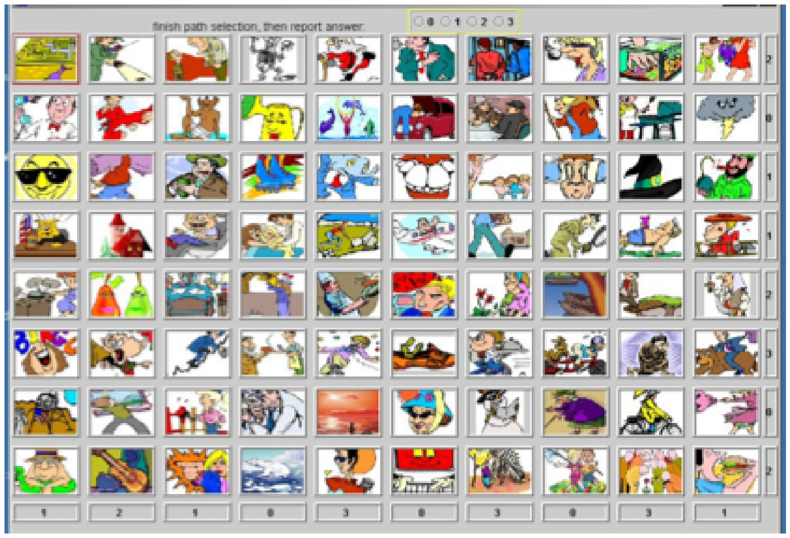


Figure 3.11: High-complexity Protocol: users compute a path through the grid based on a shared secret set of images [161].

In the low-complexity protocol, the user is shown a grid of 20 images, with a random bit beside each. The user is then asked queries about the numbers beside their secret images on the grid, such as “What is the majority value of the bits associated with the first, second, and last images?”. Weinshall suggests using 22 rounds, with $N = 240$, $M = 60$, and $n = 20$.

Both of these protocols have usability drawbacks: the user must be able to recognize a large number of images, an extensive training period is required (2-3 sessions in a secure location where feedback is provided), and login times are quite long (3 minutes for the high-complexity query, and 1.5 minutes for the low-complexity query). Furthermore, a quite severe attack was found by Golle et al. [55] (see Section 3.3.5) that is possible after observing a number of login sessions.

3.3 Established Attack Methods for Knowledge-based Schemes

3.3.1 Text Passwords

There is a large body of work demonstrating how easily text passwords are guessed, dating from Klein's 1995 study [77] until 2006 in work by Kuo et al. [82].

Using a dictionary attack, Klein [77] showed that 25% of over 13,797 user passwords were guessed in 12 *DECstation 3100* CPU months, but further that 21% were guessed in 1 week, and 2.7% in the first 15 minutes. Although these times are surely different from now (due to increased computer speeds), the dictionary success rates may be similar. He also notes that on an average system with 50 accounts, one could expect (using similar computing resources) one account to be cracked in under 2 minutes, and 5-15 accounts to be cracked in the first day. His attack dictionary was 62,727 words, composed of 130 variations of name and account name (of the target user account), words from dictionaries such as names and places, vulgar phrases, and Chinese language passwords (single and double syllables in Pinyin Romanization). Various permutations (such as substituting numbers, pluralizations, and adding suffixes) were applied to this dictionary.

Wu [167] further examined the issue of dictionary attack in text passwords in the context of Kerberos. The most interesting part of his study is that the accounts examined were subject to password policies, so it examines how well such policies prevent attacks. There were 25,000 user accounts in this study, 2045 (8.2%) of which were cracked in 100 million guesses. He performed a dictionary attack using a publicly available password cracker, with site-specific information added to the dictionaries. Although his results are not as dramatic as Klein's, they demonstrate that password policies are not a silver bullet in preventing dictionary attack. Wu's conclusion was that dictionary attacks were best avoided by cryptography instead of administrative policy.

Kuo et al. [82] examined the efficacy of using John the Ripper [114] with permutations (recall Section 2.2.2) against a database of 146 passwords. The attack guessed 6% of user passwords with a 1.2 million-entry dictionary without permutations, and another 5% by adding the permutations, stopping after 72 hours. This attack demonstrated that user choices have similar entropy to those chosen 8 years ago in Wu's study.

St. Clair et al. [26] performed a distributed attack against 3500 passwords used at Pennsylvania State University's Computer Science and Engineering department. They found that, using John the Ripper with 20 dual-core processors, they were able to guess 25% of user passwords in 2 hours, and almost 34% in 5 days. Interestingly, they found that only 10% of passwords were found using a dictionary, and the others were found using John the Ripper's intelligent brute force method (which uses frequency tables). Their dictionary attack result is similar to the findings of Kuo et al. [82], although they do not specify which dictionary was used.

3.3.2 Passphrases

Kuo et al. [82] also examined the feasibility of guessing passwords created with passphrases, a mnemonic strategy found by Yan et al. [169] (see Section 3.2.1) to result in passwords that are both difficult to guess, and easy to remember. Kuo et al. hypothesized that users will choose passphrases based on advertising slogans, children's nursery rhymes or songs, movie quotes, famous quotations, song lyrics, and television theme song lyrics. They created a 400,000-entry dictionary of passphrase-generated passwords, including permutations (e.g., a 4 in place of the word "four"), and applied it to a database of 144 passwords that were created under the suggestion to use a passphrase as a mnemonic. They successfully guessed 4% of these passwords with this dictionary, showing that although they are more resistant than regular text passwords to such attacks, they should not be considered a panacea. They also found that the most common sources to base passphrases upon are music and literature, followed by movies and television shows. It is feasible that their dictionary could be optimized by the addition of more passphrases, particularly using phrases that are freely available (and harvestable) on the Internet, since 65% of the phrases used were found to be available on the Internet.

3.3.3 Recognition-Based Graphical Passwords

Davis et al. [34] examine the impact of user choice in two recognition-based graphical password schemes: Faces (their implementation of Passfaces), and Story.

Their hypothesis was that, because people (across cultures) agree on beauty and are better able to recognize faces of people from their own race, people will tend to choose attractive faces and faces that are members of their own race in graphical passwords. That this might be the case was previously noted by Dhajma et al. [38]. They further hypothesize that users will choose their pictures based on a dependent probability: each picture depends on the one selected in the previous panel. They designed their study to verify their hypothesis by ensuring each panel had at least 9 (of 12) different groups, divided into 3 races, 2 genders, and 2 levels of “attractiveness”. The “attractive” images were of models, and the other images were of “typical” faces.

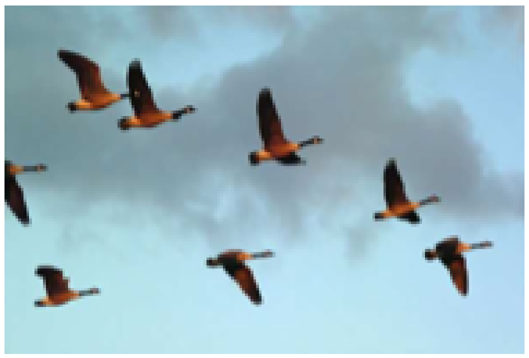
Their study was of 154 students using the system to access homework, grades, course materials, etc.They generated an ordering of these graphical passwords, based on dependent probabilities from the individual images from 80% of their user passwords. They applied this ordering (from most to least probable) to the other 20% of users, finding that it lead to successful guessing attacks. Most notable is that for Faces, they found they could guess 10% of male user’s passwords in two guesses, making the scheme vulnerable to online attack even when account locking is enabled. For Story, they found that they could guess 10% of all user’s passwords within 35 guesses.

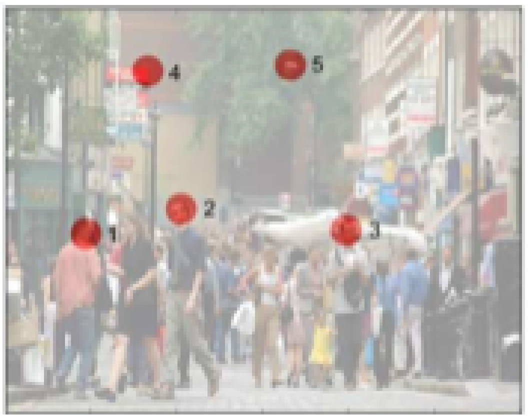
They found that the patterns they hypothesized as likely were highly probable, with most users choosing faces from their own race (62% for white males), and with the models more probable than the typical faces (63.2% of males chose female models). Additionally, they found that both males and females were more likely to choose female faces (75.9% for males and 68.8% for females). Their conclusion recommended that user choice should not be allowed in PassFaces-like graphical password system, without some way to mitigate the effects of race and attraction.

3.3.4 Cued-Recall Graphical Passwords

Dirik et al. [40] examine an image-processing based method to determine the probability of single click-points in PassPoints.[2] Their method is quite complimentary to ours (see Section 6.3.1), as it uses two different image processing techniques. Their method uses segmentation to pick out objects, calculates centroids for each object, and then uses intensity, colour, and foreground to rank each centroid. Robust discretization [13] was the assumed encoding method, and each grid square cell was ranked according to the sum of the centriods in its area, and then used in a guessing attack.

They tested their technique against two different images: *birds* and *people* (see Figure 3.12). Their study gathered 254 passwords (one password per user) in a single session (92 for *birds* and 142 for *people*), but did not test for memorability and thus their data is only for user's initial choice and does not account for the effect of resets over time. Indeed, our studies (see Sections 6.3.2 and 6.5.1) found that hot-spotting differed between a single-session lab study and a field study for some images (cf. *cars* in Sections 6.7 and 6.5.6).


(a) *birds*.



(b) *people* (unmodified image unavailable).

Figure 3.12: Images used by Dirik et al. [40].

Their results found that a 25-bit dictionary guessed 61% of passwords in the *birds* image (which has a low amount of detail; see Figure 3.12a), and a 32-bit dictionary

[2]This work emerged after the publication of our February 2007 technical report [152] (which lead to [151]); one part of our work described a different image processing method, as reported in Section 6.3.1 of this thesis, which also described the use of centroids as an avenue for improvement.

guessed 8% of passwords on the *people* image. We note that their method for the more detailed *people* image had similar success rates to some of our results (see Section 6.5.2; cf. *cars*) for the similarly detailed images used in our study.

3.3.5 Challenge-Response Graphical Passwords

Golle et al. [55] present an attack against Weinshall's high and low-complexity challenge-response protocols (recall Section 3.2.2). The attack requires observing a few successful login sessions. Their attack is based on the fact that every successful response to a challenge allows the adversary to learn a boolean relationship between the bits of the user's secret key (i.e., an array of N bits, where the i-th bit is set to 1 if the i-th picture belongs to the secret picture set F, and 0 otherwise). The observed relationships are then described in disjunctive normal form, which allows a SAT solver to recover the key (the user's secret set of images).

They show that for the high-complexity protocol, after observing 60 rounds, the suggested parameters can be broken in 102 seconds (for a single PC, dual 3.4 GHz CPUs and 1 GB of RAM). Fewer than 60 rounds recovers multiple candidates of the key. Recall that the high-complexity protocol suggested 11 rounds in a single login session, which means the attacker can learn the key by observing as few as six logins. For Weinshall's low-complexity protocol, 250-400 rounds must be observed to provide the unique key in less than a second. Recall that 22 rounds are suggested for the low complexity protocol, thus 12-19 successful logins are enough with the suggested parameters.

They examine the efficacy of their attack when increasing parameters for each protocol, concluding that these schemes are fundamentally vulnerable to attacks based on SAT solvers. For example, increasing the number of images a user must recall in the low complexity protocol to 150 is still within reach of the SAT solver.

Although this is a powerful attack, it depends on the attacker being able to observe a number of inputs (e.g., by shoulder surfing). Thus, the scheme may still offer security in environments where observation is assessed to be unlikely. Since the password is randomly assigned, it should be less vulnerable to guessing attack than text passwords; the attacker would need to guess which images are in the user's secret set. Despite this random assignment, user studies showed that they were still memorable, but this was achieved at the cost of a 2-3 session training time [161]. In this case, the security offered by the scheme is the size of the full space, which for Weinshall's [161] suggested parameters, is 73 bits for the high-complexity protocol, and 191 bits for the low complexity protocol.

3.4 Summary

Although many knowledge-based alternatives to text-based passwords have been proposed, a number of them have already been found to be vulnerable to attack as outlined in Section 3.3. Most of these attacks have something in common: they are dependent upon predictable patterns in user choice. Chapter 4 presents a generalization of these attack methods, with the goal of showing how they might be applied to other, previously unanalyzed schemes.

Chapter 4

Generalization of Attack Strategies for Knowledge-based Authentication

4.1 Introduction

The high success rate of brute-force dictionary attacks against text-based passwords is believed to be strongly related to the recall capabilities of humans and how this affects password selection: meaningful and thus more easily remembered strings are frequently chosen. This leads one to ask whether other types of passwords (e.g., graphical) are also vulnerable to dictionary attack due to users' tendencies to choose memorable passwords. For relatively new password schemes where there is an absence of large datasets from diverse populations, this thesis is motivated by the questions: (1) How might an attacker build an attack dictionary? (2) How successful would an attack using such a dictionary be?

Under the conjecture that available information relating to human memory and preferences might reveal higher-probability password choices,[1] we propose a general predictive method for modelling and defining *weak password subspaces* in Section 4.2. We show how this method maps to existing attack strategies in Section 4.3, and how this method might map to new (previously unanalyzed) schemes in Section 4.4. We expect that a clever attacker would prioritize a dictionary according to how preferable or easy the elements in the password are to remember, based on evidence from similar or related contexts.

[1]Although it is widely considered to be a fact that memorability and/or preference influence a user's password choices [102, 77, 72], and at least one (non-peer-reviewed) survey indicates that 84% of computer users consider memorability the most important attribute of a password (with 81% choosing a common word as a result)[61], to the best of our knowledge this has yet to be scientifically demonstrated.

4.2 Predictive Models

Since text-based password dictionaries focus on words people recall better, we are lead to consider analogous dictionaries for other knowledge-based password schemes (e.g., graphical) for which we lack knowledge of the distribution of user choice. We assume that users will choose passwords that minimize their complexity. We therefore model higher-probability passwords as those that have low complexity.

We suggest the following general predictive method for modelling user choice in knowledge-based authentication schemes (steps 3, 4, and 5 of which are similar to generative attacks for behavioural biometrics [6]).

1. Identify the tasks users are required to do during login, what type of demand this places on their memory (e.g., verbal recall, visual recall, recall of input order), and what sort of preferences[2] may apply.

2. Determine what relevant information is available about user's memory and preferences (possibly from other contexts) regarding these identified demands.

3. Identify password *complexity properties* based on this information. We informally define a password *complexity property* to be a characteristic that affects password memorability or preference (and by conjecture, the chance of selection by users).

4. Use these properties to model classes of probable passwords.

5. Estimate the size of these classes; any computationally exhaustible subset of the password space is a candidate weak password subspace (see Definition 1).

Definition 1 ***(Weak password subspace).*** *We define a weak password subspace as a subset W of a password space P that exhibits the following three properties.*

(1) (Smallness of W). $|W| \leq t_e$, where t_e is a threshold number of passwords that may be tested within an adversary's computational resources and environment.[3]

[2]Examples of preferences include attractiveness and familiarity.

[3]Throughout this thesis, we measure the size of a password space (or subspace) in terms of *bits*, i.e., log_2 of the number of entries.

(2) (Significance of W). $\sum_{w_i \in W} p(w_i) \geq \alpha$, where α is an upper bound on the "tolerable" probability of a password compromise (over a period of time commensurate with the resource expenditure in 1).[4]

(3) (Generability of W). W has defining characteristics that allow generation of all of its passwords.

Informally, we define a *weak password* as any password that falls within a weak password subspace. Weak passwords are similarly informally defined by Spafford [144].

Weak password subspaces are a general concept which extend beyond password dictionaries; informally, they are subsets (of a space of secrets) whose elements are more readily guessed. Other examples of this concept include weak keys in symmetric cryptosystems [31] and weak RSA primes (e.g., see [92, Note 8.8]).

It follows from Definition 1 that for a scheme to be free of weak password subspaces, it is necessary for it to have a large full password space (if less than t_e, the full password space itself is weak); however, this is not sufficient due to what an attacker might be able to predict about user choice. Different systems can tolerate different levels of risk; this is captured by the parameter α which represents the tolerable probability for an attacker's success. For example, a user might feel that it is tolerable for an attacker to have a probability 2^{-10} of compromising their email password, whereas government and banking servers might require a probability of at most 2^{-30}. An administrator of a system with many accounts might desire an even lower probability of compromise, such that it accounts for the collective probability of any one account being compromised (e.g., if there were 20 accounts on the system, he would surely want α to be less than 0.05).

[4]This condition is naturally related to that of entropy [140, 90].

4.3 How Predictive Models Map to Established Attack Methods

We believe our predictive models can apply to knowledge-based authentication schemes that permit user choice. Table 4.1 is a selective summary of how these predictive models (as outlined in Section 4.2) can map to the established attack methods of some known password schemes (outlined in Section 3.3). We do not consider the challenge-response attacks of Section 3.3.5 in this context, as the attack is based on the information leaked from observing a set of login sessions, rather than patterns in user choice.

Each of the dictionaries discussed in Table 4.1 are weak password subspaces, although the success rates are quite different. Each attack method satisfies property (1) in that they are small enough to be computationally exhaustible, and property (3) in that they are generable. Note that number of bits shown is the log_2 of the number of entries in the dictionary. According to various studies, each dictionary would guess a percentage of passwords that is significant when $\alpha \leq 0.04$. The most recent study [82] has shown that (without permutations) 5% of text passwords can be guessed in 1.2 million guesses, and 4% of passphrases can be guessed in 400,000 guesses.

For recognition-based graphical passwords, Davis et al. [34] do not explicitly evaluate the sizes and success rates of the dictionaries we mention in Table 4.1, but focus on a different (training-based) style of attack that requires a large sample of real passwords (the results were quite effective; details discussed in Section 6.6). In addition to the attacks they examine, we think some of the results they describe (i.e., the statistics on user choice of certain categories, for different demographics) could also map to a predictive model. The data provided by Davis et al. [34] allows estimation of the size and success of the predictive dictionaries mentioned in Table 4.1. For Faces, there were at most one of twelve categories of faces (white/black/asian, model/typical, and male/female) per panel; for example, there can be at most three female models (white, black, and asian) on a given panel and $3^4 = 81$. Although these particular numbers are based on their study, one could imagine a more generic method to determine which images on a panel belong to these categories (e.g., image processing or "human computation" [159]). If we assume their reported percentages in Tables 4-7 are independent for each panel, we estimate their success rates by $(.63)^4 = 16\%$ for

Step	Text Passwords	Passphrases	Recognition-based Graphical Passwords
1.	User must recall a text string.	User must recall a phrase.	User must recognize and select a set of images from a larger set.
	Demands verbal recall.	Demands phrase recall plus a method to convert into a string (e.g., first character of each word).	Demands visual recognition and (possibly) a mnemonic strategy to relate the images.
2.	Verbal recall for certain types of words (e.g., nouns). Preferences may involve words related to something they like (e.g., a sports team).	Verbal recall for phrases and word sequences. Preferences may relate to popularity and/or semantics.	Visual recognition abilities (and preferences) for certain types of image.
3.	A word's semantic meaning and concreteness (i.e., nouns are more concrete).	A phrase's popularity and/or semantic meaning.	An image's familiarity, degree of attractiveness, and associations between images (e.g., categories).
4.	Nouns and permutations (e.g., pluralization, common number placements).	Popular phrases and the easiest methods for constructing a text string from a phrase.	Which of presented set of images are most attractive and familiar to certain demographic groups, linked by common categories.
5.	Dictionary size is approx. 1 million (20 bits) without permutations.	Passphrase dictionary size is approximately 400,000 (19 bits) [82].	Although not explicitly evaluated by Davis et al. [34], some of their results appear related as follows. In Faces, a dictionary containing only female models would have 81 entries, and a single-race dictionary would have 256 entries. See text in this section for more details.

Table 4.1: Details for how our five predictive modelling steps from Section 4.2 map to some known schemes.

a dictionary composed of female models against male users, and $(.62)^4 = 15\%$ for a dictionary composed of white faces against white male users. Of course, it is possible that these dictionaries could be further optimized to show the combined effect of race and attractiveness, and that dependencies between images would lead to higher or lower success rates than what we estimate here. Their main guessing attack results (see Section 6.6) indicate that there are dependencies between the images in user passwords, but statistics regarding the characteristics of these dependencies are not currently available.

We believe that most system administrators would consider the results for all of these schemes to be of concern, or significant according to their α threshold (recall property (2), Definition 1).

4.4 How Predictive Models Can Map to New Schemes

We believe that predictive models could also be applied to new (previously unanalyzed) schemes. Here we describe how we might (and in fact do, in later chapters) apply them to instances of two other types of graphical password: pure-recall and cued-recall.

As indicated by Table 4.2, we generated dictionaries, and found them to be candidate weak password subspaces (i.e., they satisfy properties (1) and (3) of Definition 1). Note that number of bits shown is the log_2 of the number of entries in the dictionary. To show that they are weak password subspaces, we verify their significance (property 2) with user studies; further details are provided in Chapters 5 and 6.

4.5 Summary

We have presented a generalized attack method based on predictive models. We believe this method will aid in future security evaluations of knowledge-based authentication schemes. We show how predictive modelling maps to existing attack methods and consider how it might apply to two new, previously unanalyzed schemes. We pursue the application of these predictive models to two representative graphical

Step	Pure-recall Graphical Passwords	Cued-recall Graphical Passwords
1.	Users must recall and reproduce an image in a specific temporal order.	Given a background image, users must recall and select parts of the image in a specific temporal order.
	Demands pure visual recall of an image, and of temporal order.	Demands visual recall and/or search, and relocation in a particular order.
2.	Visual recall (and preference) for certain types of image, and threshold number of visual pieces of information.	Visual recall (and preference) for certain parts or places in an image, and threshold number of visual pieces of information.
3.	Image recall factors (symmetry, low number of components).	Factors for recall, attention, and preference for certain points in an image and conceptual grouping of points.
4.	For user-drawn graphical passwords, drawings that exhibit symmetry and/or a low number of components.	For click-based graphical passwords, those that contain points that are commonly preferred or draw attention, and/or sets of points that follow part of a simple temporal pattern.
5.	For a representative scheme called "Draw-A-Secret" (DAS)[72], we found a reduction from 58 to 31 bits.	For a representative scheme called PassPoints [164], we found a reduction from 43 to 18-16 bits (depending on the image).

Table 4.2: Details for how our five predictive modelling steps map to other (previously unanalyzed) schemes.

password schemes in Chapters 5 and 6. As we show in Chapters 5 and 6, these methods do produce weak password subspaces, and thus we have added two more schemes that support our general hypothesis that other knowledge-based user authentication methods are weak when based on user-chosen secrets.

Chapter 5

Application to Pure-Recall Graphical Passwords

5.1 Introduction

We apply our predictive method of Chapter 4 to a representative pure-recall graphical password scheme known as "Draw-A-Secret" (DAS; recall Section 2.4.1). We believe our methods for DAS can be applied to all *user-drawn* graphical password schemes (g_{ud}-passwords). Examples of such graphical passwords include free-form user-drawn graphical password schemes (e.g., DAS [72] and variations such as Pass-Go [146]), and schemes whereby a user might create a graphical password by dragging and dropping basic shapes.

We detail the steps to create a predictive model for g_{ud}-passwords in Section 5.2, and provide further details about its construction in Section 5.3. We describe our evaluation methodology in Section 5.4. Finally, we discuss our results in Section 5.5.

5.2 Predictive Model Creation

Here we provide further details for each step of applying our predictive modelling method to the DAS scheme. The subsections herein follow the predictive model steps outlined in Section 4.2.

5.2.1 User Tasks and Demand

We focus on user-drawn graphical passwords (g_{ud}-passwords), where a user's login task involves pure visual recall of a drawn image, and recall of the temporal order (i.e., how the image was drawn). This places a demand on the user's visual recall, and may be influenced by visual preferences.

5.2.2 Other Relevant Information

To determine relevant information about visual recall, we examine and discuss a collection of relevant cognitive studies on visual recall.

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 [88]. Various studies support this result (e.g., [76, 21, 89]). Another study [17] 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.

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 [121]. 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.

A representative overview of literature for human symmetry perception [154] notes that many objects in our environment are symmetric. There is also significant evidence [160] 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" [154].

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 objects best. This is supported by an observation by Attneave [3]: 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. In a separate study, French [53] observed that dot patterns that were symmetric were more easily remembered. Intuitively, this is no surprise – in the case of mirror symmetry, a subject need 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 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 [160]. Note that most living organisms and plants, as well as almost all forms of human construction are mirror symmetric (reflective) about a vertical axis. There is mirror symmetry in people, animals, leaves, flower petals, automobiles, planes, trains, art, buildings, tools, furniture, and religious symbols. The objects in the average office or home are another example.

Attneave's findings of shape complexity [4] also imply that people are better at recalling a low number of components. The following studies imply that values of "low" might lie between 3 and 8. Vogel and Machizawa [157] found neurophysiological evidence that the human visual short term memory is limited to 3-4 symbols. Cowan [30] reviews a large set of studies showing that, on average, the capacity for human mental storage is actually 4 (plus or minus one) "chunks" . Similar values were obtained for the number of dots recalled in grids of different sizes (recall decreased significantly after 3 or 4 dots) [64]. Alternately, French [53] found that people have optimal memory for dot patterns containing 6 to 8 dots.

5.2.3 Identify Complexity Properties

Motivated by these collective studies, we propose the following.

Conjecture 1 *Since people are more likely to recall symmetric images and patterns, and people appear to prefer and perceive mirror symmetry as having a special status, a significant subset of users are likely to choose mirror symmetric patterns as g_{ud}-passwords.*

More specifically, we propose that the mirror symmetric patterns chosen are more likely to be about vertical or horizontal axes. Findings that people are more likely to recall a low number of *components* (Definition 2) leads us to Conjecture 2.

Definition 2 ***(component).*** A *component* is a visually distinct part of an image.

For example, a component in DAS could be a drawn stroke. As another example, for a scheme wherein the user creates a password by dragging and dropping basic shapes, a component is a dragged/dropped shape.

Conjecture 2 *Since people are likely to only recall a small number (between 3 and 8) of symbols, a significant subset of users are likely to choose g_{ud}-passwords with a small number of components.*

These conjectures identify two complexity properties: degree of symmetry (more symmetry results in lower complexity), and number of components (fewer components results in lower complexity).

5.2.4 Model Classes of Probable Passwords

For g_{ud}-passwords, Conjectures 1 and 2 lead us to define a *Class D_1 password* (Definition 3) and a *Class D_2 password* (Definition 4). Here, D is used to denote that these classes apply to user-*drawn* graphical passwords (g_{ud}-passwords).

Definition 3 ***(Class D_1 password).*** A *Class D_1 password* is a g_{ud}-password that exhibits mirror symmetry about a vertical or horizontal axis in its components. Thus each component is either mirror symmetric in its own right, or is one of a pair of components that are mirror symmetric images of each other.

Definition 4 ***(Class D_2 password).*** A *Class D_2 password* is a g_{ud}-password with a small number $3 \leq c \leq 8$ of components.

The *Class D_1 password space* is composed of the set of encoded representations of Class D_1 passwords; these collectively form a graphical dictionary (i.e., a *Class D_1 graphical dictionary*). *Class D_2 password space* and *Class D_2 graphical dictionary* are defined analogously.

A clever attacker doing a guessing attack would prioritize candidate password guesses according to their probability of being chosen. We believe the graphical dictionaries D_1 and D_2 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 built from these classes as follows (see Fig. 5.1)::

1. Class $D_1 \cap$ Class D_2,
2. Class $D_2 -$ (Class $D_1 \cap$ Class D_2),
3. Class $D_1 -$ (Class $D_1 \cap$ Class D_2),
4. Full password space $-$ (Class $D_1 \cup$ Class D_2)

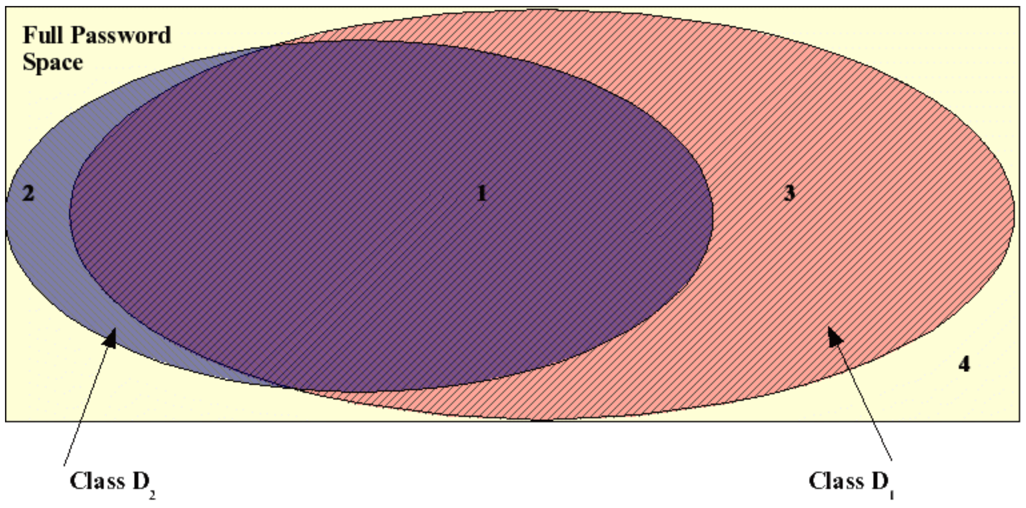


Figure 5.1: Illustrative relationship between Class D_1 and D_2.

We place Class D_2before Class D_1as it is a smaller class, which would take less time to exhaust (see Section 5.2.5). Each class might also be internally prioritized as discussed in their respective sections below.

Class D_1 Dictionaries

A logical way to prioritize the Class D_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. 5.2a), 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. 5.2b), 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. 5.2c), 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 Section 5.2.2, and our intuition suggests that global symmetry (Fig. 5.2c) is more likely than pseudo-symmetry (Fig. 5.2b), which is more likely than local symmetry (Fig. 5.2a).

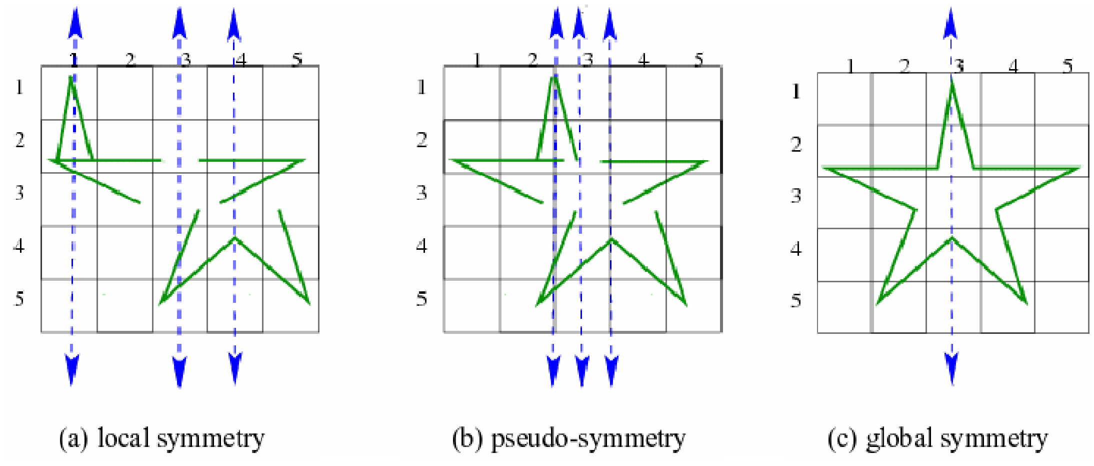


Figure 5.2: Example Class D_1 DAS passwords containing the same components, symmetric about different patterns of axes: (a) 3 different, scattered axes, (b) 3 different, nearby axes, and (c) a single central axis.

This leads us to define sub-classes of Class D_1, based on the number of axes that the image's components are symmetric about. If it turns out that global symmetry is more likely than pseudo-symmetry (and we found it is), an attacker may place passwords that are composed of components symmetric about the center-most axes at a higher priority in the graphical dictionary. Additionally, for user-drawn 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 center-most axes.

Definition 5 ***(Class*** D_{1a}***).*** *Class* D_{1a} is the subset of passwords in Class D_1 that use only the center 3 of each set of horizontal or vertical axes (e.g., the marked axes in Fig. 5.3), producing pseudo-symmetric images.

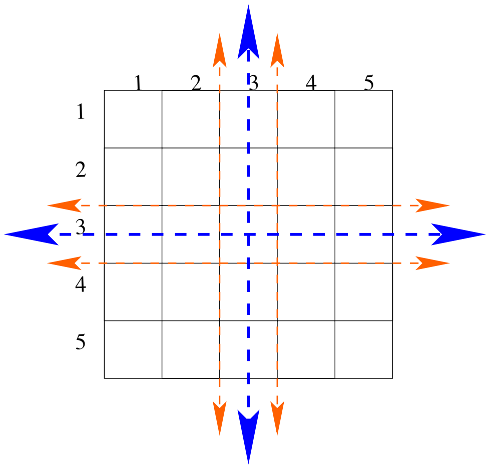


Figure 5.3: Class D_{1a} reflection axes for the DAS scheme. The thickest axes are the vertical and horizontal center axes. Adjacent axes are marked as thinner.

Definition 6 ***(Class*** D_{1b}***).*** *Class* D_{1b} is the subset of passwords in Class D_1 that use only the center of each set of horizontal or vertical axes, producing globally symmetric images.

Class D_{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. 5.4).

Class D_{1b} is a subset of Class D_{1a}, which is a subset of Class D_1. We expect that an attacker would order Class D_{1b} passwords first in a Class D_1 graphical dictionary, followed by the remaining Class D_{1a} passwords, and finally the remaining passwords in Class D_1.

Class D_2 Dictionaries

An obvious attack strategy for Class D_2 graphical dictionaries is to prioritize based on the number of components, in increasing order. A dictionary attack would thus try

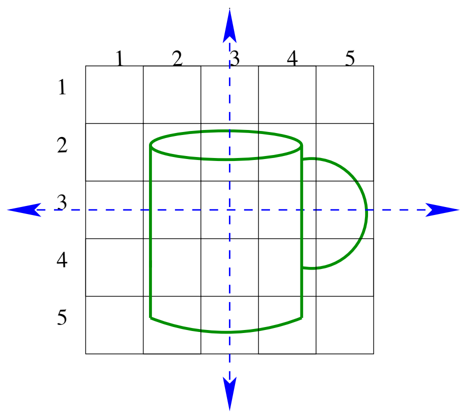


Figure 5.4: Example of a Class D_{1b} DAS drawing. One component (the handle) is symmetric about the center horizontal axis, and another (the cup) is symmetric about the center vertical axis.

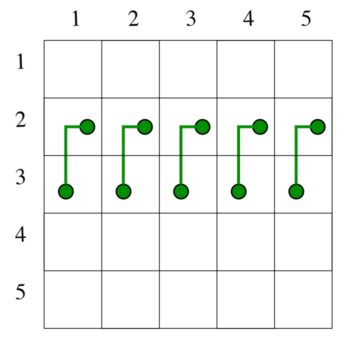
all entries with one component, then two, etc. Given the evidence to show a threshold number of 3 or 4 components is more memorable than more than 4 components (recall Section 5.2.2), we focus on those Class D_2 passwords with $c = 4$ for our model. An attacker interested in targeting a particular account might expand a dictionary to consider larger values of c.

Class D_3, D_4, and D_5 Dictionaries

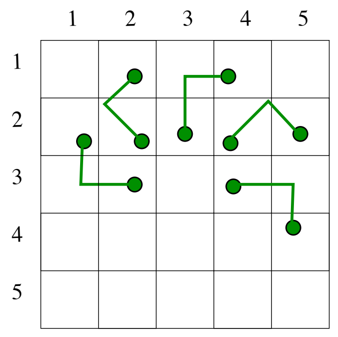
Here we mention three additional classes of graphical dictionaries. User-drawn passwords in the form of alphanumeric symbols are considered by Tao [146]; we suggest calling this *Class* D_3. We identify *Class* D_4 and *Class* D_5 based on repetitive and rotational symmetry respectively. These are common types of symmetry, although according to the previously cited cognitive studies they do not hold the same special status as mirror symmetry. We note that these classes (and possibly many others) might also be placed in a graphical dictionary. We do not pursue further details in this thesis.

5.2.5 Estimating the Size of Classes

To estimate the size of Classes D_1 and D_2, we must choose a representative scheme; we choose DAS_J. The DAS_J graphical password scheme relies on a user's ability to



(a) Repetitive symmetry



(b) Rotational symmetry

Figure 5.5: Example Class D_4 and D_5 DAS passwords.

recall their DAS_J password "exactly" (as defined by the resolution of the encoding scheme). For this analysis, we only consider passwords that are allowed within DAS_J (see Section 2.4.1).

To apply Class D_1 to DAS_J, we make further assumptions regarding the temporal order, resulting in a subset S_{D_1} as discussed below; we believe this represents the most probable subset of DAS_J Class D_1 passwords. To apply Class D_2 to DAS_J, we assume that a component is a stroke and that $c = 4$ (in Definition 4); we call this S_{D_2}. Further details concerning the construction of S_{D_1} and S_{D_2} are provided in Section 5.3.

Following the relevant parts of the attack strategy from Section 5.2.4, and using subsets of Class D_1, we expect that a DAS$_J$ graphical dictionary would be prioritized in the following order (see Fig. 5.6):

1. $S_{D_{1b}} \cap S_{D_2}$
2. $(S_{D_{1a}}\text{-}\ S_{D_{1b}}) \cap S_{D_2}$
3. $(S_{D_1}\text{-}\ S_{D_{1a}}) \cap S_{D_2}$
4. S_{D_2} - S_{D_1}
5. $S_{D_{1b}}$ - S_{D_2}
6. $S_{D_{1a}}$ - $(S_{D_2} \cup S_{D_{1b}})$
7. S_{D_1} - $(S_{D_2} \cup S_{D_{1a}})$

8. Full DAS$_J$ password space - $(S_{D_1} \cup S_{D_2})$

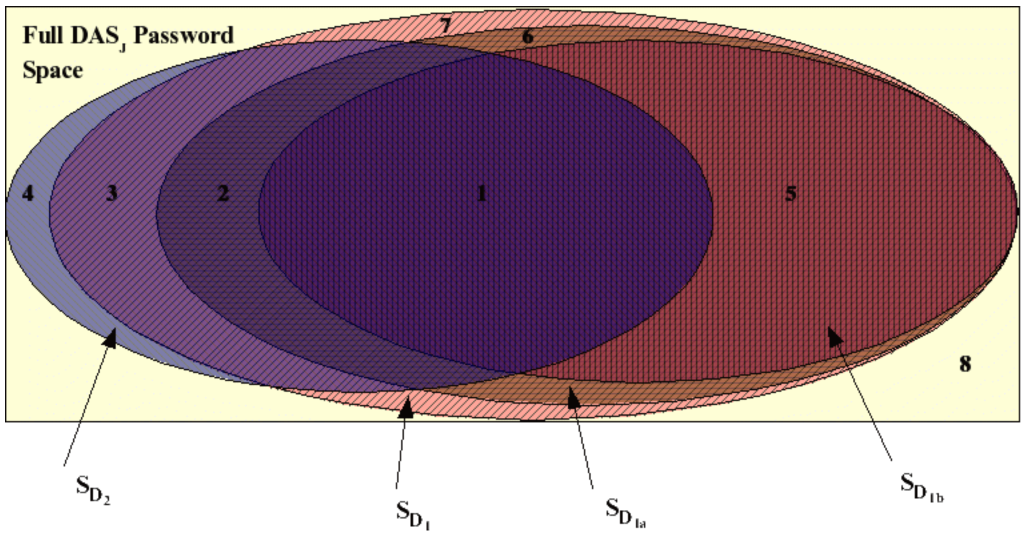


Figure 5.6: Illustrative relationship between $S_{D_{1b}}$, $S_{D_{1a}}$, S_{D_1}, and S_{D_2}.

While we expect (as mentioned in Section 5.3.1) that our Class D_1 graphical dictionary for DAS$_J$ will include most Class D_1 passwords, we recognize that some Class D_1 passwords will not be included due to our definition of S_{D_1}. However, even if our dictionary only includes as few as e.g., $\frac{1}{8}$ of the ways users would typically draw Class D_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 empirical support through user studies that our assumptions are indeed quite realistic (see Section 5.5), and thus the above error estimate appears to be conservative.

We approximate the size of password spaces $S_{D_{1b}}$, $S_{D_{1a}}$, S_{D_1}, and S_{D_2}, individually (Section 5.2.6 and Section 5.2.7) and their intersections (Section 5.2.8).

5.2.6 Approximate Size of Class D_1 Graphical Dictionaries

To estimate the size of various subsets of the password space, many (equivalent) counting methods are possible. Our definitions in Sections 5.2.4 defines what DAS$_J$ passwords are in the Class D_1 password space, and Section 5.3.1 defines which of these Class D_1 passwords belong to S_{D_1}. We give the details of counting methods for generating these results in Appendix C of a 2005 technical report [112].

Table 5.1 gives sample results for S_{D_1} and the subsets of $S_{D_{1a}}$ and $S_{D_{1b}}$ (see Definitions 12 and 13). As in Chapter 4, the number of bits shown is the log_2 of the number of entries in the dictionary. Values given are log_2(number of passwords). $S_{D_{1a}}$ and $S_{D_{1b}}$ both show an exponential reduction from the full DAS$_J$ space: $S_{D_{1b}}$ grows at an exponential rate of approximately 3.6 bits per unit increase in password length and $S_{D_{1a}}$ grows at a corresponding rate of approximately 4.0, whereas the full DAS$_J$ space and S_{D_1} grow at a corresponding rate of approximately 4.8. For example, when the maximum password length $L_{max} = 12$, the size of the full space is 57.7 bits, S_{D_1} is 57.6 bits, $S_{D_{1a}}$ is 48.1 bits, and $S_{D_{1b}}$ is 42.7 bits. The size of the full DAS$_J$ password space was cross-checked using a variation of our method (full details are given in Appendix C of the cited technical report [112]), closely matching the results given by Jermyn et al. [72].

L_{max}	1	2	3	4	5	6	7	8	9	10
Full DAS$_J$ space	4.7	9.5	14.3	19.2	24.0	28.8	33.6	38.4	43.2	48.1
S_{D_1}	4.7	9.5	14.3	19.1	23.9	28.7	33.6	38.4	43.2	48.0
$S_{D_{1a}}$	3.3	7.7	11.6	15.7	19.8	23.8	27.9	31.9	36.0	40.0
$S_{D_{1b}}$	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$_J$ space	52.9	57.7	62.5	67.3	72.2	77.0	81.8	86.6	91.4	96.2
S_{D_1}	52.8	57.6	62.4	67.2	72.0	76.8	81.7	86.5	91.3	96.1
$S_{D_{1a}}$	44.1	48.1	52.1	56.2	60.2	64.3	68.3	72.4	76.4	80.4
$S_{D_{1b}}$	39.1	42.7	46.3	49.9	53.4	57.0	60.6	64.2	67.8	71.4

Table 5.1: Bit-size of DAS$_J$ space, for total length at most L_{max} on a 5×5 grid.

Each of the three subclasses of Class D_1 passwords presented in Table 5.1 allow perceptually quite distinct classes of drawings (recall Fig. 5.2). We initially found the size of S_{D_1} to be surprisingly close to that of the full DAS$_J$ 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 possible axis (e.g., Fig. 5.2a), which includes the combinatorially large set of all permutations of dots, and lines of length two (see Section 9.1, Table 9.4 for an enumeration of this set).

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. 5.2c) are considered first. When any single axis (or two) are considered at a time to produce globally symmetric passwords, each resulting subset will never be larger than that for the two center axes (i.e., one vertical and one horizontal), 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 two center (horizontal and vertical) axes.

5.2.7 Approximate Size of Class D_2 Graphical Dictionaries

For Class D_2 graphical dictionaries, following Jermyn et al. [72] 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 5.2.

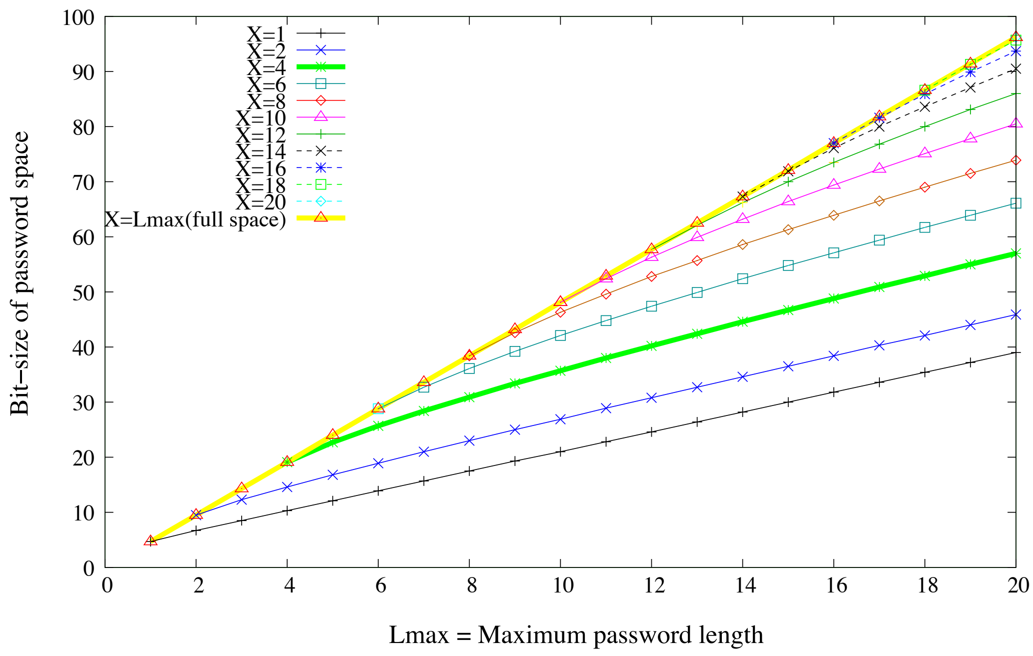


Figure 5.7: Size of DAS$_J$ password space, for passwords of at most X strokes (for a 5×5 grid and a fixed maximum password length L_{max}). S_{D_2} (for $c = 4$) is represented by the thick line where $X = 4$.

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 5.2: Bit-size of DAS$_J$ with different maximum lengths and stroke-counts as illustrated in Fig. 5.7. 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 $X = 4$.

Fig. 5.7 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$_J$ 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 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 from 1 to L_{max} accounts for at least one half of the bit-size (see the difference between the $X = 1$ line and $X = L_{max}$, when $L_{max} \geq 5$). The top line, where $X = L_{max}$, in Fig. 5.7 shows what one might expect from reading the original DAS paper [72] (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_{D_2}. We suggest that S_{D_2} is more representative of the "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$_J$ password space; the size of the password space is significantly smaller (at 40 bits) if users choose a password of length at most 12, composed of 4 or fewer strokes. The size of 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$_J$ 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. Full details for these results are provided in Tables 9.5 and 9.4.

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. Full details for these results are provided in Table 9.7. We expect that many user-chosen passwords will not contain any length-1 strokes – and note with interest that 51.5% of passwords in one large study did not contain any length-1 strokes (see Section 5.5.1).

5.2.8 Approximate Size of Combined Class D_1 and D_2 Dictionaries

Figure 5.8 shows how restricting the maximum number of strokes, while also staying within different subclasses of Class D_1 passwords, affects the size of the DAS$_J$ 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$_J$ passwords of length ≤ 12 on a 5×5 grid. Again, this 2^{58} value is the "security" measure one might originally have expected from DAS$_J$. Notice the triangle point where $X \leq 4$ – this value of 2^{40} shows the effect of number of strokes on the full DAS$_J$ password space. The $S_{D_{1b}}$ bar directly below is the intersection of $X \leq 4$ with $S_{D_{1b}}$.

5.3 Model Details

Here we discuss the details of our method to determine the size of the sets S_{D_1} and S_{D_2}. In Section 5.3.1, we make assumptions concerning the temporal order (i.e., the order of the input of cells) to map Class D_1 passwords to DAS$_J$, leading us to define the set S_{D_1} (Definition 12). Similarly, Section 5.3.2 discusses the mapping of Class D_2 passwords to DAS$_J$, leading us to define the set S_{D_2} (Definition 14).

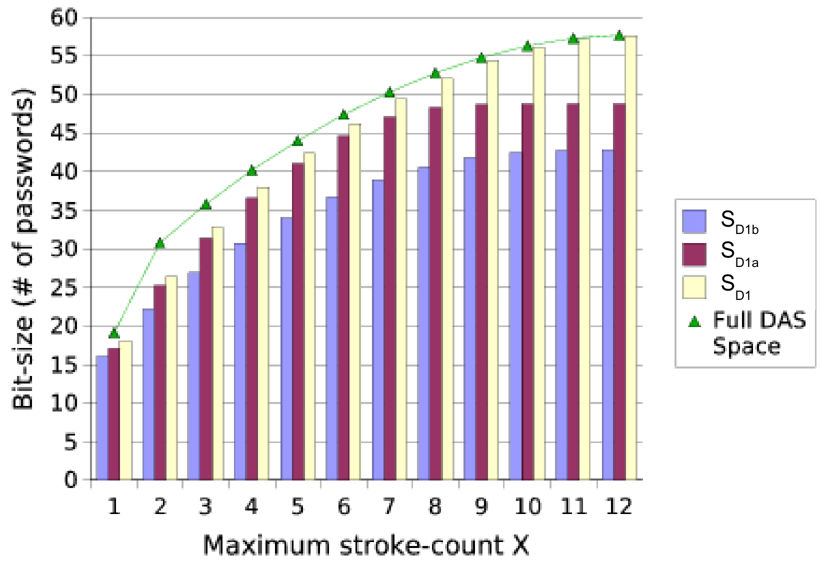


Figure 5.8: Bit-size of DAS$_J$ graphical password space. Values are given for each dictionary, with a fixed total password length at most 12, with at most X strokes on a 5×5 grid (see Table 5.3 for actual data points).

X / *Dict.*	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_{D_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
$S_{D_{1a}}$	17.6	25.2	31.4	36.6	41.0	44.6	47.1	48.3	48.7	48.8	48.8	48.8
$S_{D_{1b}}$	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 5.3: Bit-size of DAS$_J$ space as illustrated in Fig. 5.8. 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.

5.3.1 Class D_1 DAS$_J$ Graphical Dictionaries

In this section we describe how we map the visual mirror symmetry from Class D_1 to DAS$_J$ passwords. Our general approach and additional cases are described herein, leading us to define our mapping from Class D_1 to DAS$_J$.

Basic Terminology and General Approach

Our approach is to model each Class D_1 password in DAS$_J$ as a series of strokes (each representing a single component or pair of components; recall Definition 3) drawn using only symmetric strokes (Definition 8). 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 might be drawn. 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$_J$ 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. 5.9a), or are on a grid line (Fig. 5.9b). 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. 5.10a is visually symmetric when the grid is not in place, but we do not consider it part of the set of Class D_1 passwords in DAS$_J$ since its reflection axis is not on a grid line or centered in a set of cells as shown in Fig. 5.10b. 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 7 ***(symmetric area).*** The *symmetric area* (given a reflection axis a),

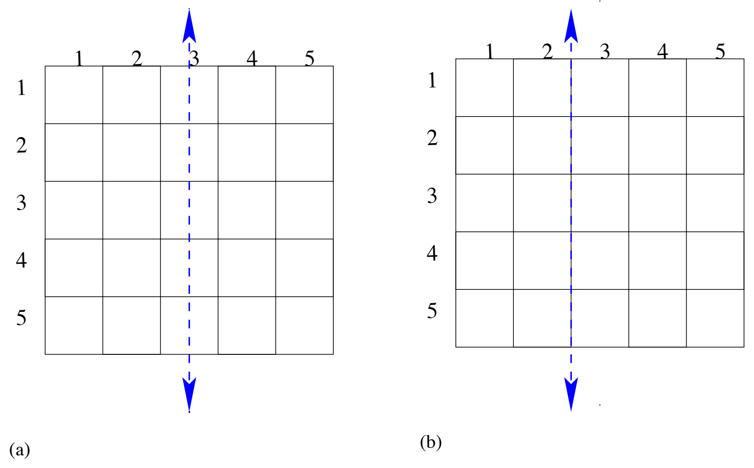


Figure 5.9: Possible axes can (a) cut a set of cells; or (b) be on a grid line between sets of cells.

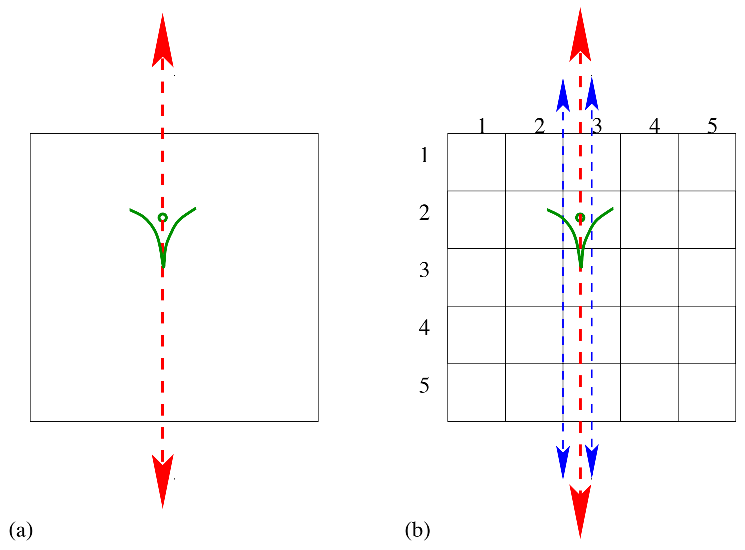


Figure 5.10: Drawing that is symmetric about a difficult to reference axis. Assuming the v is drawn before the dot, the encoding of (b) is (2,2), (3,2), (3,3), (3,2), pen-up, (3,2), pen-up. If shifted slightly right to be symmetric about the vertical axis $x = 3$, it has symmetric encoding: (3,2), (3,3), (3,2), pen-up, (3,2), pen-up.

is the area between a and the closest grid boundary parallel to a, reflected about a (see Fig. 5.11).

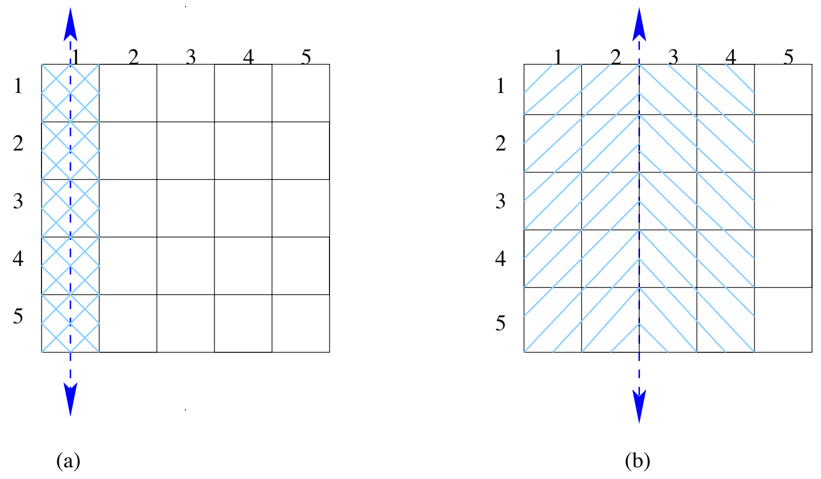


Figure 5.11: Example symmetric areas for (a) the axis $x = 1$; and (b) $x = 2.5$

One way to draw a symmetric stroke 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. 5.13a. We call the initial stroke from virtual start point s to virtual end point e that the reflection is based upon the *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$_J$.

Definition 8 ***(symmetric stroke).*** A *symmetric stroke* is a stroke (or pair of strokes) drawn such that it follows one of the *disjoint case*, *continuous case*, or *closed case* (per Definitions 9, 10, and 11). For context, see Fig. 5.12. 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 9 ***(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

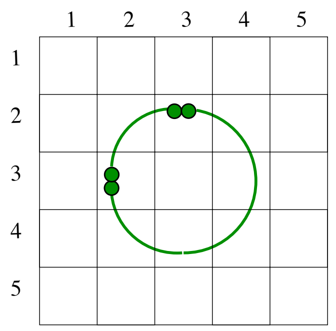


Figure 5.12: Example DAS$_J$ password in Class D_1, but *not* drawn in symmetric strokes.

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.

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 (effectively counting each way to draw the reflected stroke for each way to draw the defining stroke). The disjoint case is not the only type of symmetric stroke (see Definition 8).

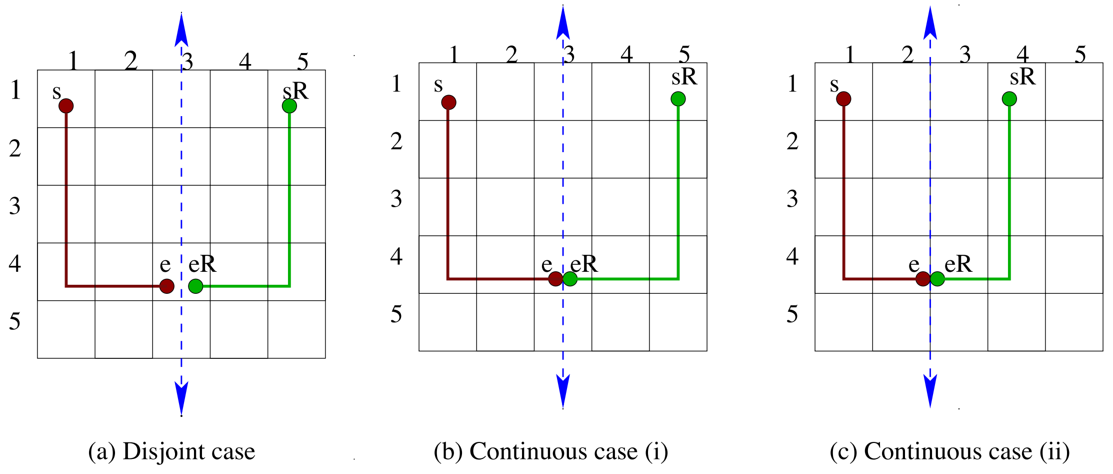


Figure 5.13: Disjoint and Continuous Cases. Symmetric strokes consist of a defining stroke (solid line from s to e) and reflected stroke (solid line from s^R to e^R). The last two, visually representing the letter 'U', show continuous cases where: (b) the axis cuts a set of cells; and (c) the axis is on a grid line.

Continuous and Closed 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 neighboring 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 closed stroke (see Definitions 9, 10, and 11).

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

Definition 10 ***(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. 5.13b 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 5.13a 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. 5.13b, 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. 5.13c), 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 a closed case when both the defining stroke's start and end points are potentially continuous (e.g., Fig. 5.14).

Definition 11 ***(closed case).*** The *closed 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 a closed shape. When a drawing is closed, the user-drawn stroke may start and end at any point in the shape (e.g., Fig. 5.14a).

As with the continuous case, the closed 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. 5.14b). When a cuts a set of cells (e.g., Fig. 5.14a), 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. Note that when the defining stroke (completely) repeats over itself is counted by this case; this is included when the defining stroke is a closed case itself. Cases where the closed case only partially repeats over itself (e.g., only a few cells) are not considered a symmetric stroke.

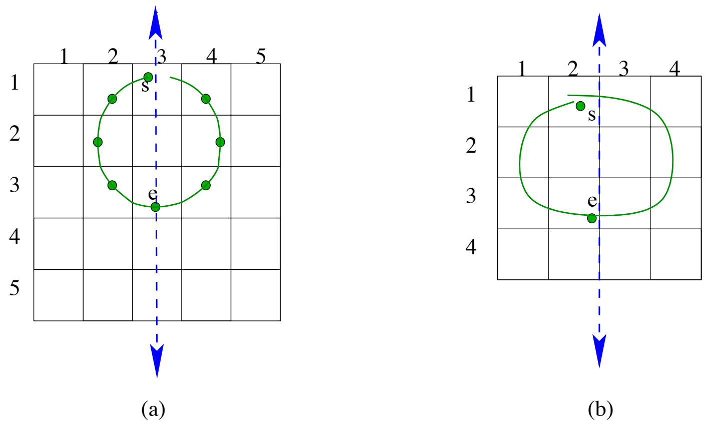


Figure 5.14: Different types of the closed case. The reflection axis in (a) cuts a set of cells, and (b) is on a grid line. Case (a) shows all possible user-drawn start/end points in the symmetric stroke modelled by s and e.

Classes S_{D_1}, $S_{D_{1a}}$, and $S_{D_{1b}}$

The definition of Class D_1 passwords takes into account only their final visual appearance. There is a one-to-many relationship between a given Class D_1 password and the number of ways it can be drawn in the DAS$_J$ scheme (which are then mapped to possibly fewer unique DAS$_J$ encodings). We believe there are some more likely *ways* that users will draw mirror symmetric components in their DAS passwords; we use S_{D_1} to denote this "more probable" subset of unique DAS$_J$ encodings of Class D_1 passwords defined as follows.

Definition 12 ***(S_{D_1}).*** S_{D_1} is the DAS$_J$-related subset of Class D_1 passwords, containing only those passwords whose components are drawn in symmetric strokes, as per Definition 8.

Definition 13 ***($S_{D_{1a}}$ and $S_{D_{1b}}$).*** $S_{D_{1a}}$ and $S_{D_{1b}}$ are subsets of S_{D_1} that belong to Class D_{1a} and Class D_{1b} respectively. More formally, $S_{D_{1a}} = S_{D_1} \cap$ Class D_{1a}; $S_{D_{1b}} = S_{D_1} \cap$ Class D_{1b}.

Informal user studies by others have shown that the temporal order has an adverse effect on a user's ability to recall a DAS password [54]. This suggests that users will choose DAS passwords with less complexity (e.g., fewer strokes). We believe that S_{D_1} captures the easiest (and thus most likely to be chosen) ways to draw Class D_1 passwords, although not all possible ways. The details of how we enumerated the DAS$_J$ Class D_1 space (or equivalently, graphical dictionaries) can be found in our 2005 technical report [112].

5.3.2 Class D_2 DAS$_J$ Graphical Dictionaries

In Definition 4, we define a Class D_2 password to have at most c components; for our DAS$_J$ analysis, we use $c = 4$, since this value has support from at least 2 cognitive studies (recall Section 5.2.2). 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$_J$, we characterize a Class D_2 password by the number of composite strokes.

This leads us to define S_{D_2} below. We quantify the relationship between the DAS$_J$ password space and the stroke-count in Section 5.2.7.

Definition 14 *(S_{D_2}).* S_{D_2} is the DAS$_J$-related subset of Class D_2 passwords, containing only those passwords having a stroke-count of 4 or less.

Our general approach is to determine how many DAS$_J$ 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. [72]. We modify their function *P(L,G)* 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. The resulting function is provided in (5.1).

$$P(L,G,X) = \begin{cases} 0 & \text{if } X = 0 \text{ and } L > 0 \\ 1 & \text{if } X \geq 0 \text{ and } L = 0 \\ \sum_{\ell=1}^{L} N(\ell) \cdot P(L-\ell, G, X-1) & \text{otherwise} \end{cases} \tag{5.1}$$

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)* (see [72]), which gives the number of *strokes* of length ℓ. We use (5.1) to determine the results in Section 5.2.7.

5.4 Evaluation Methodology

Here we discuss our method to evaluate which of our dictionaries qualify as weak password subspaces for the DAS scheme. First, we must define thresholds for how much risk we are willing to accept (i.e., α in Definition 1), and the resources of an attacker we expect protection from (i.e., t_e in Definition 1). Of course, these thresholds are not fixed as they are dependent upon the amount of risk that a system administrator (or user) is willing to accept, and the type of adversary that one would like protection against.

Although the available resources differ for each attacker and his/her motivation, we prefer to evaluate our methods using a conservative measure of how many resources a somewhat motivated attacker would likely be able to obtain, so as to not overestimate the success of our attack dictionaries. We assume that the attacker will use personal resources and give up after two weeks. Although these assumptions do not model all motivated attacker's patience and resources, it is equally valid as any other fixed point for the sake of discussion (to model an attacker who is much more motivated, a much larger dictionary could be exhausted). It is important to keep in mind that an attacker could easily have ten-thousand times more resources than what we assume herein if they are sufficiently motivated to rent a botnet.

By 2008, it is likely that an average attacker will have access to at least 8 processors (if you assume that the attacker and a friend will both own machines with the new quad-core processors [28]). We assume that MD5 is applied before storing the target passwords, and thus that checking a guess requires a single MD5 hash operation on a block of ≤ 512 bits. Using an MD5 performance result of 3.66 cycles/byte for a *Pentium 3* 800MHz processor [104] (scaled to 2.66GHz), and a 512 bit block size, approximately 1.22×10^7 MD5 hash operations can be performed per second per processor. We base our value of t_e on the approximate number of MD5 hashes that can be performed by an attacker with 2 quad-cores (at 2.66GHz per processor) for 2 weeks. This results in $t_e = 2^{46.75}$.

To verify Class D_1 and Class D_2's hypothesized status as weak password subspaces, we determine the number of g_{ud}-passwords that fall within them from two user studies [105, 146]. We declare a candidate subspace as weak if it contains at least the same percentage most recently found to fall to a basic text dictionary attack (5%) [82]. Thus, our $\alpha = .05$, meaning that out of a system of 20 users, we would expect one password to be susceptible to the dictionary attack.

We define the success of our predictive model's application to pure-recall graphical passwords by whether it produces candidate weak password subspaces (such that they contain fewer than $t_e = 2^{46.75}$ entries), and that through user studies, these candidate password subspaces are shown to facilitate correctly guessing at least 5% ($\alpha = 0.05$) of passwords.

5.5 Results for Applying Class D_1 and D_2 to DAS

Using our predictive modelling method of Section 4.2, we found two weak password subspaces that we called Class D_{1b} and Class D_2. We called $S_{D_{1b}}$ and S_{D_2} our application of these classes to DAS$_J$. We found these subspaces to be both small and generable (satisfying properties 1 and 3 of Definition 1); using the convenient parameters of a 5×5 grid and g$_{ud}$-passwords of at most length 12, we found $S_{D_{1b}} \subset S_{D_1}$ to be 42 bits (cf. Table 5.1), S_{D_2} to be 40 bits (cf. Figure 5.7), and $S_{D_{1b}} \cap S_{D_2}$ to be 31 bits (cf. Figure 5.8). Two separate studies supported that they are all significant (satisfying property 2). Nali et al. [105] found that 30% of g$_{ud}$-passwords would belong to Class D_{1b}, and 80% of g$_{ud}$-passwords would belong to Class D_2. Tao [146] found that 40% of g$_{ud}$-passwords belonged to $S_{D_{1b}}$ and 72% of g$_{ud}$-passwords belonged to S_{D_2}. These studies are described in further detail in Section 5.5.1, the limitations for which are described in Section 5.5.2.

5.5.1 Supporting Evidence for S_{D_1} and S_{D_2}

In this section, we discuss one reported study supporting the hypothesis that Classes D_{1b} and D_2 are indeed weak password subspaces, and thus that the proposed graphical dictionary attack strategy is likely to perform quite well from an attacker's perspective, at least for the implementation discussed. We also discuss two other studies that we do not rely upon, and although they are not necessary for our conclusions, they are additional data points of interest. We discuss limitations of these three user studies in Section 5.5.2.

A user study of 167 students was performed by Tao [146] on an implementation of a variation of DAS called Pass-Go (see Section 3.2.2). We discuss how Pass-Go maps to DAS$_J$ in Section 3.2.2. University students used the system to access their grades for one course over a 4-month semester. The results were analyzed using S_{D_1} and S_{D_2} as proposed herein, and a third class that Tao defines as a subset of alphanumeric characters and well-known symbols, which we call D_3 (recall Section 5.2.4). The study found that 40% of users chose g$_{ud}$-passwords that fell in our $S_{D_{1b}}$, and when no stroke-count restrictions were applied, 72% of users chose g$_{ud}$-passwords that fell in our S_{D_2}. Other findings of interest include that password creation was increasingly difficult (in terms of the password creation success rate) when more restrictive stroke-count policies were applied (the most restrictive of which was to have at least 4 strokes), and that 41% of users created passwords that fell into our class $S_{D_{1a}}$ (only 1% more than the 40% already in $S_{D_{1b}}$), implying that global symmetry is significantly more probable than pseudo-symmetry (recall Section 5.2.4). This study also supports our expectation that many passwords will have no strokes of length 1; 51.5% of users had passwords with no length-1 strokes. Tao's subset of alphanumeric characters and symbols (under specified temporal orders and lengths) was 35.9 bits in size, and 19% of users chose passwords from this subset.

Of related interest, is another study that Tao reported; although we do not rely on this study for support, we believe it is a related data point of interest. Tao posted a web site where anyone could create and practice Pass-Go passwords, and found that the results for this site (although yielding a study of smaller size and less controlled) were in line with that of the longitudinal study of 167 students. Of the 57 practice passwords created on this site, 37.5% fell into $S_{D_{1b}}$ and 67% fell into S_{D_2} [146]. Also, no additional passwords fell into $S_{D_{1a}}$ beyond those in $S_{D_{1b}}$, again showing a preference for global symmetry.

Tao's results are similar to Nali and Thorpe's [105] informal paper-based user study of 16 students. Although we do not rely on this study for support, it remains another interesting data point. This study found that 45% of users chose symmetric passwords, two-thirds of which were mirror symmetric (and thus would fall into class $S_{D_{1b}}$). It also found that 80% of users chose passwords composed of 1-3 strokes, and with a definite tendency towards centering the password on the grid provided (56% were perfectly centered; an additional 30% were centered about about a set of cells on either side of the center grid lines).

5.5.2 Limitations of User Studies

It appears that the best (most reliable, and perhaps only) method to determine user choice and behaviour for a given system is to deploy the system under investigation in the field, and study the results from a variety of populations. However, as for most user studies, those discussed in Section 5.5.1 only apply to a particular deployment environment, for a single user population. Thus, we discuss potential limitations of these studies when applying their results to other deployment environments and/or user populations.

The Pass-Go system of Tao [146] was used to protect course materials, including assignment and lab marks. Tao reports, based on responses from a post-study questionnaire, that 80% of users did not perceive this information to be sensitive and stated that for this reason, they picked passwords that were easy or simple. Thus, it is possible that if the system had protected information that was perceived to be sensitive, some of these users might have created passwords they perceived to be "more complex".[1] Without a secondary study of the same user population, it is unknown whether these users would have created different passwords, and if so, in what way they would differ (e.g., more strokes, less symmetry, longer passwords, and by what degree). How users' choice in user-drawn graphical passwords is affected by their perception of the need for heightened security remains unanswered. Also, there are differences in implementation detail between DAS and Pass-Go, such as the use of indicators on the grid (in the form of stars and shaded cells) to help users navigate. We note that these indicators were placed in a globally symmetric pattern on the grid, which may have encouraged symmetric drawings. However, the star indicators alone did not appear to increase the number of symmetric passwords; in Tao's web-based practice study, which did not use star indicators, the number of passwords starting at either stars or corners was reduced (from 68% in the longitudinal study to 39% in the web-based practice study), but the percentage of passwords in S_{D_1} remained approximately the same.

The study by Nali and Thorpe [105] examined initial user choice on a single occasion, and as such did not account for the effect of password resets over time. Thus, it is possible that if users had to remember their passwords, they would have created (or reset) them differently. It is unknown how user-drawn graphical passwords change in complexity over password resets. Also, as this study was performed on paper, it may have resulted in different choices relating to the input device; for example, in

[1]However, Riley's survey [134] on password habits indicates that 60% of users did not change their text password complexity according to the security of the site; thus we might expect a similar proportion to leave their graphical passwords unchanged for higher security sites.

mouse-based systems, users might be less likely to choose passwords with fine detail due to increased difficulty to draw such passwords with a mouse. The differences between user-drawn graphical passwords created on different input devices remains a question worth independent study. Finally, the sample size of this study was small (16 users), and thus may not have been enough to obtain a representative distribution of user choice, although we note the percentages of users choosing passwords in S_{D_1} and S_{D_2} are quite similar to those in Tao's study.

5.6 Conclusion

Our predictive models identified two separate weak password subspaces for g_{ud}-passwords: Class D_{1b} and Class D_2, of size 42 and 40 bits respectively under convenient parameter choices. These two subspaces were found to contain 40% and 72% of g_{ud}-passwords respectively from a large scale study [146]; of course, the results of this study are subject to its limitations as discussed in Section 5.5.2.

Weak password subspaces naturally define a set of password rules, based on the properties that allow them to be generable. We discuss the specific password rules that should be applied based on our results for g_{ud}-passwords in Section 5.6.1. Also, enhancements of the graphical password implementation can increase security. We discuss these in Section 5.6.2.

5.6.1 Password Rules

Our work quantitatively supports earlier suggestions [72] that in order for g_{ud}-passwords to be secure against off-line dictionary attacks, password rules and proactive checking should be employed. Given our knowledge to date of weak password subspaces for DAS_J, we suggest the following as an initial set of g_{ud}-passwords rules. For other variations on g_{ud}-passwords, stroke-count could be generalized to the smallest user-created units whose input order matters. We expect this list will grow over time, as more hypotheses of password complexity properties are developed and observed to hold in practice.

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 D_{1b}).

3. Require at least one stroke of length 1.

Under the proposed password rules, users must be able to recall asymmetric passwords with a larger number of short strokes. Usability might suffer with password rules (in both memorability and repeatability), which might in turn lead to other exploitable patterns in user choice. It is likely that mnemonic strategies would need to be developed to aid memorability. Pass-phrases, i.e., sentences that help users recall a password, are a text-based password mnemonic. An analogous mnemonic proposed for graphical passwords is to create a story based on the picture(s) [34]. Any mnemonic strategy should be analyzed for new patterns it may encourage in user choice; Kuo et al. [82] found that a 400,000 entry dictionary guessed 4% of text passwords created using pass-phrases.

5.6.2 Implementation Enhancements

Implementation enhancements for text-based passwords can also be applied to graphical passwords. DAS$_J$ passwords can be stored using a one-way hash, so could benefit from hashing algorithms with an adaptable cost (e.g., see [128]) and/or that use *password stretching* or repeated hashing of passwords (e.g., see [59]) to increase the computational cost of guessing attacks. Also, the user can specify how long each guess should take by using Boyen's halting puzzles [18], although a usable integration of this concept into the system may prove more of a challenge (recall discussion in Section 2.3.2). "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 the computational cost of guessing attacks against a set of users.

For DAS, minor implementation enhancements can increase security (albeit typically at some cost in usability), e.g., additional user-selected characteristics of drawings such as color, backgrounds, and textures. Dunphy et al. [44] found that the effect of using different backgrounds behind the drawing grid reduces the number of passwords that would fall to Class D_2 attacks (since this apparently increases both the stroke-count and password length), but that over 40% of users would still choose Class D_{1b} passwords (the effect of which was not quantified).

The g_{ud}-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. Unfortunately, we found that increasing the grid size from 5×5 to 10×10 only provides 5-20 extra bits for convenient parameter choices (see Figure 5.15, and Tables 9.1-9.3 for the complete data set).

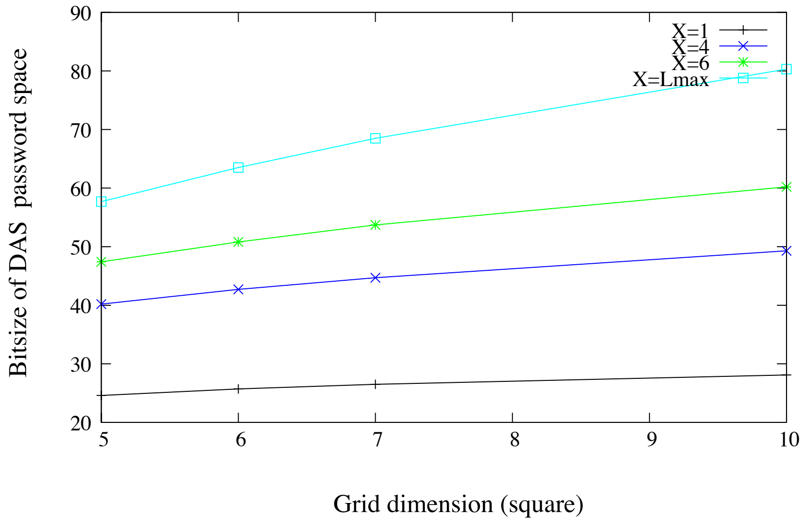


Figure 5.15: The effect of grid size on bit-size of DAS_J password space for a maximum password length of 12 and selected values of X.

Another method that might increase security is to use a form of "zooming in", originally proposed for click-based schemes [14, 71]. One grid-based analogy to zooming in, *grid selection* [150], involves a user selecting a smaller "drawing grid" (on which the password is later drawn) from a larger "selection grid". However, to have any confidence in this method, a (somewhat elaborate) user study involving a variety of implementation designs would be necessary to determine how much additional entropy it might add, at what cost to usability.

Chapter 6

Application to Cued-Recall Graphical Passwords

6.1 Introduction

To further demonstrate our predictive modelling method, we apply it to a representative cued-recall graphical password scheme known as “PassPoints” (recall Section 2.4.2). We believe our results for PassPoints [151] can be applied to other click-based graphical passwords (g_{cb}-passwords), which require the user task of logging in by clicking a sequence of points on a single background image. Other examples of g_{cb}-passwords include Picture Password [71], Blonder’s scheme [15], V-Go [120], and visKey [139].

We detail the steps from Section 4.2 to create a predictive model for click-based schemes in Section 6.2, and provide further details about its construction in Section 6.3. We describe our evaluation methodology in Section 6.4 and our results in Section 6.5. We describe another (non-predictive) attack strategy against PassPoints that achieved even better results in Section 6.6. Related results about hot-spots are discussed in Section 6.7, although these results are not required to understand the attacks presented herein. Concluding remarks are made in Section 6.8.

6.2 Predictive Model Creation

Here we provide further details for each step of applying our predictive modelling method to the PassPoints scheme. The subsections herein follow the predictive model steps outlined in Section 4.2.

6.2.1 User Tasks and Demand

The user task in a cued-recall graphical password scheme is to: (1) examine a presented background picture, (2) search for and locate the points on the image to click, and (3) enter the set of click points in the same order as during password creation. These tasks demand visual recall, or searching for and relocating a set of points on the background image, which may be influenced by preferences for certain parts of the image. These tasks also require remembering the input order of the set of points.

6.2.2 Other Relevant Information

Here we discuss relevant information relating to the user tasks and demand identified in Section 6.2.1.

Relevant information on visual recall is any research on what properties of *pieces* of images make them more memorable than others. The research on visual recall reviewed in Section 5.2.2 suggests that due to the concreteness continuum, the places in an image representing objects should be the easiest places to recall.

Cueing a user with a background image introduces other factors; there may be parts of the image that people prefer. If we assume that users prefer (and will choose) parts of an image that "stand out", then research on human visual attention may be relevant. There are two different categories of visual attention: bottom-up and top-down. *Bottom-up* visual attention captures how attention is instinctively drawn to the parts of a scene or image that are conspicuous. It is what draws us to look at the unexpected or different parts of a scene, prioritizing them from the other consistent parts. For example, if an image contains a large number of objects that are blue, and only one is yellow, human attention will instinctively focus on the yellow object. *Top-down* visual attention is task-dependent and based on cognitive, volitional control. With a priori knowledge about what objects to look for, our attention is brought to the parts of the scene containing those objects. For example, if a user decides that people with dark hair are of interest for some reason, the user's attention would shift between each dark-haired person in the image.

Indeed, there is existing research in this area, and existing computational models of "bottom-up" visual attention [67] have been shown to correlate with human visual attention in images containing little "top-down" information [116]. Current top-down visual attention models are created by training with labelled images as discussed by Olivia et al. [110]. Since many assumptions are required about what top-down processes are used in this context to train such a visual attention model, we also discuss another method of collecting information about the places that users prefer: "human computation" [159], whereby other people are asked to choose click-points in a separate context (e.g., in a game, lab, or other simplified setting).

Beyond the *places* users choose for click-points, visual recall research can tell us something about how many pieces of visual information people can recall. In particular, it appears that people are better at recalling fewer pieces of visual information (with a more dramatic decline after three pieces) [87], and tend to group those pieces together in "chunks" if possible [30]. If users somehow relate their points to one another (in essence, creating an algorithm to reproduce the password), they reduce the number of distinct pieces of information to be recalled. Example relationships between points include patterns in the objects clicked (e.g., colour and/or shape) and the order in which they are clicked.

6.2.3 Identify Complexity Properties

Motivated by the research discussed in Section 6.2.2, we propose the following:

Conjecture 3 *Since users must relocate a point with precision, and people are more likely to recall objects, a significant subset of users are likely to choose g_{cb}-passwords composed of points that represent small objects (or small, relocatable parts of an object).*

Conjecture 4 *Since people are likely to prefer some areas of an image more than others, the aggregate effect will likely be that a significant subset of users will choose g_{cb}-passwords composed of points that are more commonly preferred.*

Conjecture 5 *Points that are more commonly preferred can be defined by image processing tools (e.g., models of visual attention), or points that other groups of people prefer (as defined by popular points selected in a human-computed data set).*

Another factor that could influence the complexity of a password is the relationship between click-points that compose a password. Findings that people are better at recalling fewer pieces of visual information [87] motivate us to propose Conjecture 6.

Conjecture 6 *Since people find it easier to recall fewer pieces of information, a significant subset of users are likely to choose sets of points that are related by visual similarity (e.g., in their shape, colour, or intensity), or follow a simple click-order pattern (e.g., left to right).*

These conjectures lead us to identify three complexity properties: the size of the object underlying the click-point (the closer the object size is to the T-region size, the lower the complexity), the degree of a click-point's preference (higher preference results in lower complexity), and the degree of relation between points (a stronger relationship results in lower complexity).

6.2.4 Model Classes of Probable Passwords

For g$_{cb}$-passwords, Conjectures 3, 4, 5, and 6 lead us to define a *Class C_1 password* (Definition 15), *Class C_2 password* (Definition 16), and *Class C_3 password* (Definition 17). Here, C is used to denote that these classes apply to *click*-based graphical passwords (g$_{cb}$-passwords). Of course, the passwords that fall within these classes will differ depending on the background image, and the particular application of these classes to a g$_{cb}$-password scheme. We discuss *Class C_4 passwords* in Section 6.2.4.

Definition 15 ***(Class C_1 password).*** A *Class C_1 password* is a g$_{cb}$-password composed of small parts of the image (i.e., no larger than the system's T-region) that users prefer, as defined by an image-processing based model of user choice.

Definition 16 ***(Class C_2 password).*** A *Class C_2 password* is a g$_{cb}$-password composed of small parts of the image (i.e., no larger than the system's T-region) that users prefer, as defined by popular points observed in a human-computed data set.

Definition 17 ***(Class C_3 password).*** A *Class C_3 password* is a g$_{cb}$-password composed of a sequence of points that follow a simple click-order pattern.

For example, a password that is composed of click-points that sweep from left to right would be considered a Class C_3 password. The size of the parts of the image underlying the points of a Class C_3 password are irrelevant; here we only care about the click-order pattern of the points, rather than what is underlying the points.

The Class C_1 password space is composed of the set of encoded representations of Class C_1 passwords; these form a graphical dictionary (i.e., a Class C_1 graphical dictionary). Class C_2 and Class C_3 password spaces and Class C_2 and Class C_3 graphical dictionaries are defined analogously. We refer to a Class C_2 dictionary as being *human-seeded* (i.e., seeded with a human-computed data set).

As in Chapter 5, we believe these classes would be the basis of a prioritized ordering of password guesses. We suggest that a clever attacker may prioritize a multi-class graphical dictionary according to g_{cb}-passwords with more probable click-order patterns, and belonging to our classes as follows (see Figure 6.1):

1. Class $C_2 \cap$ Class C_3,
2. Class $C_2 -$ (Class $C_2 \cap$ Class C_3),
3. (Class $C_1 -$ Class C_2) $\cap$ Class C_3,
4. Class $C_3 -$ (Class $C_3 \cap$ (Class $C_2 \cup$ Class C_1)),
5. Class $C_1 -$ (Class $C_1 \cap$ (Class $C_2 \cup$ Class C_3)),
6. Full password space $-$ (Class $C_1 \cup$ Class $C_2 \cup$ Class C_3)

This ordering assumes that image processing based methods will not capture users preferences as well as a human-computed data set. If an image processing based method is found to accurately describe user preferences across many image types, this ordering might be revisited. Each class might also be internally prioritized as discussed in their respective sections, below.

Class C_1 Dictionaries

For the Class C_1 dictionary, we do not consider click-order patterns or dependencies between points, but assume that each click-point is independent of the others, and that the probability of a given password is defined only by the probability of each of its points. One way to create a Class C_1 dictionary is to use a model of visual attention,

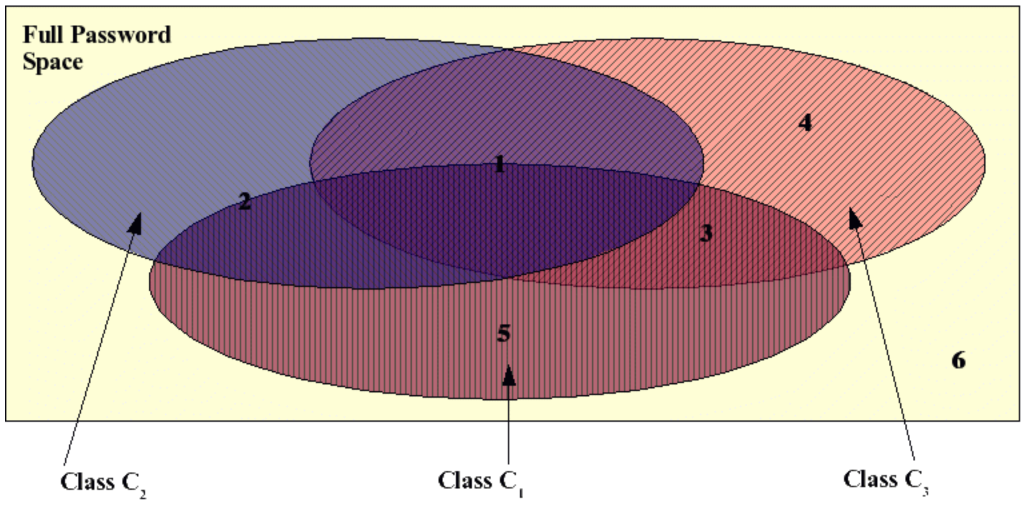


Figure 6.1: Illustrative relationship between Class C_1, C_2, and C_3.

and prioritize it by assuming that more salient[1] image locations (or combinations thereof) are more likely to belong to a password. We use this general method to create a Class C_1 dictionary; we call our particular implementation S_{C_1}.

Our image processing approach creates an ordering of points, from most to least probable. We implement a variation of Itti et al.'s bottom-up model of visual attention [67], such that it ranks a list of pixels according to their saliency, i.e., pixels that stand out the most from their background (in terms of colour, intensity, and orientation). We then create a bitmask of this list from the output of Harris corner detection [60]; the goal of this step is to isolate only those points that stand out *and* can be located and clicked again with accuracy. The resulting corners are then ranked according to their saliency as defined by the model of visual attention. For more details on this method, see Section 6.3.

Class C_2 Dictionaries

As with the Class C_1 dictionary, we do not consider click-order patterns or dependencies between points, but assume that each click-point is independent of the others, and that the probability of a given password is defined only by the probability of each

[1]A *salient* point is one that visually "stands out" from its surroundings.

of its points. A logical way to prioritize the Class C_2 dictionary is to assume that more commonly preferred image locations (or combinations thereof) are more likely to belong to a password.

The observed frequencies of click-points from a human-computed data set creates a ranking of points from most to least probable. We obtain our human-computed data from the click-point data collected from a short-term in-lab user study. This data is reduced into *clusters* (recall Section 2.4.2) according to a *clustering algorithm* (see Section 6.3.2). The resulting clusters are then ranked according to their size (the number of click-points, from the lab study data, that lie within the cluster). We call our particular Class C_2 dictionaries generated as described above S_{C_2}. This method represents a viable attack strategy, as human-computed data could be obtained through a number of means (e.g., friends, paying as small number of people, games, or restricting access to a popular website until the computation task is complete). Obtaining human-computed data does have the disadvantage of requiring more time than an image processing method as used for Class C_1. For more details on this method, see Section 6.3.

Class C_3 Dictionaries

Class C_3 dictionaries can consider any click-order pattern that a user might use to relate his/her click points to one another. For example, this might include general sweeping directions from left to right or right to left. We consider a small set of such click-order patterns as subclasses of Class C_3: DIAG (which considers g_{cb}-passwords composed of click points in a consistent horizontal *and* vertical direction; note this includes straight lines as in Figure 6.2), HOR (which considers g_{cb}-passwords composed of click points in a consistent horizontal direction), VER (which considers g_{cb}-passwords composed of click points in a consistent vertical direction), CWCCW (which considers g_{cb}-passwords composed of click points in a consistent clockwise or counter-clockwise direction). We prefix each of these subclasses with S_{C_3}- to denote our particular implementation of these click-order patterns. More details on how we generate each of these click-order patterns are given in Section 6.3.

Prior to examining a large population of real g_{cb}-passwords we had no reason to believe that any of these click-order patterns will be more likely than another, and it would seem reasonable to assume that such patterns will be influenced by patterns in the image itself, such as the horizontal rows of cars shown in Figure 6.2. Thus to prioritize a Class C_3 dictionary, we suggest ordering the Class C_3 subclasses based on their size (until there is reason to believe that a certain click-order pattern would be more likely).

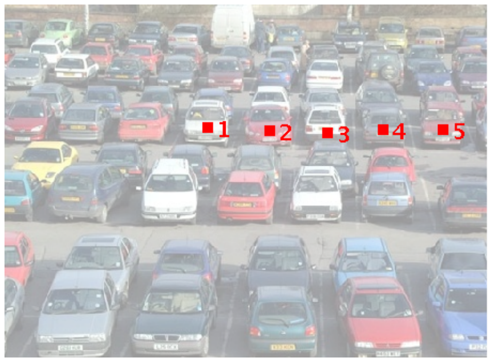


Figure 6.2: Example Class C_3-DIAG password.

Class C_4 Dictionaries

Here we mention an additional class of graphical dictionaries: Class C_4. Class C_4 captures the concept of g_{cb}-passwords composed of a sequence of points that are "visually similar". This can be considered another way of logically grouping the click-points in a password, so that less information must be recalled. Example measures of similarity between points include the shape of the objects (e.g., all license plates in the *cars* image shown in Figure 6.2), or the colour of the objects (e.g., all red objects). Thus, a Class C_4 dictionary might be defined based on a particular image processing measure (e.g., image segmentation [46] with similarity matching of segment shapes, or by filtering out all but one colour at a time and using only the remaining points).

Although we believe that Class C_4 (and possibly other classes) could be used as part of a dictionary ordering (or as an optimization within Class C_1, Class C_2, or Class C_3 passwords), we do not pursue further details in this thesis.

6.2.5 Estimating the Size of Classes

To estimate the size of our classes, we choose PassPoints as a representative scheme. In general, we use the term S_{C_X} to denote our particular implementation of Class C_X to PassPoints. For this analysis, we assume parameters similar to the original work on PassPoints [164]: a 451×331 image size, and an error tolerance of $t = 9$, and thus a T-region of 19×19.[2]

We use the term S_{C_1} to denote our implementation of Class C_1 to PassPoints, which is based on the ordering we create with our set of image processing tools detailed in Section 6.3.1. There are other ways to apply Class C_1 to PassPoints; one other example is the method recently used by Dirik et al. [40] (recall Section 3.3.4), which is a distinct implementation from our S_{C_1}. We use the term S_{C_2} to denote our implementation of Class C_2 to PassPoints. S_{C_2} is based on the ordering we create with a human-computed approach using a data set collected from an in-lab study. Further details concerning the construction of S_{C_1}, S_{C_2}, and S_{C_3} are provided in Section 6.3.

Following the relevant parts of the attack strategy from Section 6.2.4, we expect that a PassPoints graphical dictionary would be prioritized in the following order:

1. $S_{C_2} \cap S_{C_3}$
2. S_{C_2} - $(S_{C_2} \cap S_{C_3})$
3. $(S_{C_1}$ - $S_{C_2}) \cap S_{C_3}$
4. S_{C_3} - $(S_{C_3} \cap (S_{C_1} \cup S_{C_2}))$
5. S_{C_1} - $(S_{C_1} \cap (S_{C_2} \cup S_{C_3}))$
6. Full password space - $(S_{C_1} \cup S_{C_2} \cup S_{C_3})$.

The relationship between these sets is analogous to the relationship between their parent classes in Figure 6.1. We approximate the size of password spaces S_{C_1}, S_{C_2}, and S_{C_3} in Sections 6.2.6, 6.2.7, and 6.2.8. Since S_{C_1} is was not found to be a weak password space on all background images due to the number of passwords it correctly

[2]This was chosen to approximate the 20×20 tolerance square used by Weidenbeck et al. [164].

guesses (see Section 6.5.2), we only consider the intersection of S_{C_2} and S_{C_3} in Section 6.2.9.

6.2.6 Approximate Size of Class C_1 Graphical Dictionaries

The size of S_{C_1} is defined by the lowest-ranked point that it includes. We include the top one-third of all points found by our image processing tools (see Section 6.3.1 for details), which gives us an alphabet size of 138. All 5-permutations of these top 138 points provides a 35 bit S_{C_1} graphical dictionary. We also create another smaller S_{C_1} dictionary by only including the top one-quarter of all points, giving an alphabet size of 103, and a graphical dictionary of size 33 bits. As in Chapter 4, the number of bits shown is the log_2 of the number of entries in the dictionary.

6.2.7 Approximate Size of Class C_2 Graphical Dictionaries

The size of S_{C_2} is defined by the number of points available for use from the human-computed data set, which can be further pruned by only considering those clusters of at least a certain size. We use all points from our human-computed data set, since our data set is small enough that pruning would produce a very small S_{C_2} (particularly for *cars* where most clusters in the data set were of size one). Table 6.1 gives our results for S_{C_2} for the two images we focus our analysis upon: *cars* and *pool* (see Section 6.5, Figures 6.4a and 6.4b respectively). Our reasons for selecting these images are discussed in Section 6.5; in short, they were selected to represent apparently different levels of security. We give the name P^{V_u} to the dictionary composed of all 5-permutations of the *clusters* calculated (using the clustering algorithm described in Section 6.3.2) from the human-computed click-point data collected from u users. Here P denotes a set of permutations, V denotes that clusters are used, and its subscript denotes the number of users in the sample. See Section 6.3.2 for details of how this data was collected and S_{C_2} generated. With the goal of determining the number of samples of human-computed data required to generate an effective dictionary, we examine the average number of clusters, and thus dictionary size, of ten randomly selected subsets of 25, 20, and 15 people from the total data sample collected.

Although we do not formally consider it part of S_{C_2}, we also show results for

P^{R_u}, where R denotes that only the raw click-points collected are used (i.e., the alphabet is not reduced by using the clustering algorithm). The reason for including P^{R_u} is to demonstrate the clustering effect for smaller user samples, and the effect of using the clustering algorithm on both the size and significance of the resulting dictionaries (recall Definition 1, property 2). In general, this dictionary may have better significance, but will be much larger. Note that the bit-size of P^{R_u} is a slight overestimate, as there are some combinations of points that would not constitute a valid password, due to two or more points being within $t = 9$ pixels of each other. As discussed in Section 6.3.2, the total number of users we have data for is $u = 33$ for *cars*, and $u = 35$ for *pool*. Thus, the size of P^{R_u} for *cars* is $P(165, 5) = 2^{36.7}$ entries, and for *pool* is $P(175, 5) = 2^{37.1}$ entries.

Set	*cars* ($u = 33$)		*pool* ($u = 35$)	
	m	bitsize	m	bitsize
P^{R_u}	165	36.7	175	37.1
P^{V_u}	104	33.4	77	31.1
$P^{R_{25}}$	125	34.7	125	34.7
$P^{V_{25}}$	85	31.9	59	29.2
$P^{R_{20}}$	100	33.1	100	33.1
$P^{V_{20}}$	72	30.6	52	28.2
$P^{R_{15}}$	75	30.9	75	30.9
$P^{V_{15}}$	56	28.8	41	26.4

Table 6.1: Bit-sizes of dictionaries using different sets. All subsets of users (after the first two rows) are the average result of 10 randomly selected subsets of $u = 15, 20, 25$ lab-study passwords. For rows 1 and 2, note that $u = 33$ and 35. m is the alphabet size, which defines the dictionary bitsize. P^{V_u} is our S_{C_2} (other subscripts denote what our S_{C_2} would have been if generated from a smaller data set of that size); see text for descriptions of P^V and P^R.

Table 6.1 shows how using clusters (P^{V_u}) instead of raw click-point data (P^{R_u}) reduces the size of the dictionary. A larger reduction in the number of clusters (see column m) indicates that there are more click-points in the same T-region, i.e., there are more hot-spots (recall Section 2.4.2), and/or larger clusters indicating these hot-spots. It is interesting to note that when $u = 15$, *pool* achieves a 20% larger reduction (from the number of click points to the number of clusters in column m) than that of *cars*: the same difference as when $u = 20$ and $u = 25$. This may indicate that collecting data from as few as 15 people could provide as good an estimate of the relative hot-spotting between images as collecting data from 33 or more people. For a more detailed discussion of hot-spotting, see Section 6.7.

6.2.8 Approximate Size of Class C_3 Graphical Dictionaries

For Class C_3 graphical dictionaries, we calculate the size of three of the click-order patterns discussed in Section 6.2.4: DIAG, HOR, and VER. The results are provided in Table 6.2. For reference, the full password space is 43 bits. Note that these dictionaries consider all possible permutations of T-regions that can be categorized as having one of these click-order patterns, and thus it is image-independent.

Dictionary	bitsize
$DIAG$	32.6
HOR	37.6
VER	37.9

Table 6.2: Dictionaries that exploit click-order patterns.

The conditions used to define the size of these sets are provided in Section 6.3. Although we can test a set of points for membership to CWCCW, and provide this number in our combined results in Section 6.2.9, it is not clear to us how to efficiently generate all g_{cb}-passwords that belong to this set. Thus, we only consider CWCCW as an optimization for a Class C_2 dictionary.

The results in Table 6.2 for HOR and VER are sensible when considering the effect of permutations versus combinations. The full 43-bit password space considers every 5-permutation of all possible click-points. For each combination of click-points, the HOR and VER click-order patterns consider at least two permutations out of the $5! = 120$ possible. If only one permutation for each combination of all possible click-points were added to a dictionary, it would be of size $\frac{2^{43}}{120} = 2^{36}$. Considering two permutations for each combination would double the number, resulting in a 37-bit dictionary. The HOR and VER dictionaries are slightly larger since there is more than one combination when e.g., considering a straight horizontal line for the

VER dictionary (all permutations would be accepted). It is sensible that the $DIAG$ dictionary has considerably fewer elements since many combinations of points would not have any permutations that would qualify for membership (e.g., consider a zig-zag formation).

6.2.9 Approximate Size of Combined Class C_2 and C_3 Dictionaries

To demonstrate the combined effect of Class C_2 and Class C_3, we combine S_{C_2} (more specifically, P^{V_u} from Table 6.1) with the four subsets of S_{C_3} to obtain the results in Table 6.3.

Set	*cars* ($u = 33$)		*pool* ($u = 35$)	
	m	bitsize	m	bitsize
P^{V_u}	104	33.4	77	31.1
$P^{V_u} \cap S_{C_3}$-$DIAG$	104	22.7	77	20.7
$P^{V_u} \cap S_{C_3}$-HOR	104	28.0	77	25.9
$P^{V_u} \cap S_{C_3}$-VER	104	28.4	77	26.0
$P^{V_u} \cap S_{C_3}$-$CWCCW$	104	27.8	77	25.7

Table 6.3: Dictionaries built from $S_{C_2} \cap S_{C_3}$. m is the alphabet size, and all human-computed click-point data collected (see Section 6.3.2) is used.

Table 6.3 shows how despite having different alphabet sizes ($m = 104$ for *cars*, and $m = 77$ for *pool*), each click-order pattern reduces the dictionary sizes by approximately the same number of bits as when applied to the full password space (recall Section 6.2.8): approximately 5 bits for both the HOR and VER dictionaries, and approximately 10-11 bits for the $DIAG$ dictionary. We see here that the $CWCCW$ click-order pattern reduces the dictionary sizes by approximately 5.5 bits.

6.3 Model Details

Here we discuss the details of how we estimated the sizes of Class C_1, Class C_2, and Class C_3 graphical dictionaries. We discuss how we generate Class C_1 (image processing tools) in Section 6.3.1, Class C_2 (human-computed data set) in Section 6.3.2, and Class C_3 in Section 6.3.3. Combining (i.e., examining the intersection of)

Class C_2 and Class C_3 is simply a matter of checking those 5-permutations in Class C_2 for whether they conform to a click-order pattern in Class C_3.

6.3.1 Class C_1 Graphical Dictionaries (Image Processing)

Here we discuss how we create a Class C_1 attack dictionary for g_{cb}-passwords by automated image processing means. Such automated methods should be easier for an attacker to launch than an attack using a Class C_2 dictionary. We create this attack dictionary, which we call S_{C_1}, by modelling user choice using a set of image processing methods and tools detailed herein.

To begin, we identify a *candidate click-point* to be a point that is *salient*, and identifiable with precision within the system's error tolerance. We estimate candidate click-points (CCPs) by implementing a variation of Itti et al.'s bottom-up model of visual attention (VA) [67], and combining it with Harris corner detection [60].

Corner detection picks out the areas of an image that have variations of intensity in horizontal and vertical directions; thus we expect it should provide a reasonable measure of whether a point is identifiable by human users. Itti et al.'s VA determines areas that stand out from their surroundings, i.e., have higher saliency. Briefly, VA calculates a saliency map of the image based on 3 channels (color, intensity, and orientation) over multiple scales. The saliency map is a grayscale image whose brighter areas (i.e., those with higher intensity values) represent more conspicuous locations. A viewer's focus of attention should theoretically move from the most conspicuous locations (represented by the highest intensity areas on the saliency map) to the least. We assume that users will prefer, and are more likely to choose click-points from areas which draw their visual attention.

We implemented a variation of VA and combined it with Harris corner detection to obtain a prioritized list of candidate click-points *(CCP-list)* as follows.

1. Calculate a VA saliency map (see Fig. 6.3(b)) using slightly smaller scales than Itti et al. [67] (to reflect our interest in smaller image details). The higher-intensity pixel values of the saliency map reflect the most "conspicuous" (and distinguishable) areas.

2. Calculate the corner locations using the Harris corner detection function as implemented by Kovesi [79][3] (see Fig. 6.3(c)).

3. Use the corner locations as a bitmask for the saliency map, producing what we call a *cornered saliency map* (CSM). CSM points are then the remaining pixels with non-zero intensity values.

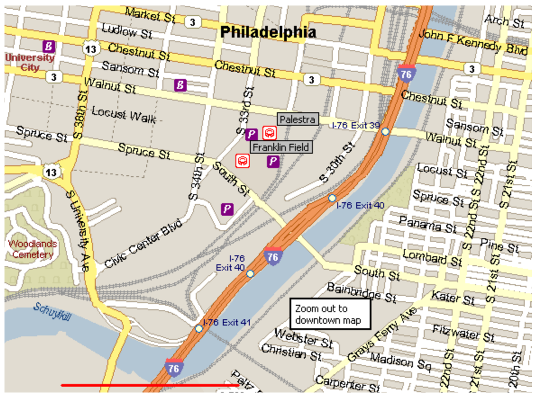
4. Compute an ordered CCP-list of the highest to lowest intensity-valued CSM points. Similar to the focus-of-attention inhibitors used by Itti et al., we inhibit a CSM point (and its surrounding T-region) once it has been added to the CCP-list so it is not chosen again (see Fig. 6.3(d)). The CCP-list is at least as long as the alphabet size (414), but is a prioritized list, ranking points from (the hypothesized) most to least likely.

Finally, we generate all 5-permutations of the top-ranked $\frac{1}{3}$ entries in the CCP-list, which creating a 35-bit dictionary. We repeat the same method to generate two other dictionaries: a 33-bit dictionary using the top-ranked $\frac{1}{4}$ entries in the CCP-list, and a 28-bit dictionary using the top-ranked $\frac{1}{8}$ entries in the CCP-list. Our reason for choosing a set of fractions of the CCP-list was to see how small an alphabet (if successful) was necessary to model user choice.

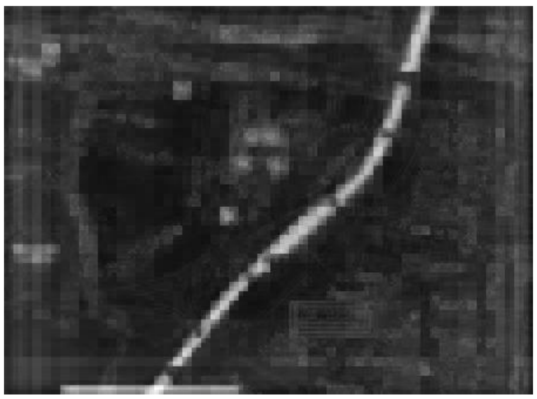
6.3.2 Class C_2 Graphical Dictionaries (Human-Seeded)

Here we describe how we calculate our Class C_2 graphical dictionary; we call this implementation S_{C_2}. We collect data from users in a separate simplified context: an in-lab, short-term study as described below. We use the data collected in this lab study as input to the clustering algorithm described further below to calculate clusters. All permutations of these clusters are then added to our S_{C_2} dictionary, ordered by decreasing probability. The probability of each permutation of clusters is defined by the product of its observed cluster probabilities (see further below for how the observed probability of a cluster is defined).

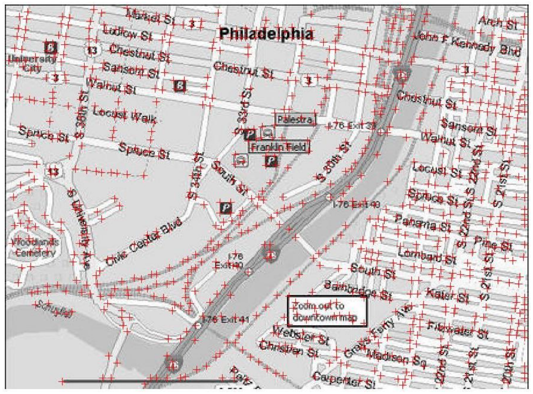
[3] As *harris(image, 1, 1000, 3)*



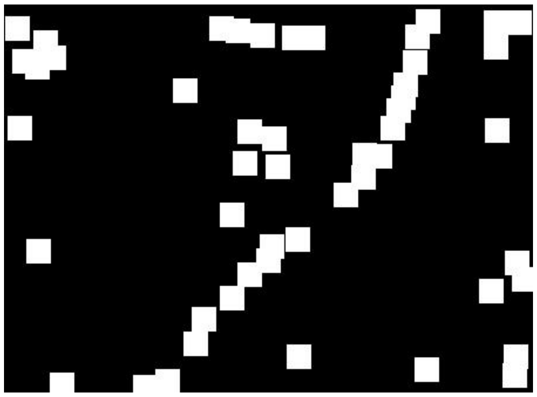
(a) Original image [163].



(b) Saliency map.



(c) Corner detection output.



(d) Cornered saliency map (CSM) after inhibiting top 51 CCP-list points.

Figure 6.3: Illustration of our method of creating a CCP-list (best viewed electronically).

Lab Study

Here we report the details of a university-approved 43-user study of click-based graphical passwords in a controlled lab environment. Each user session was conducted individually and lasted about one hour. Participants were all university students who were not studying (or experts in) computer security. Each user was asked to create a click-based graphical password on 17 different images (most of these are reproduced in Figures 6.4 and 9.1; others are available from the author). Four of the images are from a previous click-based graphical password study by Wiedenbeck et al. [163]; the other 13 were selected to provide a range of values based on two image processing measures that we expected to reflect the amount of detail: the number of segments found from image segmentation [46] and the number of corners found from corner detection [60]. Seven of the 13 images were chosen to be those we "intuitively" believed would encourage fewer hot-spots; this is in addition to the four chosen in earlier research by others [163] using intuition (no further details were provided on their image selection methodology).

We implemented a web-based study. Each user was provided a brief explanation of what click-based graphical passwords are, and given two images to practice creating and confirming such passwords. To keep the parameters as consistent as possible with previous usability experiments of such passwords [164], we used 5 click-points for each password, an image size of 451×331 pixels, and a 19×19 pixel square of error tolerance. Wiedenbeck et al. [164] used a tolerance of 20×20, allowing 10 pixels of tolerated error on one side and 9 on the other. To keep the error tolerance consistent on all sides, we approximate this error tolerance using 19×19. Users were instructed to choose a password by clicking on 5 points, with no two the same. Although the software did not enforce this condition, subsequent analysis showed that the effect on the resulting cluster sizes was negligible for all images except *pcb*; for more details, see caption of Figure 6.10. We did not assume a specific encoding scheme (e.g., robust discretization [13] or other grid-based methods); the concept of hot-spots and user choice of click-points is general enough to apply across all encoding schemes. To allow for detailed analysis, we store and compare the actual click-points.

Once the user had a chance to practice a few passwords, the main part of the experiment began. For each image, the user was asked to create a click-based graphical password that they could remember but that others will not be able to guess, and to pretend that it is protecting their bank information. After initial creation, the user was asked to confirm their password to ensure they could repeat their click-points. On successful confirmation, the user was given 3D mental rotation tasks [123] as a distractor for at least 30 seconds. This distractor was presented to remove the password from their visual working memory, and thus simulate the effect of the passage of time. After this period of memory tasks, the user was provided the image again and asked to log in using their previously selected password. If the user could not confirm after two failed attempts or log in after one failed attempt, they were permitted to reset their password for that image and try again. If the user did not like the image and felt they could not create and remember a password on it, they were permitted to skip the image. Only two images had a significant number of skips: *paperclips* and *bee*. This suggests some passwords for these images were not repeatable, and we suspect our results for these images would show lower relative security in practice.

To avoid any dependence on the order of images presented, each user was presented a random (but unique) shuffled ordering of the 17 images used. Since most users did not make it through all 17 images, the number of graphical passwords created per image ranged from 32 to 40, for the 43 users. Two users had a "jumpy" mouse, but we do not expect this to affect our present focus – the location of selected click-points. This short-term study was intended to collect data on initial user choice; although the mental rotation tasks work to remove the password from working memory, it does not account for any effect caused by password resets over time due to forgotten passwords. For this reason, we use the long-term field study (Section 6.5.1) which does account for this effect, as the primary data set for testing the success of our graphical dictionaries.

Clustering Algorithm

To calculate clusters (the size of which defines hot-spots) based on any user data set of click-points, we assign all of the observed user click-points to clusters as follows. Let R be the raw (unprocessed) set of click-points, M a list of temporary clusters, and V the final resulting set of clusters.

1. For each $c_k \in R$, let B_k be a temporary cluster containing click-point c_k. Temporarily assign all user click-points in R within c_k's T-region to B_k. Add B_k to M.
2. Sort all clusters in M by size, in decreasing order.
3. Greedily make permanent assignments of click-points to clusters as follows. Let B_ℓ be the largest cluster in M. Permanently assign each click-point $c_k \in B_\ell$ to B_ℓ, then delete each $c_k \in B_\ell$ from all other clusters in M. Delete B_ℓ from M, and add B_ℓ to V. Repeat until M is empty.

This process determines a set V of (non-empty) clusters and their sizes. We then calculate the observed "probability" p_j (based on our data set) of the cluster j being clicked, as cluster size divided by total clicks observed. The final S_{C_2} dictionary contains all 5-permutations of the final clusters in V (i.e., P^V in our notation). The probability of each 5-permutation in P^V is then defined by the product of the probability of its 5 composite clusters.

6.3.3 Class C_3 Graphical Dictionaries

We count all Class C_3 passwords by using the centers of all T-regions in the entire alphabet space. Our base set of T-region centers are aligned such that their T-regions do not overlap, meaning that they begin at $(10, 10)$, and are in subsequent increments of the T-region size (19). Only those 5-permutations whose click-points $(x_i, y_i), i = 1, 2, \ldots, 5$ follow one of the following click-order pattern conditions are added to that particular dictionary:

- HOR: LR or RL, where $LR = (x_1 \le x_2 \le x_3 \le x_4 \le x_5)$ and $RL = (x_1 \ge x_2 \ge x_3 \ge x_4 \ge x_5)$.
- VER: TB or BT, where $TB = (y_1 \le y_2 \le y_3 \le y_4 \le y_5)$ and $BT = (y_1 \ge y_2 \ge y_3 \ge y_4 \ge y_5)$.
- DIAG: LR and (TB or BT), or RL and (TB or BT).
- CWCCW: All sequences of three consecutive points are in the same direction (clockwise or counter-clockwise as computed as in [16]), and the sum of the angles between these sequences of three consecutive points is no greater than 360 degrees.

In words, HOR is a horizontal sweep from right to left or left to right; VER is a vertical sweep from top to bottom or bottom to top; DIAG is a sweep in both a certain horizontal and vertical direction; and CWCCW is either a clockwise or counter-clockwise, non-overlapping sweep.

Note that in each of the listed conditions, equality takes the error tolerance into account; for example, if x_1 to x_4 all follow a left to right click-order pattern, and then

$x_5 \geq x_4 - t$, the entire password will be considered to have a HOR click-order pattern. We call our implementation of Class C_3 dictionaries S_{C_3}, and use S_{C_3}- to prefix each of the dictionaries described above (e.g., S_{C_3}-DIAG).

6.4 Evaluation Methodology

Here we discuss our method to measure which of our dictionaries qualify as weak password subspaces for the PassPoints scheme. To do so, we must define thresholds for how much risk we are willing to accept (i.e., α in Definition 1), and the resources of an attacker we expect protection from (i.e., t_e in Definition 1). Of course, these thresholds are not fixed as they are dependent upon the amount of risk that a system administrator (or user) is willing to accept, and the attackers that he/she would like protection against. As in Chapter 5, we choose conservative thresholds so as to not overestimate the success of our attack dictionaries, but we also stress that attackers could easily have ten-thousand times more resources than what we assume herein if they are sufficiently motivated to rent a botnet.

The full password space for PassPoints using a background image of size 451×331 and a 19×19 T-region is only 43 bits. Thus, if we use the same threshold value of 46.75 bits as for pure-recall graphical passwords in Section 5.4, the full space would immediately qualify as a candidate weak password space. Clearly, if PassPoints were to be deployed with these parameters and remain reasonably secure against offline guessing attacks, it must somehow increase the computational expense of each offline guess. One way to increase the cost of a guess is through increasing the cost of the hashing method used for storage and comparison (e.g., using the methods from Section 2.3.2). If we assume that a higher-cost hashing method is used, we must adjust our value of t_e. We assume that if a password scheme is to replace text passwords, it should be at least as computationally expensive to exhaust as the full 53-bit password space of 8-character text passwords (using upper and lowercase characters, numbers, and the 34 special characters available on a keyboard). For the time to exhaust a 53-bit text password space to equal the time to exhaust a 43-bit PassPoints password space, PassPoints might use a hashing method that is $2^{10} = 1024$ times more computationally expensive. Using the same MD5 measure as Section 5.4, this means that approximately 12000 MD5 hashes could be performed per second on a 2.66GHz processor. Using the same attacker resource assumptions as in Section 5.4, that the attacker is using personal resources, which by 2008 could easily be two quad-core machines with 2.66GHz processors (for a total of eight processors), and that the attacker will give up after 2 weeks, provides $t_e = 2^{33.75}$.

To verify Class C_1, Class C_2, and Class C_3's hypothesized status as weak password subspaces, we determine the number of g_{cb}-passwords that fall within them from two separate user studies; our primary data set is from a long-term field study using two different background images (see Section 6.5.1), and for Class C_1, we also use the short-term, single session lab study using 17 different background images. We use data from the lab study as a human-computed data set to generate Class C_2 dictionaries, and use the data from the field study to test the success of these dictionaries. We declare a candidate subspace as weak if it contains at least the same percentage most recently found to fall to basic text dictionary attack (5%) [82]. Thus, our significance threshold $\alpha = .05$, meaning that out of a system of 20 users, we would expect one account to be compromised.

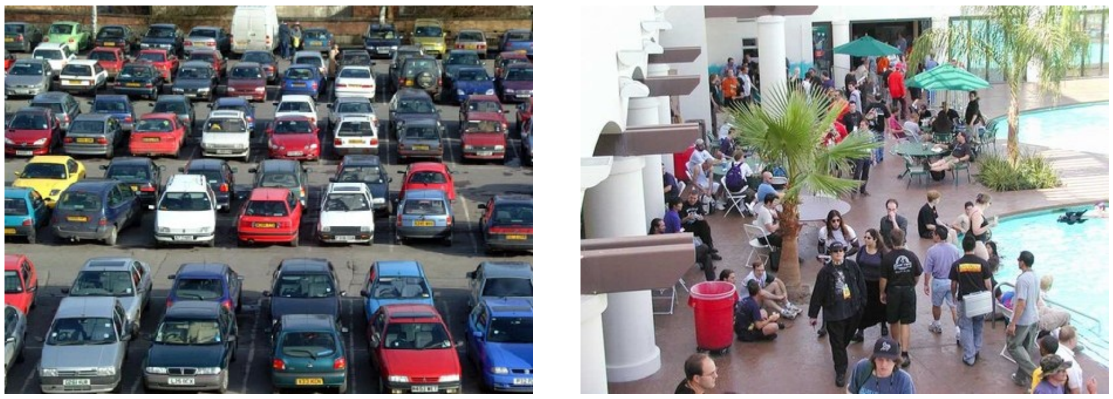
We define the success of our predictive model's application to g_{cb}-passwords by whether it produces candidate weak password subspaces (such that they contain fewer than $t_e = 2^{33.75}$ entries), and that through user studies, these candidate password subspaces are shown to facilitate correctly guessing at least 5% ($\alpha = 0.05$) of passwords.

6.5 Results for Applying Class C_1, C_2, and C_3 to PassPoints

Our predictive models found two weak password subspaces that we call S_{C_2} and S_{C_3}-DIAG. Our S_{C_1} (when using the 33 bit dictionary) creates a weak password subspace for some images, but not all. We found the S_{C_2} and S_{C_3}-DIAG subspaces to be both small and generable (satisfying properties 1 and 3); using the parameters of 5 clicks, a 19×19 T-region, and an image size of 431×331, we found S_{C_2} to be 31.1 bits for the *pool* image and 33.4 bits for the *cars* image, S_{C_3}-DIAG to be 32.6 bits, and $S_{C_2} \cap S_{C_3}$-DIAG to be 20.7 bits for the *pool* image and 22.7 bits for the *cars* image. Our field study supports that they are significant (satisfying property 2); we found that 20-36% of g_{cb}-passwords would belong to S_{C_2}, 26-46% of g_{cb}-passwords would belong to S_{C_3}-DIAG, and 8-10% would belong to $S_{C_2} \cap S_{C_3}$-DIAG. The field study used to evaluate our dictionaries is described in further detail in Section 6.5.1, and its limitations are discussed in Section 6.5.6. The detailed results for applying each of our dictionaries are discussed in Sections 6.5.2–6.5.5.

6.5.1 Supporting Evidence: Field Study

Here we describe a 7-week or longer (depending on the user), university-approved field study of 223 user accounts on two different background images. We collected click-based graphical password data to evaluate the security of this style of graphical passwords against various attacks. We use the entropy and expected guesses measures from our lab study to choose two images that would apparently offer different levels of security (although both are highly detailed): *pool* and *cars* (see Figure 6.4). The lab study showed that the *pool* image had a medium amount of clustering (cf. Fig. 6.15 in Section 6.7), while the *cars* image had nearly the least amount of clustering among the 17 images tested. Both images had a low number of skips in the lab study, indicating that they did not cause problems for users with password creation. We chose the *cars* image to give this scheme the best chance we could in terms of anticipated security.



(a) *cars* (originally from [19]). (b) *pool* (originally from [163, 164]).

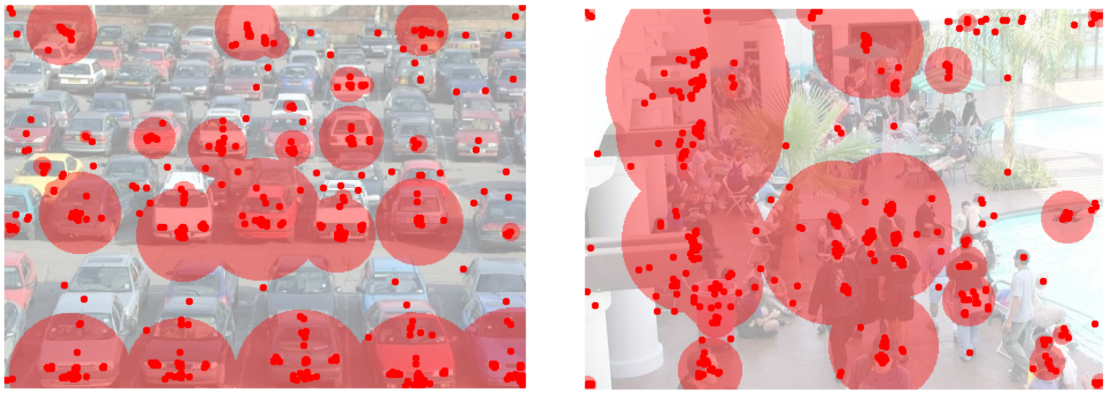
Figure 6.4: Images used in field study.

We implemented a web-based version of PassPoints, used by three first-year undergraduate classes: two were first year courses for computer science students, while the third was a first year course for non-computer science students enrolled in a science degree. The students used the system for at least 7 weeks to gain access to their course notes, tutorials, and assignment solutions. For comparison with previous usability studies on the subject, and our lab study, we used an image size of 451×331 pixels. After the user entered their username and course, the screen displayed their background image and a small black square above the image to indicate their tolerance square size. For about half of users (for each image), a 19×19 T-region was used, and for the other half, a 13×13 T-region.[4] The system enforced that each password had to be 5 clicks and that no click-point could be within $t = 9$ pixels of another (vertically and horizontally). To complete initial password creation, a user had to successfully confirm their password once. After initial creation, users were permitted to reset their password at any time using a previously set secret question and answer.

Users were permitted to login from any machine (home, school, or other), and were provided an online FAQ and help. The users were asked that they keep in mind that their click-points are a password, and that while they will need to pick points they can remember, they should not pick points that someone else will be able to guess. Each class was also provided a brief overview of the system, explaining that their click-points in subsequent logins must be within the tolerance shown by a small square above the background image, and that the input order matters. We only use the final passwords created by each user that were demonstrated as successfully recalled at least one subsequent time (i.e., at least once after the initial create and confirm). We only consider these final passwords in order to have some confidence that the passwords we analyze have some degree of memorability. We also only use data from 223 out of 378 accounts, as this was the number that provided the required consent. These 223 user accounts map to 189 distinct users as 34 users in our study belonged to two classes; all but one of these users were assigned a different image for each account, and both accounts for a given user were set to have the same error tolerance. Of the 223 user accounts, 114 used *pool* and 109 used *cars* as a background image.

[4]Analysis showed little difference between the points chosen for these different tolerance groups.

Using our clustering algorithm (recall Section 6.3.2), we created a visualization of the data collected, which illustrates the hot-spots, shown in Figure 6.5. The small red dots represent single click-points, and the halos (i.e., the transparent larger red circles) have size proportional to the number of underlying clicks (similar to population diagrams). For further discussion of this hot-spotting and its relationship to the hot-spotting from the lab study, see Section 6.7.



(a) *cars* (originally from [19]). (b) *pool* (originally from [163, 164]).

Figure 6.5: Observed clustering (field study). Halo diameter is 5 times the number of underlying clicks.

6.5.2 Class C_1 (Image Processing) Dictionary Results

We evaluated the performance of S_{C_1} using the data from both the lab and field user studies. We also use the lab study for evaluating our results here, as it provides us with a larger set of images to evaluate our image processing tools against.

We first examined how well the first half (top 207) of the CCP-list overlaps with the observed high-probability clusters from our lab user study (i.e., those clusters of size at least 5). We found that this half-alphabet contained all high-probability clusters on the *icons*, *faces*, and *cars* images, and most of the high-probability clusters on 11 of the 17 images. Most of the images that our model performed poorly on appeared to be due to the saliency map algorithm being overloaded with too much detail (*pcb*, *citymap-gr*, *paperclips*, *smarties*, and *truck* images). The other image on which this approach did not perform well (*mural*) appears to be due to the corner masking in step (3); the high probability points were centroids of circles.

To evaluate how well the CCP-list works at modelling users' *entire* passwords (rather than just a subset of click-points within a password), we used the top ranked one-third of the CCP-list values (i.e., the top 138 points for each image) to build a graphical dictionary and carry out a dictionary attack against the observed passwords from both user studies (i.e., on all 17 images in the lab study, and the *cars* and *pool* images again in the field study). We found that for some images, this 35-bit dictionary was able to guess a large number of user passwords (30% for the *icons* image and 29% for the *philadelphia* map image). For both short and long-term studies, our 35-bit S_{C_1} guessed 9.1% of passwords for the *cars* image. Our 33-bit S_{C_1} (built from the top 103 ranked CCP-list alphabet) correctly guessed 27% of passwords for *icons* and 20% for *philadelphia*, but did not guess any for 9 out of 17 images. For the field study, this dictionary guessed 5.5% of passwords for *cars*, but none for *pool*. A 28-bit S_{C_1} (built from the top 51 ranked CCP-list alphabet) correctly guessed 8 passwords (22%) from the *icons* image and 6 passwords (17%) from the *philadelphia* image, but did not guess any for 15 out of 17 images (thus no table is shown).

Results of these automated graphical dictionary attacks are summarized in Tables 6.4 and 6.5.

Tables 6.4 and 6.5 show that our S_{C_1} is a weak password subspace for some images, but not all images. Using the 35-bit S_{C_1}, only 10 of the 17 images from the lab study, and only one of the two images from the field study, have at least 5% of passwords guessed. Using the 33-bit S_{C_1} (which using our methodology described in Section 6.4 defines a candidate weak password subspace), only 4 of the 17 images from the lab study, and only one of the two images from the field study, have at least 5% of passwords guessed.

These results imply that on some images, an attacker performing an automated attack is likely to be able to significantly cut down his search space. This method also seems to perform well on the images for which the visual attention model made more definite decisions – the saliency map shows a smaller number of areas standing out, as indicated visually by a generally darker saliency map with a few high-intensity (white) areas. An attacker interested in any one of a set of accounts could go after accounts using a background image that the visual attention model performed well

Image	passwords guessed (lab study)	passwords guessed (field study)
1. *paperclips*	2/36 (5.5%)	–
2. *cdcovers*	2/35 (5.7%)	–
3. *philadelphia*	10/35 (28.6%)	–
4. *toys*	2/39 (5.1%)	–
5. *bee*	1/40 (2.5%)	–
6. *faces*	0/32 (0.0%)	–
7. *citymap-nl*	1/34 (2.9%)	–
8. *icons*	11/37 (29.7%)	–
9. *smarties*	5/37 (13.5%)	–
10. *cars*	3/33 (9.1%)	10/109 (9.1%)
11. *pcb*	3/36 (8.3%)	–
12. *citymap-gr*	0/34 (0.0%)	–
13. *pool*	1/35 (2.9%)	2/114 (0.9%)
14. *mural*	1/36 (2.8%)	–
15. *corinthian*	3/35 (8.6%)	–
16. *truck*	1/35 (2.9%)	–
17. *tea*	2/38 (5.3%)	–

Table 6.4: Passwords correctly guessed (using a 35-bit S_{C_1} dictionary based on a CCP-list). The number of target passwords is different for most images (32 to 40 for the lab study).

Image	passwords guessed (lab study)	passwords guessed (field study)
1. *paperclips*	0/36 (0.0%)	–
2. *cdcovers*	0/35 (0.0%)	–
3. *philadelphia*	7/35 (20.0%)	–
4. *toys*	1/39 (2.6%)	–
5. *bee*	1/40 (2.5%)	–
6. *faces*	0/32 (0.0%)	–
7. *citymap-nl*	0/34 (0.0%)	–
8. *icons*	10/37 (27.0%)	–
9. *smarties*	2/37 (5.4%)	–
10. *cars*	1/33 (3.0%)	6/109 (5.5%)
11. *pcb*	0/36 (0.0%)	–
12. *citymap-gr*	0/34 (0.0%)	–
13. *pool*	0/35 (0.0%)	0/114 (0.0%)
14. *mural*	0/36 (0.0%)	–
15. *corinthian*	3/35 (8.6%)	–
16. *truck*	0/35 (0.0%)	–
17. *tea*	1/38 (2.6%)	–

Table 6.5: Passwords correctly guessed (using a 33-bit S_{C_1} dictionary based on a CCP-list). The number of target passwords is different for most images (32 to 40 for the lab study).

on. The background images could be obtained for analysis by entering known user IDs.

In essence, this method achieves a reduction (by leaving out some "less likely" points) from a 43-bit full password space to a 35 or 33-bit dictionary that is successful on some images. The 43-bit full password space is the proper base for comparison here, since an actual attacker with no a priori knowledge must consider all T-regions in an image. However, we believe this S_{C_1} dictionary could be improved through a few methods. The images that it performed poorly on appeared to be due to failure in creating a useful visual attention model saliency map. The saliency maps seem to fail when there are no areas that stand out from their surroundings in the channels used in saliency map construction (color, intensity, and orientation). Varying some parameters involved in the saliency map creation (e.g., the scales used for defining centers and surrounds, or the weighting applied to intermediate maps for the final linear recombination) might lead to better results. Further, centroids of objects that "stand out" to a user will not be included in this model (as only corners are included); adding object centroids to the bitmask is thus an avenue for improvement. Dirik et al. [40] (recall Section 3.3.4) use centroids in their work, but adding object centroids to our bitmask would not result in an equivalent method, as theirs is not based on the saliency map from Itti et al.'s model of bottom-up visual attention [67]. Instead, they describe their own algorithm that involves differences in colour and intensity between segments.

6.5.3 Class C_2 (Human-Seeded) Dictionary Results

We evaluate the performance of S_{C_2} using the data collected in the field study. Table 6.6 shows the number of passwords guessed after applying S_{C_2}. Recall that P^{R_u} is a dictionary containing all 5-permutations of the raw click-points collected from u users, and P^{V_u} contains all 5-permutations of the clusters calculated from the click-points collected from u users. We only formally consider P^V as S_{C_2}.

The results in Table 6.6 shows that although the clustering algorithm reduces the size of the dictionary, it also reduces its significance. Despite this reduction in significance, these dictionaries still guess a large number of passwords (20-36% when all users are used). Table 6.6 also shows the effect of reducing the number of people for creating a human-computed data set: as expected, it also reduces the significance of the dictionary generated, but as few as 15 people can generate enough information to guess 11-23% of passwords (on average).

Set	*cars* ($u = 33$)				*pool* ($u = 35$)			
	bit-size	# passwords guessed out of 109			bit-size	# passwords guessed out of 114		
		avg	min	max		avg	min	max
P^{R_u}	36.7	37(34%)	†	†	37.1	59(52%)	†	†
P^{V_u}	33.4	22(20%)	†	†	31.1	41(36%)	†	†
$P^{R_{25}}$	34.7	24(22%)	9(8%)	35(32%)	34.7	42(37%)	29(25%)	56(49%)
$P^{V_{25}}$	31.9	21(19%)	7(6%)	27(25%)	29.2	34(29%)	19(17%)	47(41%)
$P^{R_{20}}$	33.1	22(20%)	8(7%)	32(29%)	33.1	35(31%)	24(21%)	55(48%)
$P^{V_{20}}$	30.6	17(16%)	8(7%)	30(28%)	28.2	28(25%)	18(16%)	43(38%)
$P^{R_{15}}$	30.9	14(13%)	4(4%)	25(23%)	30.9	30(27%)	20(18%)	45(39%)
$P^{V_{15}}$	28.8	12(11%)	4(4%)	24(22%)	26.4	26(23%)	14(12%)	43(38%)

Table 6.6: Dictionary attacks using different sets. All percentages in the table (after the first two rows) are the result of 10 randomly selected subsets of $u = 15, 20, 25$ short-term study user passwords. For rows 1 and 2, note that $u = 33$ and 35. See text for descriptions of P^V and P^R. †The first two rows use all data from the short-term study to seed a single dictionary, and as such, there are no average, max, or min values to report.

The most striking result shown is that initial password choices harvested from 15 users, in a setting where long term recall is not required, can be used to generate (on average) 23% of user passwords for *pool* (see $P^{V_{15}}$). As we expected, *cars* was not as easily attacked as *pool* (guessing on average 11% for $P^{V_{15}}$); more user passwords are required to seed a dictionary that achieves similar success rates (see $P^{V_{25}}$).

Formally, we consider our Class C_2 dictionary to be P^V (i.e., when clusters are used). We can place an ordering on the Class C_2 dictionary such that the passwords are ordered from most to least probable (as defined by the probabilities of each cluster in the password). We further examine the cumulative distribution function (CDF) of P^{V_u} for each image, as shown in Figure 6.6.

Figure 6.6 illustrates how much more effective the S_{C_2} dictionary is for *pool*: about 10% of passwords are guessed in the first 10000 guesses, and 5% are guessed within the

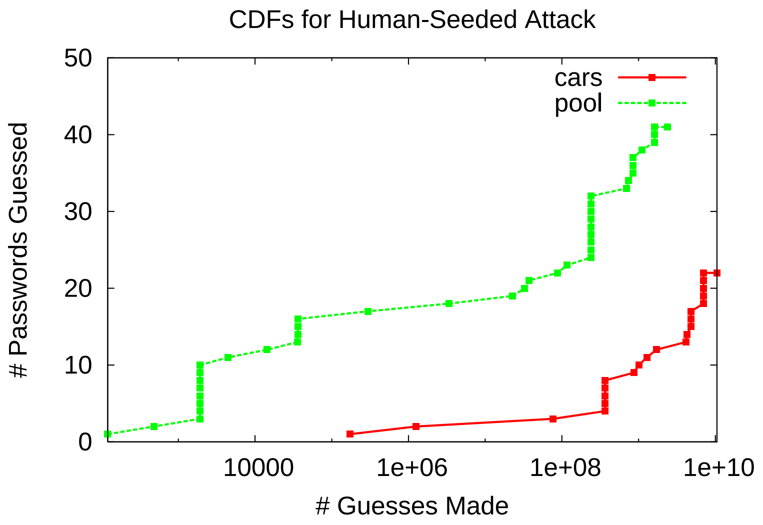


Figure 6.6: CDF of S_{C_2} (i.e., P^{V_u}) for *pool* and *cars*. The total number of passwords in the field study password database is 109 for *cars* and 114 for *pool*.

first 2000 guesses. In contrast, the S_{C_2} dictionary for *cars* guesses 10% of passwords only after over 10^9 guesses, and 5% after over 4×10^8 guesses. This is likely due to the low amount of clustering we observed in the data collected in the lab study on *cars*, leading to most clusters having the same probability, producing a less optimal ordering.

In Figure 6.6 (and in later CDF figures), it appears that some guesses match a large number of passwords. In this attack, a particular ordering of a combination of five click-points is not given a higher priority over another. Thus, we only report the number of successful guesses for a combination of points after having guessed all 120 permutations, meaning that when a guess appears to be particularly popular, it indicates the combination of points is popular, not necessarily a single permutation. All permutations are guessed in this attack, we simply choose to report the results after all permutations have been guessed.

6.5.4 Class C_3 Dictionary Results

We evaluate the performance of S_{C_3} using the data collected in the field study. Table 6.7 shows the number of passwords guessed after applying S_{C_3} for various click-order patterns. Recall Section 6.2.4 for a description of each of the click-order patterns shown.

Dictionary	*cars*		*pool*	
	bitsize	# passwords guessed	bitsize	# passwords guessed
$DIAG$	32.6	50/109 (45.9%)	32.6	30/114 (26.3%)
HOR	37.6	65/109 (59.6%)	37.6	53/114 (46.5%)
VER	37.9	71/109 (65.1%)	37.9	39/114 (34.2%)
$CWCCW$	†	8/109 (7.3%)	†	13/114 (11.4%)

Table 6.7: S_{C_3} dictionary attack results using various click-order patterns for all T-regions. †CWCCW would take many years to calculate the size of (for all T-regions).

The most striking result in Table 6.7 is that the DIAG click-order pattern, which is the smallest S_{C_3} dictionary, guesses almost 46% of passwords for *cars*, and 26% for *pool*; this result shows that S_{C_3}-DIAG is more significant than that of S_{C_2}. These results also give some insight as to which click-order patterns are most popular, and how much these click-order patterns can differ depending on the image. For example, the $DIAG$, HOR, and VER click-order patterns are much more popular in *cars* than *pool*, which is sensible given that *cars* depicts cars parked in straight rows. It is interesting that despite few obvious straight-line structures in *pool* (aside from the pillars on the left hand side), the $DIAG$, HOR, and VER patterns are all still quite popular. These results also show that $CWCCW$ only guesses 7-11% of passwords, showing that this click-order pattern is the least popular of those examined.

Overall, the success rates observed in Table 6.7 suggest the following as the best attack ordering within S_{C_3}: $DIAG$, HOR, VER, $CWCCW$. It also suggests that the $DIAG$ dictionary is the weakest subspace of all those considered in S_{C_3}.

6.5.5 Combined Class C_2 and C_3 Dictionary Results

We evaluate $S_{C_2} \cap S_{C_3}$ using the data collected in the field study. To initiate exploration of combining these two classes, we begin by examining the effect of intersecting S_{C_3} with S_{C_2} (when $u = 15$, i.e., $P^{V_{15}}$). For each image, we select one $P^{V_{15}}$ dictionary to intersect with S_{C_3}. This dictionary is one of the ten randomly selected P^V subsets that were averaged (results of this average are in Table 6.6). We selected the dictionary whose guessing success was closest to the average reported in Table 6.6. The success rate that these dictionaries achieve (before applying click-order patterns) is provided in the first row of Table 6.8.

Click-order pattern	*cars* image		*pool* image	
	# passwords guessed of 109	dictionary size (bits)	# passwords guessed of 114	dictionary size (bits)
$P^{V_{15}}$ (with no pattern)	13 (12%)	29.2	22 (19%)	27.1
$P^{V_{15}} \cap S_{C_3}$-*HOR*	11 (10%)	23.8	19 (17%)	22.0
$P^{V_{15}} \cap S_{C_3}$-*VER*	12 (11%)	24.4	15 (13%)	21.9
$P^{V_{15}} \cap S_{C_3}$-*CWCCW*	0 (0%)	24.0	4 (4%)	21.7
$P^{V_{15}} \cap S_{C_3}$-*DIAG*	11 (10%)	18.4	14 (12%)	16.2

Table 6.8: Effect of intersecting S_{C_2} with S_{C_3}, as applied to a representative dictionary of clusters gathered from 15 users. Results indicate that the DIAG pattern produces the smallest dictionary, and still guesses a relatively large number of passwords.

The results shown in Table 6.8 indicate that, on average for the *pool* image, optimizing with S_{C_3}-DIAG will reduce the dictionary size to 16 bits, while still guessing 12% of passwords. Similarly, for the *cars* image, optimizing with only S_{C_3}-DIAG will reduce the dictionary to 18 bits, while still guessing 10% of passwords. We highlight that since data from only 15 users are used, the results for $P^{V_{15}}$ are closer to $P^{R_{15}}$, as there are fewer clusters. We explore the effect of using all u users in Figures 6.7 and 6.8 below.

When clusters are used, we can place an ordering on the intersected Class C_2 (P^{V_u}) and Class C_3 dictionary such that the passwords are ordered from most to least probable (as defined by the product of the probabilities of each cluster in the password). We examine the results of applying this ordering of $P^{V_u} \cap S_{C_3}$ for each image, as shown in Figure 6.7 and 6.8. Recall Section 6.2.4 (discussion of Class C_3 dictionaries) for a description of each of the click-order patterns shown.

One may find it counter-intuitive that using all u users as opposed to 15 in Table 6.8 results in lower success for some of the click-order patterns in S_{C_3}. However, it is sensible when we consider that the more click-points used to generate S_{C_2}, the larger the clusters, and the more likelihood that some points will be "lost" due to their belonging to another cluster. Although the clustering algorithm provides an overall

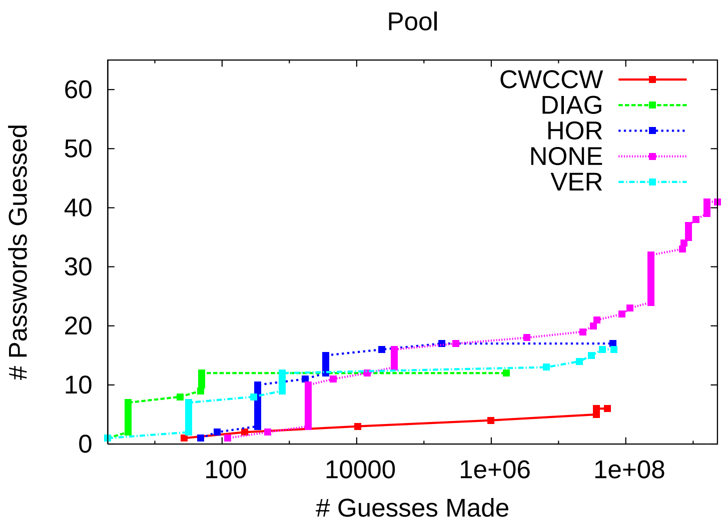


Figure 6.7: CDF of $S_{C_2} \cap S_{C_3}$ as applied to the 114 field study users for *pool*. Each click-order pattern in the legend shown is abbreviated (normally has a preceding S_{C_3}-). NONE is the human-seeded attack, without any click-order pattern applied.

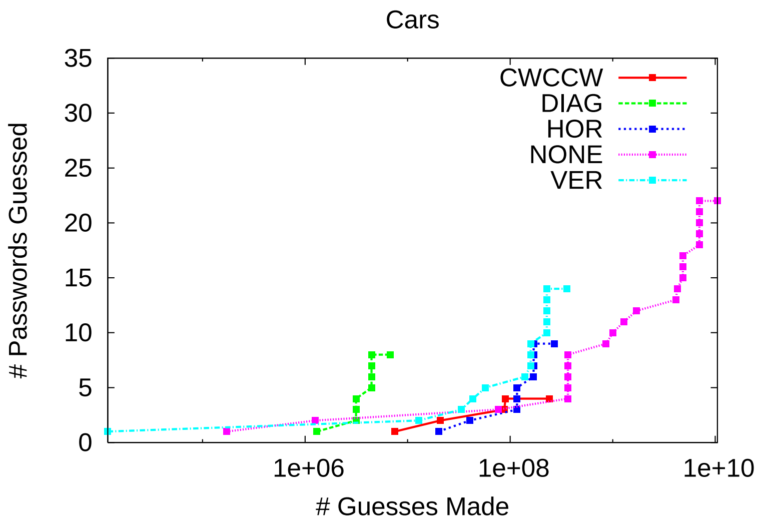


Figure 6.8: CDF of $S_{C_2} \cap S_{C_3}$ as applied to the 109 field study users for *cars*. Each click-order pattern in the legend shown is abbreviated (normally has a preceding S_{C_3}-). NONE is the human-seeded attack, without any click-order pattern applied.

dictionary size reduction, it does not necessarily guess as many passwords, and this effect increases with more data. For example, $P^{R_u} \cap S_{C_3}$-DIAG is 27 bits and guesses 18% of passwords for *cars*, meaning that the clustering reduces the efficacy of the dictionary by half, but with a dictionary that is less than one-sixteenth of the size.

Overall, the results show that intersecting S_{C_2} with the S_{C_3} dictionaries (except $CWCCW$) provides better performance than the S_{C_2} dictionary alone (at least initially). The effect is more striking for *pool* than for *cars*; for example, in Fig. 6.7 we see the $S_{C_2} \cap S_{C_3}$-DIAG dictionary guesses the first 7 passwords (6% of the total) within 5 guesses. In general for *pool*, Fig. 6.7 shows that all $S_{C_2} \cap S_{C_3}$ dictionaries (except $CWCCW$) perform better than S_{C_2} alone initially, but by the time they are exhausted, the performance is better for S_{C_2} alone. For *cars*, Fig. 6.8 shows that all $S_{C_2} \cap S_{C_3}$ dictionaries perform better than S_{C_2} alone (except for the first three correct guesses).

6.5.6 Limitations of User Studies

As with all user studies, it is important to discuss possible limitations. There are differences between the amount of clustering seen in our lab study and our field study for the *cars* image, but the clustering for the *pool* image is quite similar in each study. Using "leave-out-k" experiments on both the lab and field studies also indicates some differences between the two studies.

Our first leave-out-k experiment was using only the field study data. We generated a P^R dictionary using a small set of field study user passwords to attack the remaining field study user passwords. We use P^R to ensure we do not incur any information loss (which occurs to some extent when clustering is used). The result, in Table 6.9, shows a difference between the lab study and the field study (final) passwords, as using actual field study passwords to generate a P^R dictionary had much higher success rates (13-24% more) than when 20 users from the lab study were used to generate a P^R dictionary (cf. Table 6.6).

Our second leave-out-k experiment was using only the lab study data. We use 10 randomly selected sets of 25 users from the lab study to generate both P^R and P^V against the remaining 8-10 lab study users. The attack appeared to work similarly to

Dictionary	*cars*			*pool*		
	m	bitsize	# passwords guessed	m	bitsize	# passwords guessed
$P^{R_{20,longterm}}$	100	33.1	29/89 (33%)	100	33.1	52/94 (55%)
$P^{R_{10,longterm}}$	50	27.9	23/99 (23%)	50	27.9	22/104 (21%)

Table 6.9: Dictionary attack results, using the first 20 and 10 users from the long term study to seed an attack against the others. m is the alphabet size. See text for descriptions of P^R.

when applied to the field study for *pool*, but not for *cars*: the average percentage of guessed lab study passwords for *pool* is 28% using $P^{R_{25}}$ and 20% using $P^{V_{25}}$ (about 9% less than the results in Table 6.6), but no passwords were guessed for *cars*. Although not statistically significant due to the small sample size used for testing, these results may also indicate differences, for some images, between the passwords selected by the lab study and field study user's final passwords.

Here we discuss possible reasons for these differences. One possible reason for the differences in user choice between the two studies is that the field study users may not have been as motivated as the lab study users to create "difficult to guess" graphical passwords. It is unclear how a user might measure whether they are creating a graphical password that is difficult to guess, and whether in trying, if users would actually change their password's strength; one study [134] shows that only 40% of users actually change the complexity of their text passwords according to the security of the site. Another equally possible explanation might be that the lab study users chose more difficult passwords than they would have in practice, as they were aware there was no requirement for long term recall, and also did not have a chance to forget and subsequently reset their passwords to something more memorable. Further, it is possible that the user's task focus in the lab study had an influence, such that they were more motivated to create a more complex password than they might be in a regular usage environment. With our current data, it seems unlikely that we can conclusively determine a reason for these differences.

Despite these differences, we have demonstrated that there is still enough similarity between the two groups to launch effective Class C_2 attacks.

6.6 Other (Non-Predictive) Attack Models

In Chapters 4, 5, and in this chapter, we found sets of weak password subspaces by defining and applying predictive models. However, this raises the question of whether there might be other undiscovered weak password subspaces (in the general sense). Showing a weak password subspace exists defines a vulnerability, but it does not prove that all passwords outside of the weak password subspace do not belong to other (unknown) weak subspaces. We discuss this concept here, and another attack model that, in some cases, might lead to a better understanding of the distribution of passwords for a given scheme.

The key to a predictive model is that it is generable; if there is a pattern in user choice, and passwords that exhibit that pattern cannot be generated in a known way, then this pattern simply cannot be exploited. If patterns are not predicted and subsequently tested for, then how do we find exploitable structure or patterns? One method that has been used by password cracking software (e.g., [106]) is to use Markov models of language (under the predictive assumption that users will choose passwords from their language), and to generate passwords using bi- or tri-grams from that language, ordered by decreasing probability. To effectively train a Markov model, we need enough data to capture the frequency of each pattern, and the data must be as close as possible to user choice in practice. Thus, an ideal candidate for training is a large, clear-text password database.

A variation of this method was used by Davis et al. [34] to determine (for certain sex/race groups) which sequences of images users were more likely to select for the Faces and Story schemes (Faces is a variation of PassFaces; recall Section 3.2.2). They create bi-grams using a training data set containing 80% of their collected user passwords. A bi-gram in this case is an ordered pair of two images from at least one user password. The assumption in the bi-grams model is that each image is dependent upon the image chosen in the previous panel. They use those bi-grams to regenerate passwords, and created a dictionary ordered by decreasing probability as mainly defined by the bi-gram's frequencies. They further created an ordering of the entire password space, such that those passwords without representative bi-grams in the training set of passwords are included in the dictionary. They found that 25% of passwords for Faces could be guessed in 13 guesses, and 25% of passwords for Story could be guessed in 113 guesses. For such an attack to be effective, there must be sufficient data to effectively create enough bi-grams and approximate their probability of occurring. Intuitively, it would make sense that the amount of data required would increase when there are fewer (and less severe) patterns in user choice.

Although it might seem that this type of attack would define all patterns in user choice, we caution that this may not be the case, as it could easily depend on the scheme and severity of patterns in user choice. It is not clear how well this attack would work against schemes that do not have such dramatic patterns emerging. For example, the DAS scheme might not have many users choosing the exact same strokes (or sequences of strokes) in their password. In this case, it is not clear how well this sort of attack would work, and whether it is simply a matter of collecting more user passwords for training. It is possible that in cases where the patterns are more abstract for the set of training passwords, that a predictive model could work better.

For a comparative example of this method with our predictive methods on one scheme, we detail its application to the PassPoints scheme in Section 6.6.1. We show that this method produces a very effective attack, guessing 8-11% of user passwords in 3 guesses, which leads us to question the security of PassPoints even in environments where online attack is the only assumed threat.

6.6.1 Using Markov Models to Attack PassPoints

To use a Markov model to attack PassPoints, we assume that each click-point in a g_{cb}-password only depends on the previous click-point. To capture this dependency, we create bi-grams based on subsets of the passwords collected in the field study. In the case of g_{cb}-passwords, a bi-gram is an ordered pair of click-points, and each 5-click password will produce four bi-grams. We further assume that bi-grams are more likely to occur at the specific positions in which they were observed within the training passwords; for example, if $[(100, 100), (200, 200)]$ was only observed as the first bi-gram in a password, we assume that it is more likely that it will occur as the first two points in a password than in any other position. Thus, we include counts of the observed bi-gram positions in our training (i.e., each bi-gram would be observed in at least one of the four possible bi-gram positions).

We perform the following method thirty times, for a given image:

- Select a random subset B of 80% of user passwords from our field study.[5]

- Create position-aware, normalized bi-grams as follows. Using each password $p \in B$: (1) calculate clusters as discussed in Section 6.3.2, (2) for each $p \in B$, normalize each of the five click-points in the password to the cluster it belongs to, and (3) split each normalized password into four bi-grams (i.e., $[(x_1, y_1), (x_2, y_2)], [(x_2, y_2), (x_3, y_3)], [(x_3, y_3), (x_4, y_4)], [(x_4, y_4), (x_5, y_5)]$).

- Each position-aware, normalized bi-gram has a set of 5 counts: one for total frequency, and four position frequencies (i.e., one frequency count for each of four possible observed bi-gram positions).

- Generate passwords based on the bi-grams created in the above steps. We only use the bi-gram if the frequency count at that position is greater than zero (i.e., it was observed to occur at that specific password position at least once). The probability of a generated password is based on the product of the frequencies of each bi-gram at its position within the generated password.

- Sort generated passwords by decreasing probability, and use the sorted order to guess the remaining 20% of passwords.

Results Using Field Study Data

From the thirty trial runs, we average the number of passwords guessed after making 1, 2, 3, 4, 5, 10, 50, 100, 150, 200, 500, 5000, and 10000 guesses. Our results are plotted in Figure 6.9.

We note that the limitations discussed in Section 6.5.6 should also be taken into consideration when interpreting these results. In general, our results shown in Figure 6.9 demonstrate that there are other probable patterns than are captured by considering the cluster probabilities alone (as in Class C_2), or the simple click-order patterns

[5]This could be done with any password data set, but in this case, we use those from our field study.

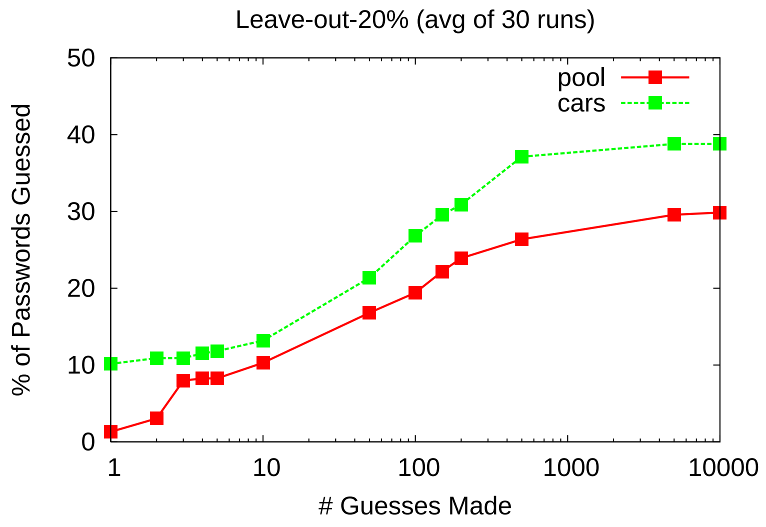


Figure 6.9: CDF for applying a Markov model to attack the PassPoints scheme.

(as in Class C_3). Clearly, finding a handful of weak password subspaces does not necessarily imply that there are no more left to find, and/or that the existing weak password subspaces could be smaller and more significant (as defined in properties 1 and 2 from Definition 1). In effect, this method demonstrates that there are more patterns to be found, although it is unclear at this point how they can be characterized without having access to a large clear-text password database as we do from our field study. However, the efficacy of such an attack should be seriously considered, as it still represents how well an attacker *could* do with another (not yet known) dictionary-generation technique.

6.7 Hot-Spots Computed for Click-Based Graphical Passwords in Lab and Field Studies

Here we describe further details concerning the hot-spots found for click-based graphical passwords in both the lab and field studies (recall Sections 6.3.2 and 6.5.1). This section is intended to discuss related material of interest that is not required to understand the attacks presented in this chapter. Section 6.7.1 discusses the clustering found on the 17 images from the lab study, and Section 6.7.2 discusses the clustering found on the two images from the field study.

6.7.1 Hot-Spots Computed from Lab Study Results

We collected data from a short-term in-lab study of 43 users as described in Section 6.3.2, and used a clustering algorithm (recall Section 6.3.2) to determine a set V of (non-empty) clusters and their sizes. To begin comparing the 17 images studied, Figure 6.10 shows the sizes of the top five most popular clusters, and the total number of popular clusters.

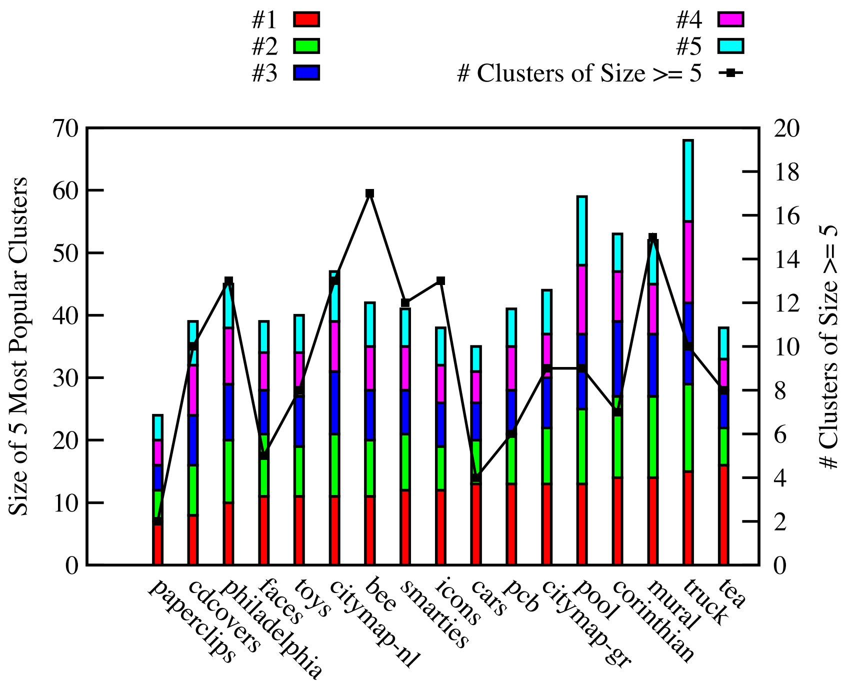


Figure 6.10: The five most popular clusters (in terms of size, i.e., # of times selected), and # of popular clusters (of size ≥ 5). Results are from 32-40 users, depending on the image, for the final passwords created on each image. For *pcb*, which shows only 6 clusters of size ≥ 5, the size of clusters 2-5 become 5, 5, 4, and 3 when counting at most one click from each user.

Given the cluster sizes, we then calculate the observed "probability" p_j (based on our user data set) of the cluster j being clicked, as cluster size divided by total clicks observed. When the probability p_j of a certain cluster is sufficiently high, we can place a confidence interval around it for future populations (of users who are similar in background to those in our study) using formula (6.1) as discussed below.

Each probability p_j estimates the probability of a cluster being clicked for a *single* click. For 5-click passwords, we approximate the probability that a user chooses cluster j in a password by $5p_j = 5 \times p_j$. Note that the probability for a cluster j increases slightly as other clicks occur (due to the constraint of 5 distinct clusters in a password); we ignore this in our present estimate.

Our results in Figure 6.10 indicate a significant number of hot-spots for our sample of the full population ($32 - 40$ users per image). Previous "conservative" assumptions [164] were that half of the available alphabet of T-regions would be used in practice – or 207 in our case. If this were the case, and all T-regions in the alphabet were equi-probable, we would expect to see some clusters of size 2, but none of size 3 after 40 participants; we observed significantly more on all 17 images. Figure 6.10 shows that some images were clearly worse than others. There were many clusters of size at least 5, and some as large as 16 (see *tea* image). If a cluster in our lab study received 5 or more clicks – in which case we call it a *popular* or *high-probability* cluster – then statistically, this allows determination of a confidence interval, using Formula (6.1) which provides the $100(1-\alpha)\%$ confidence interval for a population proportion [37, page 288].

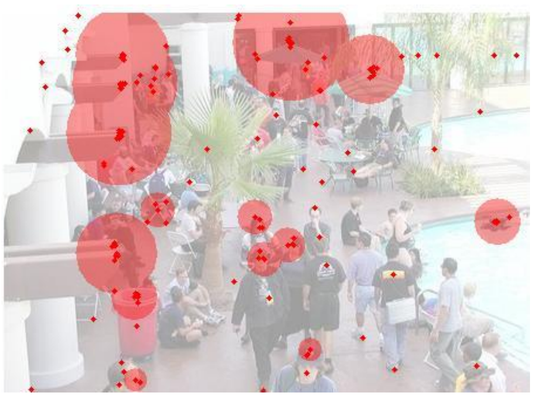
$$p \pm z_{\alpha/2}\sqrt{\frac{pq}{n}} \tag{6.1}$$

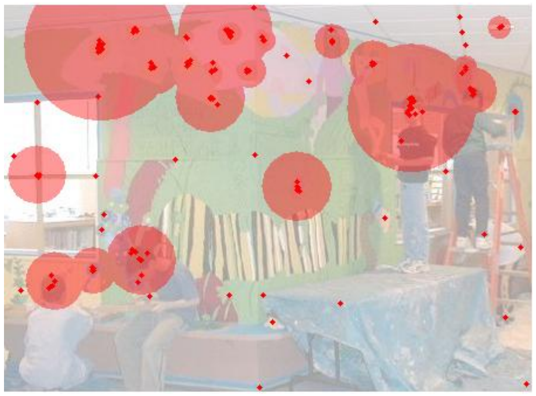
Here n is the total number of clicks (i.e., five times the number of users), p takes the role of p_j, $q = 1 - p$, and $z_{\alpha/2}$ is from a z-table. A confidence interval can be placed around p_j (and thus $5p_j$) using (6.1) when $np \geq 5$ and $nq \geq 5$. For clusters of size $k \geq 5$, $p = \frac{k}{n}$, then $np = k$ and $nq = n - k$. In our case, $n \geq 32 \cdot 5$ and $n - k \geq 5$, as statistically required to use (6.1).

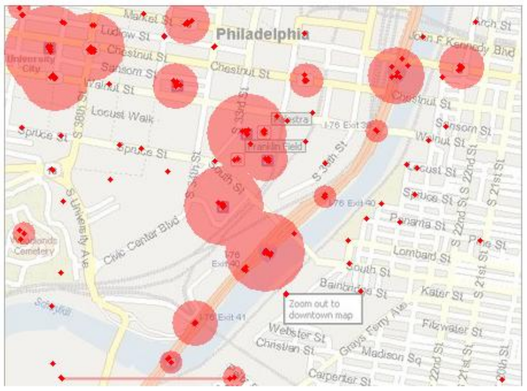
Table 6.10 shows these confidence intervals for four images, predicting that in future similar populations many of these points would be clicked by 10-50% of users, and some points would be clicked by 20-60% of users with 95% confidence ($\alpha = .05$[6]). For example, in Table 6.10(a), the first row shows the highest frequency cluster (of size 13); as our sample for this image was only 35 users, we observed 37.1% of our

[6]This is *not* the same α as used in Definition 1, but is the significance level used to compute the confidence level.

participants choosing this cluster as part of their password. Using (6.1), between 17.7% and 56.6% of users from future populations are expected to choose this same cluster (with 95% confidence).


(a) *pool* (originally from [163, 164]).


(b) *mural* (originally from [163]).


(c) *philadelphia* (originally from [163]).

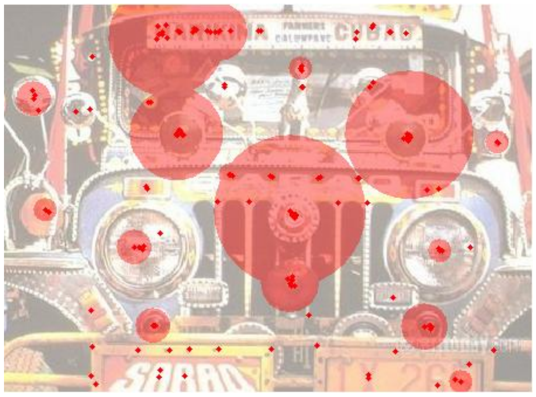

(d) *truck* (originally from [49]).

Figure 6.11: Observed click-points from lab study. Halo diameters are 10 times the size of the underlying cluster, illustrating its popularity.

Figure 6.10 and Table 6.10 show the popularity of the hottest clusters; Figure 6.10's line also shows the number of popular clusters. The clustering effect evident in Figures 6.10, 6.11, and Table 6.10 clearly establishes that hot-spots are very prominent on a wide range of images. We further pursue how these hot-spots impact the practical security of full 5-click passwords in Section 6.3.2. As a partial summary, our results suggest that many images have significantly more hot-spots than would be expected if all T-regions were equi-probable. The *paperclips*, *cars*, *faces*, and *tea* images are not as susceptible to hot-spotting as others (e.g., *mural*, *truck*, and *philadelphia*). For example, the *cars* image had only 4 clusters of size at least 5, and only one with

(a) *pool* image			(b) *mural* image		
Cluster size	$5p_j$	95% CI ($5p_j$)	Cluster size	$5p_j$	95% CI ($5p_j$)
13	0.371	(0.177; 0.566)	14	0.400	(0.199; 0.601)
12	0.343	(0.156; 0.530)	13	0.371	(0.177; 0.566)
12	0.343	(0.156; 0.530)	10	0.286	(0.114; 0.458)
11	0.314	(0.134; 0.494)	8	0.229	(0.074; 0.383)
11	0.314	(0.134; 0.494)	7	0.200	(0.055; 0.345)

(c) *philadelphia* image			(d) *truck* image		
Cluster size	$5p_j$	95% CI ($5p_j$)	Cluster size	$5p_j$	95% CI ($5p_j$)
10	0.286	(0.114; 0.458)	15	0.429	(0.221; 0.636)
10	0.286	(0.114; 0.458)	14	0.400	(0.199; 0.601)
9	0.257	(0.094; 0.421)	13	0.371	(0.177; 0.566)
9	0.257	(0.094; 0.421)	13	0.371	(0.177; 0.566)
7	0.200	(0.055; 0.345)	13	0.371	(0.177; 0.566)

Table 6.10: 95% confidence intervals for the top 5 clusters found in each of four images. The confidence intervals are for the percentage of users expected to choose this cluster in future populations.

frequency at least 10. The *mural* image had 15 clusters of size at least 5, and 3 of the top 5 frequency clusters had frequency at least 10. Given that our sample size for the *mural* image was only 36 users, these clusters are suprisingly popular. This demonstrates the range of effect the background image can have (for the images studied).

Although previous work [163] suggests using intuition for choosing more secure background images (no further detail was provided), our results apparently show that intuition is not a good indicator. Of the four images used in other click-based graphical passwords studies, three showed a large degree of clustering (*pool*, *mural*, and *philadelphia*). Furthermore, two other images that we "intuitively" believed would be more secure background images were among the worst (*truck* and *citymap-nl*). The *truck* image had 10 clusters of size at least 5, and the top 5 clusters had frequency at least 13.

Finding reliable automated predictors of more secure background images remains an unsolved problem. Our work to determine whether simple measures (image segmentation, corner detection, and image contrast measurement) correlate with the amount of observed hot-spotting (as measured by number of clusters observed divided by the number of points collected) is shown in Figures 6.12, 6.13, and 6.14. The graphs indicate that these measures do not appear to offer reliable indicators of hot-spotting. Image contrast was measured using by (1) converting the image to grayscale using Matlab's *rgb2gray* function, and (2) obtaining the contrast using *graycoprops* Matlab function with every surrounding pixel considered in the offsets parameter. Contrast is 0 for a constant image. The number of corners was measured using Harris corner detection function as implemented by Kovesi [79].[7] Number of segments was measured using graph-based image segmentation [46] with $\sigma = 0.5$, $k = 300$, and $min = 16$.[8]

We would have expected that higher contrast images, images with more corners, and/or images with more segments would have had less observed hot-spotting (i.e.,

[7]As *harris(image,1, 1000, 3)*

[8]min defines the minimum segment size, k influences the resulting segment sizes (larger values favor larger segments), and $\sigma = 0.5$ is the recommended value used for smoothing the image with a Gaussian filter prior to segmentation.

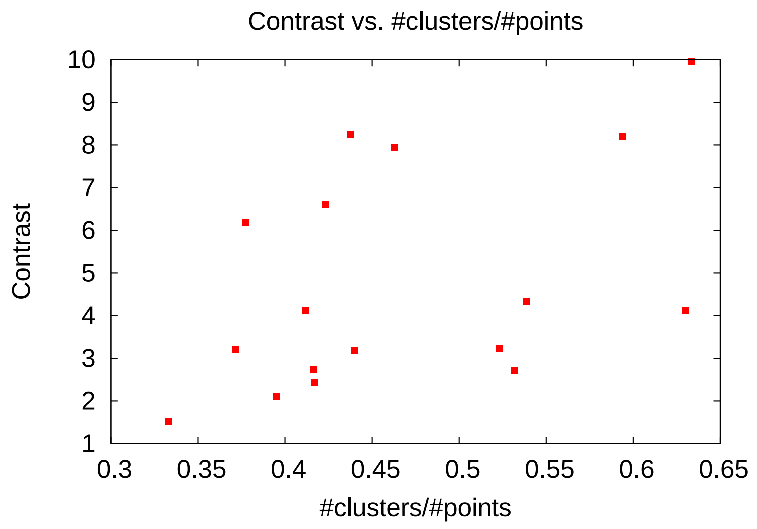


Figure 6.12: Correlation of lab study hot-spotting with image contrast.

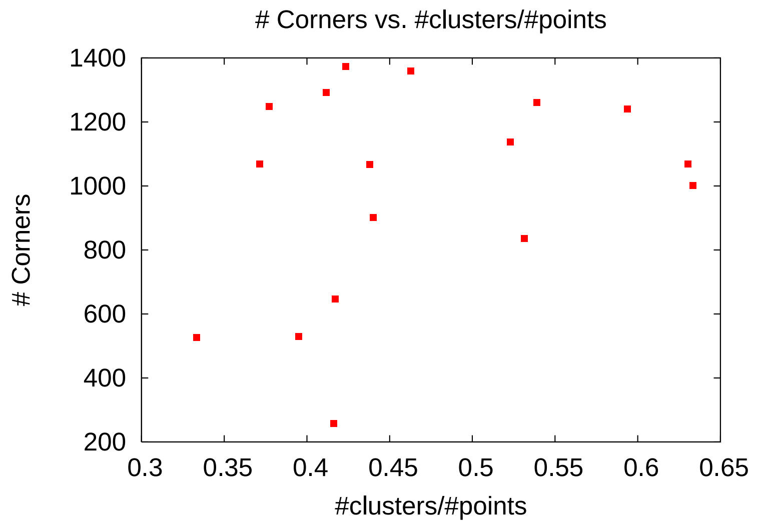


Figure 6.13: Correlation of lab study hot-spotting with number of corners.

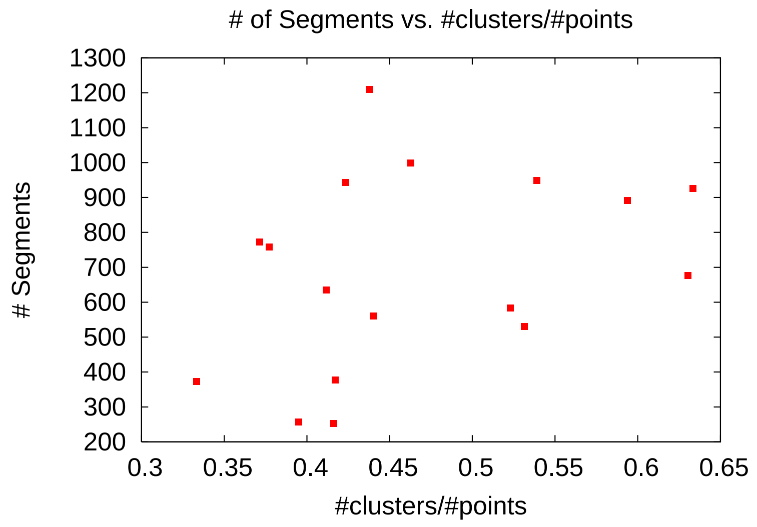


Figure 6.14: Correlation of lab study hot-spotting with number of segments.

would have values closer to one). If this were the case, Figures 6.12, 6.13, and 6.14 would have shown points that would appear to be points on a positive slope. However, the points are quite scattered, indicating a weak correlation between each of these three measures and the observed hot-spotting.

Thus, we next explore the impact of hot-spotting across images to help choose two images for further analysis.

Measurement and Comparison of Hot-Spotting for Different Images

To compare the relative impact of hot-spotting on each image studied, we calculated two formal measures of password security for each image: entropy $H(X)$, in equation (6.2), and in equation (6.3), the expected number of guesses $E(f(X))$ to correctly guess a password assuming the attacker knows the probabilities $w_i > 0$ for each password i. The relationship between $H(X)$ and $E(f(X))$ for password guessing is discussed by Massey [90]. Of course in general, the w_i are unknown, and our study gives only very coarse estimates; nonetheless, we find it helpful to use this to develop an estimate of which images will have the least impact from hot-spotting. For (6.2) and (6.3), n is the number of passwords (of probability > 0), random variable X ranges over the passwords, and $w_i = Prob(X = x_i)$ is calculated as described below.

$$H(X) = -\sum_{i=1}^{n} w_i \cdot log(w_i) \tag{6.2}$$

$$E(f(X)) = \sum_{i=1}^{n} i \cdot w_i \quad \text{, where } w_i \geq w_{i+1}\text{, and} \tag{6.3}$$

$f(X)$ is the number of guesses before success.
We calculate these measures based on our observed user data. For this purpose, we assume that users will choose from a set of click-points (following the associated probabilities), and combine 5 of them randomly. This assumption almost certainly over-estimates both $E(f(X))$ and $H(X)$ relative to actual practice, as it does not consider click-order patterns or dependencies. Thus, popular clusters likely reduce security by more than we estimate here.

We define P^V to be the set of all 5-permutations derivable from the clusters observed in our user study (as computed in Section 6.3.2). Using the probabilities p_j of each cluster, the probabilities w_i of each password in P^V are computed as follows. Pick a combination of 5 observed clusters $j_1, \ldots, j_5$ with respective probabilities $p_{j1}, \ldots, p_{j5}$. For each permutation of these clusters, calculate the probability of that permutation occurring as a password. Due to our lab study instructions that no two click-points in a password can fall in the same T-region, these probabilities change as each point is clicked. Thus, for password $i = (j_1, j_2, j_3, j_4, j_5)$, $w_i = p_{j1} \cdot [p_{j2}/(1 - p_{j1})] \cdot [p_{j3}/((1 - p_{j1}) \cdot (1 - p_{j2}))] \cdot \ldots)$.

The resulting set P^V is a set of click-based graphical passwords (with associated probabilities) that coarsely approximates the effective password space if the clusters observed in our user study are representative of those in larger similar populations. We can order the elements of P^V using the probabilities w_i based on our user study. An ordered P^V could be used as the basis of an attack dictionary; this ordering could be much improved, for example, by exploiting expected patterns in click-order. See Section 6.3.3 for more details.

For comparison to previous "conservative" estimates that simply half of the available click-points (our T-regions) would be used in practice [164], we calculate P^U as defined below. We compare to P^U as it is a baseline that approximates what we would expect to see after running 32 users (the lowest number of users we have for any image), if previous estimates were accurate, and T-regions were equi-probable. P^U is the set of all permutations of clusters we expect to find after observing 32 users, assuming a uniformly random alphabet of size 207.

Fig. 6.15 depicts the entropy and expected number of guesses for P^V. Notice the range between images, and the drop in $E(f(X))$ from P^U to values of P^V. Comparison to the marked P^U values for (1) $H(X)$ and (2) $E(f(X))$ indicates that previous rough estimates are a security overestimate for practical security in all images, some much more so than others. This is at least partially due to click-points not being equi-probable in practice (as illustrated by hot-spots), and apparently also due to the previously suggested effective alphabet size (half of the full alphabet) being an over-estimate. Indeed, a large alphabet is a big part of the theoretical security advantage that these graphical passwords have over text passwords. If the effective alphabet size is not as large as previously expected, or is not well-distributed, then we should reduce our expectations of the security.

These results appear to provide fair approximation of the entropy and expected number of guesses for the larger set of users in the field study; we performed these same calculations again using the field study data, with the following results. For both of the two images, the entropy measures were within one bit of values measured here (less than a bit higher for *pool*, and about one bit lower for *cars*). The number of expected guesses increased for both images (by 1.3 bits for *cars*, and 2.5 bits for *pool*).

The variation across all images shows how much of an impact the background image can have, even when using images that are "intuitively" good. For example, the image that showed the most impact from hot-spotting was the *mural* image, chosen for an earlier PassPoints usability study [163]. We note that the *paperclips* image scores best in the charted security measures (its $H(X)$ measure is within a standard deviation of P^U); however, 8 of 36 users who created a password on this image could not perform the subsequent login (and skipped it – as noted earlier), so the data for this image represents some passwords that are not repeatable, and thus we suspect it would have lower relative security in practice.

Overall, one can conclude that image choice can have a significant impact on

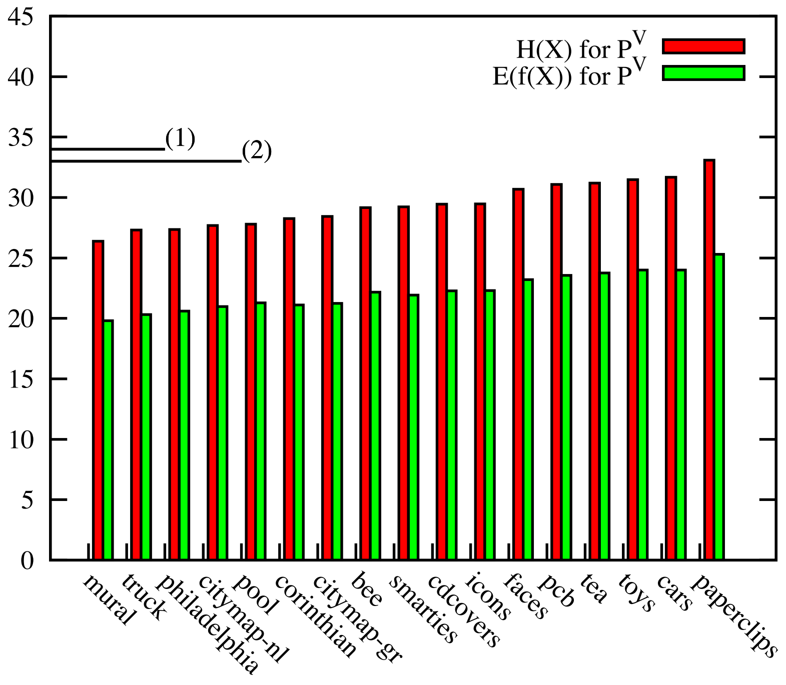


Figure 6.15: Security measures for each image (in bits). P^V is based on data from lab user study of 32–40 passwords (depending on image). For comparison to a uniform distribution, (1) marks $H(X)$ for P^U, and (2) marks $E(f(X))$ for P^U.

the resulting security, and that developing reliable methods to filter out images that are the most susceptible to hot-spotting would be an interesting avenue for future research.

We used these formal measures to make an informed decision on which images to use for our field study. Our goal was to give the PassPoints scheme the best chance (in terms of anticipated security) we could, by using one image (*cars*) that showed the least amount of clustering (with the best user success in creating a password), and also using another that ranked in the middle (*pool*) that was also used in previous PassPoints studies.

6.7.2 Field Study Hot Spots and Relation to Lab Study Results

Here we present the clustering results from the field study, and compare results to those on the same two images from the lab study. Fig. 6.5b shows that the areas that were emerging as hot-spots from the lab study (recall Fig. 6.11a) were also popular in the field study, but other clusters also began to emerge. Fig. 6.5a shows that even our "best" image from the lab study (in terms of apparent resistance to clustering) also exhibits a clustering effect after gathering 109 passwords. Table 6.11 provides a closer examination of the clustering effect observed.

Image Name	Size of most popular clusters					# clusters of size ≥ 5
	# 1	# 2	# 3	# 4	# 5	
cars	26	25	24	22	22	32
pool	35	30	30	27	27	28

Table 6.11: Most popular clusters (field study).

These values show that on *pool*, there were 5 points that 24-31% of users chose as part of their password. On *cars*, there were 5 points that 20-24% of users chose as part of their password. The clustering on the *cars* image indicates that even highly detailed images with many possible choices have hot spots. Indeed, we were surprised to see a set of points that were this popular, given the small amount of observed clustering on this image from our smaller lab study.

The prediction intervals calculated from our lab study (recall Section 6.3.2) provide reasonable predictions of what we observed in the field study. For *cars*, the prediction intervals for 3 out of the 4 popular clusters were correct. For *pool*, the prediction intervals for 8 out of the 9 popular clusters were correct. The anomalous cluster on *cars* was still quite popular (chosen by 12% of users), but the lower end of the lab study's prediction interval for this cluster was 20%. The anomalous cluster on *pool* was also still quite popular (chosen by 18% of users), but the lower end of the lab study's prediction interval for this cluster was 19%.

Our studies allow us to update previous assumptions that half of all T-regions on an image will be chosen by users. After collecting 570 and 545 points, we only observed 111 and 133 unique clusters (for *pool* and *cars* respectively); thus, one quarter to one third of all T-regions would be a more reasonable estimate even from highly detailed images, and the relative probabilities of these regions should be expected to vary quite considerably.

6.8 Conclusion

Our predictive models identified two weak password subspaces for g_{cb}-passwords: Class C_2 (31.1-33.4 bits) and Class C_3-DIAG (32.6 bits). These two subspaces were found to correctly guess 20-36% and 26-46% of g_{cb}-passwords respectively from our large scale field study. Further, we found that combining Class C_2 and Class C_3-DIAG produces dictionaries of 11 bits fewer in size, while still guessing a significant number of passwords (8-10%).

Our non-predictive attack model (using markov models, recall Section 6.6.1) was found to correctly guess 8-11% of passwords from our large scale field study in 3 guesses. This result makes it difficult to recommend the use of PassPoints in any environment, as it is apparently vulnerable to this attack even if account lockout is set after 3 incorrect guesses. This allows us to update our recommendations, such that user choice not be allowed in PassPoints, without some way to mitigate the effects of hot-spotting and dependencies between click-points.

We present a set of tentative recommendations for reducing the impact of our predictive attacks in Sections 6.8.1 and 6.8.2.

6.8.1 Password Rules

Given our knowledge of weak password subspaces for PassPoints (based on our predictive models), we suggest the following as an initial set of g_{cb}-password rules.

1. Disallow passwords that follow the DIAG click-order pattern.

2. Disallow passwords with any click-point on a hot-spot that occurs with probability $\geq \alpha$ (from property 2, Definition 1), as defined by human computation.

The rationale for using α as the maximum probability for a hot-spot, is that it appears there are dependencies between a user's 5 click-points (as indicated by the high success rates of both S_{C_3}-DIAG and the markov model attacks described in Section 6.6). The worst-case scenario is then that each hot-spot observed in a human-computated data set will be part of a 5-click sequence that occurs with that same probability.

We believe that the first rule should be fairly straightforward to implement; however, implementing the second rule may be more difficult communicate (that a certain area is disallowed). For example, any method to visually blot out the regions must be studied to ensure it does not create usability problems, or new patterns related to the disallowed points. An interesting avenue for future work would be to determine whether graphical password users create other predictable patterns when their choices are disallowed by proactive checking.

Although these rules will help prevent our S_{C_1} and S_{C_3}-DIAG attacks, we caution that there is no reason to believe they will help protect against our non-predictive attack (recall Section 6.6.1).

6.8.2 Implementation Enhancements

Implementation enhancements for text-based passwords can also be applied to graphical passwords. PassPoints using robust discretization [13], can be stored using a one-way hash and could thus benefit from hashing algorithms with an adaptable cost (e.g., see [128]) and/or that use *password stretching* or repeated hashing of passwords (e.g., see [59]) to increase the computational cost of guessing attacks. Also, the user can specify how long each guess should take by using Boyen's halting puzzles [18], although a usable integration of this concept into the system may prove more of a challenge (recall discussion in Section 2.3.2). "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 the computational cost of guessing attacks against a set of users.

Our implementation of Itti's [67] visual attention model may be used proactively to determine background images to avoid, as those images on which the visual attention model performed well (e.g., identifies some areas as much more interesting than others) appear more vulnerable to the Class C_1 image processing attacks from Section 6.3.1. Unfortunately, filtering images using this method would not necessarily offer protection against the Class C_2 human-seeded attacks.

The theoretical password space could be increased by changing a number of parameters such as increasing the number of click-points used, the image size, and "zooming in" [14, 71]. Whether this method achieves acceptable usability and security requires further study. One exception is reducing the error tolerance of each click-point: this would increase the size of the full password space, and usability studies on the issue indicate that an error tolerance T-region of 9×9 pixels would be reasonable [24].

Finally, Cued Click-Points (CCP) [25] presents a sequence of 5 images on which the user clicks each once. While CCP does not increase the full password space, it increases the difficulty of generating an attack, as an attacker would need to obtain and analyze *at least* five images instead of a single one (for more details, see Section 3.2.2).

Chapter 7

Pass-thoughts: a Preliminary Proposal

7.1 Introduction

With the goal of identifying a promising direction in user authentication that might be more immune to attack, in this chapter we discuss a preliminary proposal for a new idea that we call “pass-thoughts”. This idea is now also being actively pursued by other groups [35].

The goal of a pass-thoughts system would be to extract as much entropy as possible from a user’s brain signals upon “transmitting” a thought using a brain-computer interface (BCI). A pass-thoughts system might prove to have some interesting characteristics (but these are not proven herein). Provided that these brain signals can be recorded and processed in an accurate and repeatable way, a pass-thought system might provide a quasi two-factor, non-static, authentication method resistant to shoulder-surfing. The potential size of the space of a pass-thought system would seem to be unbounded in theory, although in practice it will be finite due to system constraints.

We discuss the motivation for pass-thoughts in Section 7.2, the basic concept in Section 7.3, its potential in Section 7.4, a few example deployment scenarios in Section 7.5, discussion of security in Section 7.6, general discussion in Section 7.7, and required next steps in Section 7.8.

7.2 Motivation

As discussed in Chapter 2, text passwords have many well-known limitations. While some proposed replacements for passwords do not have the same limitations, most schemes have other limitations or requirements as discussed herein. For example, static biometric systems rely upon unchanging features that have a lifetime as long as the individual; this characteristic, combined with the threat of theft of biometric templates leaves static biometrics (on their own) unsuitable for remote authentication. In contrast, smart cards can be used to securely authenticate users to remote servers, but at the cost and inconvenience of per-user hardware tokens.

Now more than ever, shoulder-surfing is a problem for passwords. The ubiquity of cell phone cameras and wireless video cameras brings a new ever-present threat upon us: recorded shoulder-surfing. As users, we can no longer simply look over our shoulder to be aware of an adversary observing our password. To combat such a threat, we require an authentication method that is unobservable under any circumstance.

Shoulder surfing can be addressed in the context of "what you know" authentication methods, for example through the use challenge-response protocols (recall Section 3.2.2).[1] These methods require cognitive processing by the user for each bit of information, and thus unfortunately would require a large amount of a user's input time to increase the size of the password space. Although they are unlikely to become a ubiquitous authentication scheme for that reason, they may be useful in some high-security settings.

A user may also pose a threat to a "what you know" authentication system by writing down their password, or sharing it with others. Both textual and some types of graphical passwords are susceptible to this threat. Users often share passwords to bypass the administrative overhead of setting up the proper group access control. Sharing passwords may be more convenient for the user; however, it defeats the purpose of user authentication. The risk of writing a password down can be minimized by storing it in a physically secure location and shredding it upon disposal; otherwise without these cautions, writing a password down makes it available to anyone with physical access (e.g., co-workers, cleaners, and dumpster-divers all have access). When a user can describe their password to others, the user is also susceptible to social engineering methods whereby the user is tricked by an adversary into providing their password [57]. Thus, we currently have a paradox: we want a scheme whereby we *can* write a password down as a reminder, yet we want to ensure the password cannot be used by others.

[1]Although for at least one scheme (Weinshall's protocols [165]), observing a number of login sessions allows for recovery of the password (recall Section 3.3.5).

The susceptibility of "what you know" authentication methods to guessing attack is largely created by patterns in user choice. Authentication schemes that are based on "what you have" (physical tokens, such as smart cards [1]) have the advantage of high entropy, and if they are lost or stolen, they can be changed and replaced. However, they are not necessarily always in the owner's possession. During the period before a user's lost token(s) is revoked, their system and the information on it may be vulnerable [74]. An idea to solve this problem is to have users wear their physical token, as in *zero-interaction authentication* (ZIA) [27]. ZIA is an authentication scheme that provides security against physical attacks by continuously polling the token to ensure it (and thus presumably the user) is present. ZIA appears to be a useful scheme to protect against physical attacks, particularly for mobile devices. However, it suffers from the same scalability problem as smart cards and other physical token methods: the tokens become burdensome if required for many different domains, resulting in a stack of such tokens for the user to bear.

Biometrics attempt to solve the problem of "what you know" and "what you have" authentication methods by the use of an appealing concept: authentication by using the unique physical or behavioural characteristics of users, e.g., fingerprints [135], the iris [32], voice recognition [99], on-line (hand-written) signature verification [126, 84], and keystroke dynamics [100]. Static biometrics (e.g., fingerprints and iris) suffer from a major drawback: they cannot be (easily) changed. Because this biometric information is valid for the lifetime of the user and risks being stolen, such information cannot be used as keying material for remote authentication purposes. Furthermore, even when performing local identification, certain types of biometric readers cannot detect fraudulent inputs. For fingerprints, it has been shown that a gelatin finger that models a legitimate user's fingerprint (e.g., lifted from a glass) can fool many commercial fingerprint readers [91]. Other practical problems of biometrics include that a percentage of users will simply be unable to enroll [70].

Lopresti et al. [86] introduce the concept of generative attacks for behavioural biometrics. Generative attacks use information obtainable from the target user (or similar populations in general), and recombine the information to create a candidate login attempt. Ballard et al. [6] generate and successfully apply a generative handwriting-recognition attack based on population statistics of handwriting, collected from a random sample of 15 users with the same writing style. In arguably the most realistic study to date of the threats faced by behavioural biometrics, they found their generative attacks to be more effective than attacks by skilled and motivated forgers [6]. Their results call into question the use of handwriting biometrics, and their methods might be also be applicable to other behavioural biometrics (although this remains to be shown), particularly when population statistics for the features used have little variation, and when the information used is publicly available (e.g., the user's voice, keystroke dynamics, or handwriting).

The following set of authentication method requirements emerge from the unresolved problems outlined herein, and other well-known requirements for user authentication (see Section 8.3).

R1. *Changeability.* If the user's authentication information is compromised, we must be able to replace this information (and revoke any old password or access credential).

R2. *Shoulder-surfing resistance.* The scheme must not be vulnerable to shoulder-surfing, particularly in the presence of ubiquitous visual recording devices.

R3. *Theft protection.* This includes physical theft and the infeasibility of generative and guessing attacks. If we must rely on the entropy of an authentication scheme for protection against off-line dictionary attack, we require an authentication method whose entropy can scale with processor speeds, Moore's Law, and increasing distributed collaboration.

R4. *Protection from user non-compliance.* To discourage unintended transfer to other parties, the user should not be able to write down (in a manner useful to an attacker) or share their authentication information "too easily".

R5. *Usability.* Authentication should be usable in terms of both (a) elapsed time for authentication, and (b) ease to perform authentication task. If the authentication process takes too long, or is difficult to perform, people will prefer to use another (more usable) method or none at all.

R6. *Cost.* System setup and maintenance should not be costly in terms of time and/or money.

R7. *Acceptability.* This is dependent upon on the user base and environment, but can be defined in terms of being: socially acceptable, not destructive to personal privacy, and perceived as filling a need [29].

Although several known methods meet a subset of these, none meet all of these desiderata. Until 2006, it appeared that on-line (hand-written) signature verification met the last six requirements, and possibly even the first; a signature could be changed in the sense that it is not the user's signature written and verified, but a password or graphical design. It now appears that handwriting is particularly vulnerable to skilled forgers and generative attacks [6].

Knowledge-based schemes suffer from lowered entropy due to user choice, physical tokens can be expensive to deploy and are susceptible to loss or theft, static biometrics are not changeable, and some behavioural biometrics are susceptible to generative attacks. Although $R1 - R7$ are not necessarily essential in all environments, we believe they are representative for many workplaces and labs.

We conjecture that pass-thought systems have the long-term potential to satisfy most of the requirements identified herein, except $R5$(a) and $R6$ depend upon advances in hardware as discussed in Section 7.8. $R7$ is unknown at this point, and would require investigation as discussed in Section 7.8.1.

7.3 Basic Concept

Imagine if we could authenticate by *thinking* a password. We could avoid the shoulder surfing problem associated with most "what you know" schemes by simply "transmitting" some chosen thought or response, authenticating with our minds. This type of authentication might also provide a "who we are" by virtue of the uniqueness of our individual brains (see Section 7.4). While such an idea might appear to lie in the realm of science fiction, recent advances in Brain-Computer Interface (BCI) technology give evidence that authenticating with our minds is within our technological reach.

The main goal of BCI research is to provide an alternative communication and control channel that does not depend on the brain's normal output pathway of peripheral nerves and muscles [156]. The driving application for BCI research is the communication and control of prosthetic devices for disabled patients. BCIs have been a hot topic for the past few years, making notable progress such as using a paralyzed patient's thoughts to control a robotic arm [43], and allowing paralyzed patients to communicate (albeit slowly) [12]. In general, these BCIs work by observing a brain signal S, extracting its features F, and then translating or classifying these features into some recognizable command C through the use of signal processing and machine learning techniques. While BCI technology is still very immature, current efforts have demonstrated that the electrical signals generated by our brains can be recorded and interpreted by man-made sensors. Further details on BCI technology are available elsewhere [11, 153].

We propose an authentication method for access to computing devices, whereby a user *thinks* a password. We call this method a *pass-thought*, the general concept of which is illustrated in Figure 7.1. Steps 2-5 from the general pass-thought description in Figure 7.1 could be performed using a BCI.

There is a significant difference between what the BCI research to date can offer, and the BCI requirements of pass-thoughts. BCI research has been focussed on enabling a user to control something external (e.g., movement of a cursor) using their thoughts. For a user to provide control commands using their thoughts, their thoughts must undergo translation (interpretation). Pass-thought input should undergo feature extraction to filter out the non-repeatable parts, but there is no need to translate the brain signals. Such translation is not only unnecessary but undesirable as it might reduce the entropy the user's brain signals provide. Given that the translation of signals is one of biggest challenges for BCI research, our proposed pass-thought application might be better suited to BCI technology than the current communication and control applications under research.

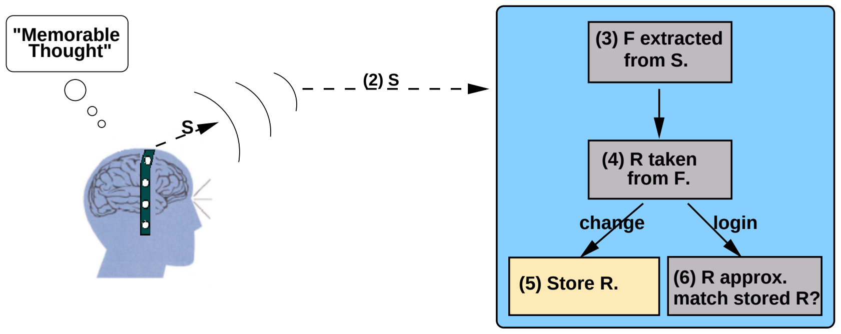


Figure 7.1: General concept of a pass-thought. 1) The user presses a key when ready, thinks of their previously chosen pass-thought (or responds to a presented challenge), and presses the key when done. 2) Electrodes record the signal S emitted during the time between the start and stop key presses. 3) S is processed into signal features F. 4) The subset of features $R \in F$ are those that best capture the user's thought and are consistent over time (determined by trials). 5) R acts as a template, most likely stored using error-tolerant encryption where R would encrypt a key or checkword in such a way that permits some small level of error (e.g., using methods of Monrose et al. [100]). 6) The stored R is used for user authentication to a computing device; login success is indicated by the user's pass-thought entry R^{ℓ} approximately matching the stored R.

7.4 The Potential of Pass-thoughts

Our authentication problem is one of extracting high-entropy information from a user to prove that they are who they claim to be. This essentially means extracting something that makes a person unique. It is interesting to consider how people recognize one another: aside from appearances, we recognize a person's movements, actions, expressions, and speech, all of which are initiated by thought in the brain.

There are a number of reasons to believe that there is uniqueness (given genetic and environmental differences) within our brains: certain areas of our brains are developed more depending on our training and experience. For example, string musicians are known to have larger somatosensory cortical areas associated to the fingers than the average person [45]. Also, the alpha frequency (a signal feature in an electroencephalogram signal) has been found to have considerable variability between subjects [42].

Other results show that the electroencephalogram (EEG) has biometric potential [118, 117, 127]. This biometric has been shown to differentiate 40 people with 95%

accuracy [118]. Although using the EEG as a biometric is interesting, these studies overlook the aspect of changeability. The brain has a vast number of states and responds differently to various stimuli. Pass-thoughts focus on the changeable (or non-static) nature of these signals, which might be augmented by our individual EEG characteristics (or other brain phenomena) to produce a non-static biometric.

These results may imply that the signals emitted from our brains are different upon thinking “the same thing”. Thus, it is plausible that if two people think of what they could best describe as the same thing, the brain signals emitted would be distinguishable. Similarly, we also expect two different thoughts by the same person to result in distinguishable signals, as differing signals are what allow BCIs to perform different tasks on behalf of the user.

If we conjecture that a user’s pass-thoughts could be recorded with enough accuracy to distinguish between different thoughts and distinguish the differences between different users’ “same” thoughts, pass-thoughts may be a natural two-factor (what we know and who we are), non-static authentication scheme. A pass-thought is changeable (the thought itself; the “what you know” portion), and the physiological uniqueness of a user’s brain that emits the pass-thought would act as a second, biometric factor.

In addition to users creating pass-thoughts, a pass-thought could be the measured response to a stimuli (e.g., pictures, music, video clips, or the touch of raised pin patterns).

The theoretical entropy of pass-thoughts is potentially enormous. A pass-thought could belong to a language (as in textual passwords), an image (as in graphical passwords), a type of (imagined) movement, an abstract thought, an emotion, a memory, etc. An entire sentence, picture, or memory (or sequence thereof) can be represented by a simple thought. Even pieces of music can be represented by a thought. There is also a significant amount of variation within the same type of thought. For example, a user could think of their first dog in countless ways through combinations of a variety of factors including the dog’s name, breed, smell, bark, colour, visualizing the dog doing activities such as running through the park, sleeping, eating, licking one’s face, etc. (not to mention the places, movements, and emotions associated with each of these actions). It is impossible for us to know the theoretical size of the pass-thought space; a meaningful measure would depend upon many variables including the method of capturing signals, the duration of the capture, the stimuli used (if any), and signal processing method.

While pass-thoughts have the potential to be an authentication scheme with an extremely large password (pass-thought) space, in practice there would be boundaries on the size of the pass-thought space that correspond to the encoding scheme. Since the large theoretical pass-thought space will be mapped to a smaller encoding space, it is possible that collisions will occur between different users (or the same user) with different pass-thoughts. There may also be a more probable subset of pass-thoughts as with other "what you know" authentication schemes that correlate with the strengths of human memory. While these factors may limit the theoretical strength of pass-thoughts, we conjecture that the "who you are" factor, or the use of a challenge-response scheme (see Section 7.5.2), might compensate for such limitations.

Despite this promising potential, we recognize that BCI technology is still in its infancy, and thus the potential accuracy of signal recording and processing is unknown. However, as indicated by an analysis of the number of publications per year on the topic of "Brain Computer Interface" using the ISI Web of Knowledge [66], research interest in this topic is steadily increasing. For example, from 1995-1999, there were only a total of 28 publications, but there were more than 50 in each year from 2003-2005, and over 100 in 2006 alone.

7.5 Example Pass-thoughts Deployment Scenarios

Here we discuss a few example deployment scenarios for pass-thoughts. We discuss the parameters, type of memory and user response involved, advantages, and disadvantages of each method. The example described in Section 7.5.1 is a scheme that should be possible today, but only involves user choice, and is less likely to contain an additional biometric factor. Thus, the extra security provided by this method is limited to that of preventing shoulder-surfing. The example described in Section 7.5.2 is on the other side of the spectrum, not necessarily involving user choice or voluntary response, but an innate response to a set of stimuli.

7.5.1 Simple Scenario: Flashing Grid

In this section, we discuss a feasible (in the sense it could be built) pass-thoughts system given current BCI technology, guided by the 6 steps in Figure 7.1. We assume the same BCI capability as that shown in recent BCI prototypes such as the Thought Translation Device [12]. We also assume that the user is logging onto a desktop PC, where the pass-thought will be used exactly as a password.

To begin, the user presses a special key sequence when ready (e.g., CTRL-ALT-DEL). For a currently possible pass-thought system, we propose a scheme similar to that which uses evoked P300 potentials for a spelling device for the disabled [10]. A P300 potential is a positive potential that is evoked about 300ms after a surprising or exciting event. By randomly highlighting the components (either textual or graphical) on the desktop's monitor, when the user sees the part of their "pass-thought" highlighted (e.g., see Figure 7.2), they presumably generate a P300 spike as for the spelling device [10]. The results of the P300 potential spikes are silently recorded and determine whether the user's P300 firing matched the expected template that represents the account's password. This type of scheme could be used in conjunction with either textual or graphical passwords, where a sequence of letters, pictures, or points on a picture are highlighted at random times (thus randomly generating the user's P300 potential spikes, where the verification side of the system knows when to expect which bits).

The size of the pass-thought space for this scheme depends on the number of components on the screen and the number of screens presented to the user. If we assume a textual password scheme where all 95 printable ASCII characters are displayed on each screen, and the user must select a sequence of 8 characters, the size of the full pass-thought space is 95^8, approximately 53 bits. Of course, we would not expect 53 bits of security for the same reasons that we do not for textual passwords. This pass-thought based system is shoulder-surfing resistant and otherwise provides the same amount of security as textual passwords. One disadvantage is that given the current state of BCI technology (P300-based approaches have shown a bit rate of 4.8 characters/minute to obtain 90% accuracy on a 36-character grid [41]), this process would take 1 minute and 40 seconds if the performance did not degrade with a larger

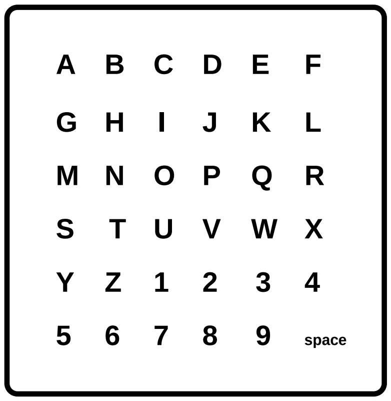


Figure 7.2: Illustration of a screen intended to randomly highlight letters (from the spelling device in [10]). The technology exists such that this approach could be used for a pass-thought system.

character grid.

Electrodes record the P300 spikes generated by the user. In this scheme only one signal feature is used, since without future studies to determine the variation and repeatability of brain signals, we only have the BCI literature to date as a guide of what is possible. The results for BCI communication so far have low bit-rates, thus we can only assume a yes/no answer. Thus, in the context of this scheme F (recall Figure 7.1) is a set of P300 potentials, and $F = R$.

Depending on the rate of highlighted screen components, the user may not generate a few P300 potentials during the expected time; however, the algorithm should only record those P300 spikes that occur, not the ones that were missed. This allows the system to be exactly repeatable, and we could use a one-way hash function h to store the pass-thought instead of a type of error-tolerant encryption.

Authentication using this basic pass-thought system can then be performed exactly as with textual passwords. Input completion occurs either after a certain number of P300 potential spikes have been received, or by the user pressing the key sequence again. The hashed pass-thought $h(R)$ is compared upon input completion to the stored pass-thought file hash for the user, and login success occurs if they match.

In addition to the system described above, there are other simple pass-thought systems that could be implemented using today's technology. For example, a pass-thought system could be built that is based on touch. In this type of system, a user might place their palms down on a surface that raises pins that cannot be observed, but can be felt by the user. The user would have a subset of raised pin patterns that they should think "yes" to upon feeling; other raised pin patterns should evoke a "no" thought from the user. The "yes" answers could be obtained using the user's P300 potential spikes. Lack of a P300 spike could be interpreted as a "no" answer.

7.5.2 Challenge-Response Scenario: Stimulus-Based

This scenario would require the system to present the user with a set of challenge stimuli, and record the user's brain signal during a short period after presentation. This sort of challenge-response system is not cryptographic in nature, but challenges the user with stimuli to evoke a set of one or more responses. A pass-thoughts challenge-response system would record a user's innate (involuntary) response to stimuli, and optionally a voluntary response (e.g., a memory associated with the challenge). This type of pass-thoughts system could be configured such that no user choice was required, and would not necessarily require memory on the part of the user (although the user would likely recognize the stimuli over time).

Alternately, it could be configured such that it records a user's voluntary response to the challenge at a certain time, after the period whereby an involuntary response occurs. In this configuration, the system cues the user to remember an associated thought, thus it is a "cued-recall" style of system. For example, if images are used as stimuli, any signs of a Visual Evoked Potential (VEP) are typically complete by 400-500ms. If the user is instructed to think of a voluntary response after the image disappears (at 500ms), then the signal during this time could be measured.

Examples of challenge stimuli include images, sounds, or words. Almost anything could be used as a challenge stimulus, provided that it has been shown to produce repeatable signals as a response. The security of this scenario is defined by the number of challenge stimuli and the number of bits obtainable by measuring each response. The number of challenge stimuli provided in a login session should be set based on the number of repeatable bits obtainable (from the signal associated with that particular stimuli), and the desired level of security for the system. For example, if only 10 bits are obtainable from each stimuli, then 6 stimuli would provide 60 bits of security, and if only innate responses are measured (and found to have sufficient variability between different stimuli), then this system would be free of weak password subspaces.

Our preliminary lab experiments,[2] whereby the user is presented a visual stimulus, and is cued to think of a previously associated thought 500ms after the stimuli is presented, have indicated that using an involuntary response is much easier than voluntary responses with today's EEG technology. This is due to the difficultly of determining (in milliseconds) when the voluntary portion of a response begins. The starting time of the response is required to align and average a number of trials to reduce the signal-to-noise ratio of the EEG. This would make it seem that a cued-recall style of pass-thoughts system is unlikely to work with the current EEG-based BCI technology. However, using the innate portion of the response still appears to have promise, and the cued-recall style system might be reconsidered when other, possibly better methods become more convenient (e.g., functional near-infrared spectroscopy [11]).

7.6 Discussion of Security

In this section, we discuss how pass-thought-based systems might respond to different theft and guessing attacks. A pass-thought system is visually unobservable and thus resistant to shoulder-surfing attacks. Pass-thoughts are also resistant to acoustic attacks [170], but might be vulnerable to a tempest attack [80] (i.e., reconstructing content from electromagnetic radiation). Interception attacks (i.e., when using a pass-thought for remote authentication) can be avoided using the same protection mechanisms as for traditional passwords (e.g., encrypted channels and challenge-response protocols; see Section 7.5.2).

If signal recording and processing methods advance such that they are able to capture thoughts in detail such that the thought itself (or response to a challenge) could be used, a pass-thought would be quite difficult to communicate to a social engineer.

[2]Using the first 16 channels of an EEG cap that uses the International 10/20 electrode placement system, using an amplifier gain of 5000, sampled at 4000Hz and subsequently filtered at 2000Hz.

Using such a scheme, even if a particular pass-thought is successfully communicated, a social engineer's brain signal may be different than the user's upon thinking "the same thing" (recall Section 7.4). For these reasons, we (perhaps optimistically) hope that the size of the pass-thought space might be sufficiently large to protect against most dictionary attacks.

A pass-thought based system cannot avoid the threats of phishing attacks [39] and of recording the pass-thought through either a hardware or software tap (i.e., when the input device has been compromised). It is unclear whether any authentication method (on its own) could be entirely immune to phishing attacks. However, if pass-thoughts could be constructed such that each transmitted pass-thought was used only once, we could potentially defeat passive interception attacks.

7.7 Discussion

There is a clear need for shoulder-surfing proof user authentication, especially given the ubiquity of cell phone cameras and wireless video cameras. One of the primary potential benefits of pass-thoughts (see Section 7) is that they are visually unobservable and thus are resistant to shoulder-surfing. Pass-thoughts are also silent, and thus are resistant to acoustic attacks [170]. Another idea for unobservable authentication method might be to make use of eye-gaze tracking (using e.g., the LC Technologies Eyegaze Systems [83]). Such an eye-gaze based method could permit unobservable passwords of the same strength provided by textual or graphical password schemes by allowing the user to select parts of the password with their eyes (e.g., by eye fixation for a specified period denoting selection), and not echoing the input on the screen.[3] While the security of such a system is comparable to the pass-thought systems that can be built with today's technology, advances in BCI technology could lead to much more powerful pass-thought authentication systems (recall Section 7.4).

The ultimate feasibility of pass-thoughts is dependent upon the accuracy of a BCI in recording the repeatable parts of a brain signal. For a brain signal to be repeatable, the signal features that represent the intended pass-thought must be extracted. Given

[3]This idea has been implemented and studied for usability by Kumar et al. [81] since our original publication of this work [153].

the BCI research to date, we know that at least a binary response can be evoked, and would thus be at least as effective (if not considering input time) as an authentication method that makes use of an eye-tracking device (e.g., Eyegaze Systems). If BCI technology advances towards enabling accurate and repeatable brain signal input (which we conjecture is likely given the recent advances), we might have a new authentication method that could solve many of the problems associated with current systems. The flexible nature of pass-thoughts could allow for strong authentication to scale with increasing processor speeds. Increasing the complexity of a pass-thought might be as simple as recording the response to an extra challenge. A pass-thought system would be visually and acoustically unobservable, defeating the threat of recorded (and unrecorded) shoulder-surfing and recorded acoustic attack. It would be difficult for users to share or write down exact thoughts, as the pass-thought might not be describable by communication mediums such as language, drawings, and demonstrated movement. Users could still write down a note for themselves to remind them of their pass-thought, which would presumably be of little value to an attacker due to the user's signal feature variations.

Perhaps the problem of choosing repeatable signal features will only be solved by user training and using an authentication method that does not require exact repeatability. It would be interesting to determine how difficult it is for most people to control their brain state enough to reduce noise upon entering a pass-thought, and whether mood and stress would interfere. A low training time is very important for user acceptance, and is yet another important area for future work. Perhaps during training, a method for providing the user some feedback about their brain signals would be useful (e.g., a real-time changing image that represents the signal features, where each feature is shown in different colours).

7.8 Preliminary Steps and Future Requirements

There are many unknowns to resolve before pass-thoughts might become the method we envision. It is our hope that this idea for a pass-thought system will inspire research into the area of BCI signal processing algorithms that retain as much repeatable information as possible. Section 7.8.1 outlines what we believe should be the first steps in proving the potential of pass-thoughts, and Section 7.8.2 details other practical requirements that must be met before a pass-thoughts system could be deployed.

7.8.1 Preliminary Steps

We have reason to believe (recall Section 7.4) that the signal encoding a transmitted thought (or response) would be dissimilar from one user to another. No particular experiments to verify this appear to exist; thus an interesting first step in this area would be to confirm and/or quantify the base premise that there are differences between the brain signals generated by people who are thinking or responding to "the same thing". A confirmation might include a visual signal analysis, a statistical analysis of signal features, or the use of machine learning techniques to separate signal features.

Once the base premise has been confirmed, it would be interesting to measure the likelihood of signal collisions between different users with different pass-thoughts, how many people have different pass-thoughts when thinking "the same thing", and how stable signals such as EEG and the P300 are over time.

Also, the repeatability of these signals over time will need to be evaluated. Existing "EEG as a biometric" studies have shown that classification (to separate EEG signals of different users) works well using data from a single session, but no studies have yet examined how much day-to-day variability exists. If the signals shift slightly over time, then it may be possible to update the signal template used when it goes beyond a pre-specified drift threshold. Also, it is important to study whether individuals with mental disorders have unique repeatability considerations.

Finally, how users would accept this technology is important to determine. As discussed in Section 8.3.2, many users are uneasy about using eye-scanners, so it is conceivable that people will think that a pass-thought reader is actually interpreting their thoughts. If/when the above steps achieve positive results, and the necessary hardware is developed to make a usable pass-thoughts system, this issue must be examined.

7.8.2 Future Requirements

If the preliminary experimental steps outlined in Section 7.8.1 provide evidence for the base premise, there are still practical considerations that must be met prior to deploying a pass-thoughts system.

The most common method to record brain signals for BCIs is an electrode cap to record EEG signals. This method is known to be noisy, with a long setup time. Although electrode caps are fine for a proof-of-concept, they would not provide acceptable usability in most real-world systems. Thus a better hardware interface is required. A hardware interface that has the look and feel of headphones may feel less awkward than electrode caps. A wireless headphone-style electrode interface might have electrodes lining the band going over the head, around the ears, and/or on a small disc that is attached to the hardware that extends towards the back of the skull (to place electrodes at the top of the skull if necessary). For example, Figure 7.1 depicts the user wearing such an interface that wirelessly transmits S. If the device is wireless, of course the communication channel between it and the receiving device must participate in a cryptographic protocol that prevents eavesdropping, capture, and replay.

This sort of BCI hardware is already in development [107]. It is possible that the number and placement of electrodes of such an interface could obtain an acceptable signal. An acceptable number of electrodes has experimentally been found to be as low as 8 [93], and the optimal placement of electrodes depends on the signals the BCI wishes to focus upon (e.g., event-related potentials, slow cortical potentials, or mu-rhythms) [166]. We hope that the cost of such devices will eventually decrease; present portable BCI systems cost approximately \$4,300 EUR [58].

7.9 Summary

We propose the high-level concepts of a novel idea for user authentication that we call "pass-thoughts", whereby a user simply thinks their password, transmitting it directly to a computer using a BCI. While the potential characteristics of such a system are interesting, it's not clear at this point whether a production-ready system would ever be possible. A tremendous amount of work remains to verify whether even the basic ideas will work in a lab environment. While this lab work is beyond the scope of this thesis, we hope to continue to pursue this path in future work.

Chapter 8

Providing Context: Discussion and Conclusions

8.1 Introduction

In Chapters 4, 5, and 6, we have shown how predictive models can serve as a starting point for creating attack dictionaries for five different types of knowledge-based user authentication schemes. To recap, these five schemes are: traditional text-based passwords, passphrases, recognition-based graphical passwords (i.e., Faces and Story), pure-recall graphical passwords (i.e., DAS_J), and cued-recall graphical passwords (i.e., PassPoints).

Resistance to dictionary attacks is one of many important properties of an authentication scheme. There are of course other attacks to consider, and other considerations (than security) when deciding what authentication scheme to adopt or deploy. We survey a set of schemes reviewed in Chapter 3, and perform a critical analysis of their resistance to dictionary and other attacks in Section 8.2. We discuss other practical factors (e.g., usability and cost) in Section 8.3. Section 8.4 discusses resulting recommendations that reflect our updated knowledge. Finally, Section 8.5 discusses future work and concludes this thesis.

8.2 Critical Analysis of User Authentication Schemes and Attacks

Here we perform a critical analysis of user authentication schemes (reviewed in Chapter 3) and their resistance to various types of attack. We consider the following attacks: brute force attack (recall Section 2.2.1), dictionary style attacks which rely on reduced entropy of user choice (recall Sections 2.2.2, 4, 5, and 6), generative attacks which rely on available statistics for similar populations or for the user (recall Section 7.2), shoulder surfing (recall Section 7.2), and social engineering attacks (recall Section 7.2).

Table 8.1 compares vulnerabilities to the known attacks discussed in Section 7.2. If the scheme is vulnerable to the attack in an atypical way, a special footnote is provided in place of a ‘y’ in the caption. If there is reason to believe that a scheme may be vulnerable to an attack, but no supporting analyses have been performed yet, it is marked with a ‘?’. Finally, a blank means that there are no foreseeable applications of that attack to the particular scheme.

In addition to the schemes discussed in Chapters 2 and 3, we include alternatives that do not rely on a user’s memory: static and behavioural biometrics and physical tokens. Static biometrics measure unchanging physiological traits of a user such as fingerprints, the iris, and facial features [69]. Behavioural biometrics include keystroke dynamics [100], voice [99], and handwriting [126]. Physical tokens include smart cards [1], zero-interaction authentication (ZIA) [27], and passcode generators such as SecureID tokens [137].

In summary, Table 8.1 shows that all schemes are vulnerable to some attacks. In the case of behavioural biometrics, only one particular type (handwriting) has been analyzed for generative attack, and it was found to be vulnerable. Other behavioural biometrics have yet to be examined for vulnerabilities to generative attack. The only schemes that we can be sure are not vulnerable to brute-force or dictionary attacks are those with large theoretical entropy that do not involve user choice; for example, Weinshall’s protocol uses system-assigned secrets and has a large full space (although this extra security comes at the cost of a long training time of 1.5-3 minutes). Shoulder-surfing is another common problem: the only schemes that we can say are secure against it are some biometrics and physical tokens. Finally, social engineering is another common problem: the only schemes that are not vulnerable to this style of attack are Déjà Vu (assuming that the random art images used cannot be described as indicated by Dhamija et al. [38]), and behavioural biometrics.

8.3 Additional Considerations

There are of course other considerations, aside from security, that are important in deciding an authentication scheme to adopt or deploy; this is why passwords are still commonly used. For example, a smart card is very secure in terms of the entropy it

Category	Authentication Method	BF	DA	GA	SE	SS
Text	Text Passwords		y		y	y
	Passphrases		y		y	y
	Inkblot Authentication [145]		?		y	y
Graphical Pure-Recall	DAS/Pass-Go [72, 146]		y		y	y
Graphical Cued-Recall	PassPoints [164]		y		y	y
	Blonder's Scheme [15]		$\diamond$		y	y
	Picture Password [71]		$\diamond$		y	y
	VisKey [139]		$\diamond$		y	y
	V-Go [120]		$\diamond$		y	y
	Cued Click-Points [25]		?		y	y
Graphical Recognition-Based	Passfaces [131]	y	y		y	y
	Story [34]	y	y		y	y
	Déjà Vu [38]	y				y
Graphical Challenge-Response	Weinshall's Protocols [161]				y	†
Other	Static Biometrics	$\bullet$			‡	$\triangleleft$
	Behavioural Biometrics	$\bullet$		$\circ$		$\triangleright$
	Physical Tokens				$\star$	

Table 8.1: Existing authentication schemes and attack types. A scheme is vulnerable to an attack if it is marked with a 'y'. We consider the following attacks: brute-force attack (BF), dictionary attack (DA), generative attack (GA), social engineering (SE), and shoulder surfing (SS). $\diamond$ Under conjecture that this scheme will suffer from similar attacks as shown for PassPoints. † Weinshall's Protocols have been shown to be susceptible to SAT solver attacks (see Section 3.3.5). $\bullet$ Hill-climbing attacks [155] can be successful when feedback regarding the matching scores of each guess can be observed by the attacker. ‡ Some static biometrics have been shown to be easily forged using a stolen physical sample of the user's biometric (e.g., a fingerprint [91]). $\triangleleft$ Shoulder-surfing (with cameras) is a threat for biometrics whose features can be reconstructed from an image (e.g., a fingerprint). $\circ$ This has been shown for handwriting biometrics [6, 5], but may also be the case for others, particularly when the information used is publicly available. $\triangleright$ A static image of a handwriting biometric allows for easier generative attacks and attacks by skilled forgers [5]. $\star$ Users can be tricked into "lending" physical tokens, or they can be stolen.

provides, but it is costly to deploy and can be lost or stolen. Biometrics are easy for people to use as they need not recall anything, but if the biometric data is stolen, an attacker could masquerade as the user, and the user cannot easily change their biometric. Here we discuss a set of other problems that exist with otherwise secure systems; usability in Section 8.3.1, acceptability in Section 8.3.2, changeability in Section 8.3.3, cost in Section 8.3.4, and loss or theft in Section 8.3.5.

8.3.1 Usability

Usability is a broad topic, which can be defined as "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [65]. The usability of an authentication system includes its accessibility, memorability, and convenience.

Accessibility refers to barriers the user may experience in terms of being able to complete the authentication task. Accessibility barriers include physical (e.g., a disability that prevents a user from using a mouse), cognitive (e.g., reduced memory skills), sensory (e.g., blind or aging users may have difficulty seeing an image), and technical (e.g., requiring special hardware, software, or technical expertise) [133]. For example, biometrics have the problem that some users are un-enrollable [29], thus backup provisions must be made. For fingerprint biometric systems, some people are not able to use them due to loss of fingers, or poor fingerprint definition.

Memorability refers to knowledge-based systems and whether users find it easy to remember the required information to login, both in the case of a single account using the system, and multiple accounts using the system (which may suffer from interference). For example, although using two graphical passwords reduces success rate in click-based graphical passwords [24], findings for interference in graphical passwords are more positive than for PIN numbers: Moncur et al. [96] show that five recognition-based graphical passwords were forgotten less often (as measured by three login failures) than five PIN numbers.

Convenience includes the time to enroll, authenticate, and replace (when forgotten or lost) [133]. If any of these tasks take too long, users will be annoyed with the system and may resort to insecure measures such as disabling the system. The time to authenticate is arguably the most important, as this is presumably the most frequent task a user must complete.

8.3.2 Acceptability

Coventry [29] states that for consumers to adopt biometrics, they must find that the technology is: socially acceptable, appropriate for a given environment, filling a perceived need, usable, and not destructive to personal privacy. For a user to find a technology socially acceptable and appropriate for a given environment, they must not feel embarrassed to use the system in its intended environment (e.g., trying many times before success in a public place).

Usability factors that could influence acceptability are discussed in Section 8.3.1. Usability might also relate to a user's perception of whether an authentication system is an imposition. For example, some users may object to carrying physical tokens with them, as it could result in lost productivity in the event that the token is lost, stolen, or simply left at home (or elsewhere).

A user's perception of how an authentication measure impacts their privacy (regardless of whether it is accurate), is an extremely important factor as to whether a system is widely accepted. For example, some users might feel uneasy about iris or retina scanners, as some studies report that the eye provides information about your health and/or personality [7]. One could imagine this also being a problem with other biometrics.

8.3.3 Changeability

If the user's authentication information is compromised, we must be able to replace this information (and revoke any old password or access credential). For example, static biometrics cannot be (easily) changed, which is a major drawback. Because biometric information is valid for the lifetime of the user and is subject to theft, such information should not be used as keying material for remote authentication purposes.

This issue becomes particularly important if many different systems are using static biometric authentication; it is analogous to the problem of having the same password on many different systems, whereby an attacker with a user's compromised biometric might be able to gain access to many of the user's accounts.

8.3.4 Cost

The cost of initial setup and operating costs of an authentication system (in terms of both time and money [133]) are extremely important considerations. The relatively low expense of implementing and maintaining passwords is a major reason for their ubiquity. Items that factor into the cost of initial setup of an authentication system include initial hardware and software costs, cost of training and enrolling users, and the cost of a backup method (for users who cannot access the primary system). The cost of operating the authentication system includes the cost of software or hardware maintenance, and of replacement (e.g., resetting forgotten passwords, or replacing lost physical tokens).

The issue of cost is very dependent on the environment in which the scheme will be deployed; to a large company that is highly concerned about security and has the resources, a higher-cost system might make more sense if it provided better overall security. However, to a small start-up company offering an online service, low cost could be the most important consideration for a system that authenticates their users.

8.3.5 Loss or Theft

This includes physical loss and theft of a token (e.g., a smart card or passcode generator) that is required for successful authentication. If such a token is lost or stolen, it renders the system temporarily inaccessible to the user. Further, if possession of the token is the sole authentication factor, or if it is combined with a weak password as a second factor, loss or theft of the token could allow an attacker to masquerade as the legitimate owner.

8.4 Resulting Recommendations for Administrators

One lesson that we can take away from our results is that graphical passwords are still immature, and are only beginning to undergo serious analysis. Our results indicate that for both schemes analyzed (DAS and PassPoints), weak password subspaces exist, and without further study it is difficult for us to generally recommend their use at this point in time. However, DAS may still prove to have better effective security than text passwords; the smallest weak password subspace identified in this work is 10 bits larger than the smallest text dictionary, but we caution that if other complexity properties are considered (such as the number of turns) this may change. The implementation enhancements discussed in Sections 5.6 and 6.8 could make these schemes secure enough to use in some environments, but hesitate to recommend their use with these enhancements until the updated schemes are re-evaluated to determine whether new patterns in user choice emerge, e.g., for proactive checking when a user's first choices are disallowed, and whether usability remains acceptable.

Although our results do not allow us to (at present) recommend the general use of the particular schemes analyzed, they may be suitable for some specific environments. When choosing a suitable method of user authentication for a particular environment, an administrator should consider the following questions:

- What are the security requirements (i.e., would it be acceptable for confidentiality, integrity, or accessibility to be compromised)? For example, is account lockout a reasonable defense despite the denial-of-service risk?

- Is the system to be remotely accessible?

- Are there any special considerations for the user(s), e.g. accessibility, usability, acceptability, loss/theft, or cost? Recall Section 8.3.1.

- Has the authentication scheme undergone thorough analysis, and if so, how large is the smallest weak password subspace?

Due to the number of factors and the variety of possible answers for each of these questions, there is no general set of recommendations that we can give administrators at present. Therefore, we consider only a few example scenarios to illustrate how schemes with different levels of security can be acceptable in different environments.

8.4.1 Example Scenarios

Our example scenarios are chosen to represent different levels of desired security: a bank ATM environment that only requires security against online attack, a mobile device where data integrity is important, and a mobile device where both data integrity and confidentiality are important. Of course, one can imagine many other scenarios that have different requirements; our hope here is to illustrate how some schemes, despite having weak password subspaces, can still provide sufficient security for use in certain environments.

Environments Where Offline Attack is Not a Threat

Consider bank ATMs: an environment where it is accepted that offline attack is not a relevant threat. The authentication process requires both a bank card (a "what you have" factor), and a PIN number (a second, "what you know factor"). The PIN is a 4-digit number, which has 13.3 bits of theoretical security, and alone would be vulnerable to an offline attack. ATMs normally lock out an account after three unsuccessful login attempts – a strategy that increases security against guessing attacks, but decreases availability. Further, banks typically limit the amount of risk associated with a guessed PIN by limiting the amount of cash that can be withdrawn each day. In this environment, a PIN number, despite being vulnerable to an offline attack, has provided sufficient security for bank ATMs. Thus, other authentication schemes that provide at least 13.3 bits of effective security, with similar additional factors (as discussed in Section 8.3) might provide a competitive (or even better) alternative to the PIN number in this environment.

Two of the graphical password schemes shown in Table 8.1 have been found to provide unsuitable security even in this environment: Faces and PassPoints. Davis et al. [34] show that 10% of passwords in Faces can be guessed in 2 guesses. In Section 6.6 we show that 8-11% of passwords in the PassPoints scheme can be guessed in 3 guesses using a similar attack strategy. Story [34] was shown to have 10% of passwords guessed in 35 guesses; although this could be secure against online attack with lockout after 3 unsuccessful guesses, it is rather close, and thus we hesitate to recommend it even for this environment.

The authentication schemes from Table 8.1 that have been analyzed and not yet found to be vulnerable to online guessing attack, are (to date): text passwords and passphrases [82], inkblot-based passwords [145], DAS (see Chapter 5), and the other schemes that do not involve user choice (Weinshall's Protocols [161] and biometrics). Graphical password schemes based on recognizing images have been shown to suffer less interference than PINs [96], however we do not consider these here as there is no analysis that supports their effective security.

Studies to date have shown that all of these schemes appear to have acceptable usability, except Weinshall's protocols which have a much longer login and training times. Regarding memorability, DAS/Pass-Go was found to have 21 forgotten passwords in the three-month user study (out of 167 users), providing a memorability rate of 87.4% (if we assume that each user forgot a password no more than once), which is a better rate than shown for PIN numbers and for 6-character text passwords, of 65% and 70% respectively, by and Dhajima et al. [38] (measured by 3 successive failed logins). This would indicate that DAS/Pass-Go could be considered a reasonable alternative to PIN numbers (as it has better security, despite its weak password subspaces, than PINs; and better memorability). However, further study is required to show whether performance and interference results for this scheme are competitive to PINs. Further, if any graphical password scheme were to replace an ATM's PIN, care would be needed in redesigning the ATM to ensure that the screen is much more easily shielded from shoulder-surfers.

Finally, biometrics would provide enough security to replace a PIN; however, acceptability and possibly cost (depending on the biometric) are considerations that make biometrics (static or behavioural) less appealing than PINs in this environment. Also, a backup method would be required for users that are unable to enroll.

In conclusion, at this time, there is not sufficient evidence to recommend any scheme that has been analyzed to date as a replacement for PINs. Biometrics might serve best as an option for users that can enroll, and have difficulty remembering their PIN, if the bank believes the cost (of readers, installation, and enrolling users) will be offset by the cost of resetting PINs.

Mobile Devices Requiring Data Integrity, Not Confidentiality

Consider a mobile device whose owner only cares about the integrity of the data it contains, rather than its confidentiality; for example, a PDA with no sensitive/private information stored. In this case, the user wants to ensure that no one is able to pick it up for a short time when it's left somewhere and modify or delete their data (e.g. calendars, phone numbers, etc). If the device is stolen, it is inconvenient to the user, but there are (by assumption), only minor concerns of confidentiality. A simple password would provide sufficient security, so long as it is expected to provide at least 2 days worth of offline attack protection.

With our current assumptions (see Section 5.4) about MD5 hashing and personal resources of 8 processors (2 quad cores), this translates to a threshold of $t_e = 44$ bits (recall Definition 1). Every knowledge-based scheme that has been analyzed to date has weak subspaces less than this size. It is however possible that increasing the cost of hashing could help with some schemes. A reasonable increase in the time for hashing would be to make each hash take no more than 0.1 seconds, as this is about the time limit before users start to notice a delay [94]. As discussed in Section 5.4, approximately 1.22×10^7 MD5 hashes can be performed per second on a single 2.66GHz processor, and $1.22 \times 10^6 = 2^{20}$ MD5 hashes can be performed in 0.1 seconds. The hashing algorithm can thus be modified to add 20 bits of security before a delay is noticed, making a scheme with a smallest weak subspace of 24 bits acceptable for use in this scenario.

Of the analyzed knowledge-based schemes, the one with the smallest weak subspace to date is DAS: Class D_{1b} intersected with Class D_2 is 31 bits and guesses about 16-17% as shown by Tao [146]. However, as previously mentioned, performance and interference still must be studied in this scheme to determine whether it is comparable to text passwords. Weinshall's protocols could also be secure enough for this scenario, since there are no weak password subspaces (the images are system-assigned), but its usability suffers from long login and training times. Finally, some biometrics could provide enough security here; however, acceptability and cost are considerations that make biometrics (static or behavioural) less appealing.

In conclusion, at this time, when considering both security and usability, there is not sufficient evidence to recommend any scheme that has been analyzed to date as a replacement for regular passwords with password stretching of at least 23 bits to compensate for its smallest weak password subspace of 21 bits [82]. Twenty-three bits of stretching would likely be noticeable to the user (approximately a one second delay), but might still have better performance results compared to the remaining alternatives.

Mobile Devices Requiring Data Integrity and Confidentiality

Consider a user who owns a laptop that contains highly sensitive corporate data (e.g., the design and marketing plan for a new product). The company would like to keep this data confidential until the product is scheduled for release in a year. The password in this case not only controls access to the laptop, but also is used to either generate a key (or decrypt a key) that decrypts the data on the hard drive. The company would thus like to have at least one year of security against offline attack in the event that the laptop is lost or stolen.

With the same assumptions about MD5 hashing and personal resources of 8 processors (2 quad cores), this translates to a threshold $t_e = 51.5$ bits (recall Definition 1). There are weak password subspaces of this size for most schemes studied, even when a reasonable amount of password stretching is assumed. The exceptions are DAS (with at least 20 bits of password stretching) and Weinshall's protocols; however, as mentioned in the last section, DAS still must be evaluated for performance, and Weinshall's protocols have usability problems. Thus, in this environment, a second factor is necessary to increase the level of security, e.g. a physical token or biometric. One commonly deployed solution as a second factor is a passcode generator such as RSA's SecurID, which is synchronized with a remote server to display a new number every minute. The user enters both the number and a memorized password. However, such a method is more appropriate for remote access (as it requires an Internet connection for verification of the number it generates). Alternately, if the user can enroll, a biometric may be a more appropriate second factor, as any physical token used to access a mobile device might be stored near the device itself (so theft would also obtain the physical token). However, physical tokens may be necessary for those users who cannot enroll in the chosen biometric. Both of these solutions will have additional costs associated with the hardware readers, but in such a high-security scenario, the cost would likely be considered worthwhile.

8.5 Conclusions and Future Work

This thesis shows that some established attack strategies for knowledge-based user authentication methods can be generalized, describes how they work against two previously unanalyzed representative graphical authentication schemes, and discusses other types of user authentication schemes that might be more immune to these (and other) attacks, such as "pass-thoughts".

Our results, discussed in Section 8.5.1, indicate that the method we introduce and call *predictive modelling* can be applied to previously unanalyzed schemes to identify weak password subspaces. We do not view this as a negative result, but as a step towards better understanding how to evaluate the security of new schemes prior to their widespread deployment. Section 8.5.2 discusses the specific conclusions that fall from our results. We end with a discussion of future work in Section 8.5.3.

8.5.1 Summary of Results

In Chapter 4, we introduce the idea of predictive models, and how they map to existing authentication schemes and attack methods. In Chapter 5, we used predictive models to demonstrate three weak password subspaces for DAS: Class D_{1b} (global symmetry), Class D_2 (4 or fewer strokes), and Class D_{1b} $\cap$ Class D_2. In Chapter 6, we used predictive modelling to demonstrate three weak password subspaces for PassPoints: Class C_2 (based on human-computed data), Class C_3-DIAG (the diagonal click-order pattern), and Class C_2 $\cap$ Class C_3-DIAG. These results add two more schemes to the body of literature supporting our hypothesis that other knowledge-based user authentication methods are weak when based on user-chosen secrets. This result, along with earlier results by others [77, 167, 34, 82], would suggest the emergence of a general pattern that calls into question the effective security of any candidate knowledge-based authentication scheme, when based solely on user-chosen secrets.

Naturally, these security analyses lead us to a set of specific tentative recommendations that may help improve the security of DAS and PassPoints, and to outline existing and new ideas for future work that might prove to reduce the problem of predictable patterns in user choice. Finally, we propose a new idea for user authentication that we call "pass-thoughts" in Chapter 7, which may prove to have some interesting characteristics that might lead to better immunity from the attacks discussed in this thesis.

8.5.2 Conclusions

One of the biggest problems with new knowledge-based authentication schemes is that their designers often fail to recognize that patterns in user choice could result in weak password subspaces. Our results contribute to the literature showing that the effective security of a scheme often does not match its theoretical security, and that an analysis of patterns in user choice is necessary to understand the extent of this problem. The primary lesson that we can take away from our results is that cautious optimism is a good policy when considering newly proposed authentication schemes, at least until a few independent analyses have been performed to test for patterns in user choice and show that there are no weak subspaces.

A secondary lesson that we can take away from our results is that graphical passwords are still immature, and are only beginning to undergo serious analysis. Our results indicate that for both schemes analyzed, weak password subspaces exist, and thus it is difficult for us to generally recommend their use at this point in time. However, DAS may still prove to have better effective security than text passwords; the smallest weak password subspace identified in this work is 10 bits larger than the smallest text dictionary, but we caution that this may change if other complexity properties are considered (such as the number of turns). The implementation enhancements discussed in Sections 5.6 and 6.8 could make these schemes secure enough to use in some environments, but hesitate to recommend their use with these enhancements until the updated schemes are re-evaluated to determine whether new patterns in user choice emerge, e.g., for proactive checking when a user's first choices are disallowed, and also to determine whether usability is acceptable.

Finally, a third lesson that comes out of our work is that we *can* be proactive about patterns in user choice prior to deployment. Using predictive models to evaluate a scheme can reveal guidelines that wouldn't otherwise be considered until attacks are reported, and users inconvenienced. They also allow us to understand which target deployment environments are sensible; for example, even though we found DAS to have a weak password subspace of 31 bits, it may still provide sufficient security to replace PIN numbers. On the other hand, for PassPoints using the *pool* image, we were able to guess over 10% of passwords in fewer than 100 guesses using a combination of our weak password subspaces (cf. Figure 6.7); this result makes it difficult to recommend PassPoints (with the parameters set as proposed) even for PIN replacement.

8.5.3 Future Work

Based on our results, we believe that promising directions for future research include research into usable methods for the system-assignment of passwords, and developing non-standard mnemonic strategies for users. In otherwise secure authentication systems, dictionary-style attacks are only possible because of patterns in user choice, thus one possible avenue for protection is to have the system assign random passwords. If passwords are uniformly random, then the full password space provides an accurate representation of the security. The problem with this approach is memorability [169]. Future directions of interest are thus to see whether there are ways to make some form of system-assigned password usable, particularly in terms of memorability. We are aware of three such approaches that we discuss further below: one that focuses on making system-assignment memorable (e.g., using mnemonics), another that encourages different mnemonic strategies that should produce more random passwords, and another to use persuasion to help users choose passwords that are more random.

It is possible that certain systems, e.g. types of graphical password, would naturally achieve more memorable system-assigned passwords. Indeed, Weinshall [161] found that a recognition-based system using system-assigned images had high success rates averaging around 95% (although only for a sample size of nine participants). However, this system has the usability problems of a long training and login time, and may suffer worse interference due to the large number of images used. To the best of our knowledge, there are no other documented evaluations of system-assigned graphical passwords. For system-assigned text passwords, one idea for increasing memorability is to also provide a corresponding mnemonic passphrase. Jeramyan et al. [73] propose using variations of newspaper headlines for this purpose, although the resulting memorability of their system has not yet been examined.

Mnemonic strategies to aid user's retention of stronger (but not system-assigned) passwords could also be used. One idea for graphical passwords was posited by Salehi [138], who suggested that a user provide a story to the system, and then the system fetches a picture (or sequence of pictures) that contains the story items. The user then selects the image(s) and click-points based on their pre-constructed story. If the users must create a story to begin password creation, they cannot skip the mnemonic story creation altogether, as was reported to be the case for 50% of users in the Story scheme ([34]; recall Section 3.2.2).

Inkblot authentication [145] is an existing variation of the idea of encouraging different mnemonic strategies; here the user is cued with abstract "inkblot" images in which people tend to see different things. The user enters a two-character association for each of 10 images shown, creating a theoretically more secure text password (recall Section 3.2.1). One of our related ideas for text passwords is to help users create customized passphrases, such that they would not be susceptible to guessing based on popular phrases available online [82]. If users create custom passphrases, based on existing user-specific knowledge, it might have similar memorability to more traditional passphrases and be less susceptible to automated attack. For example, one way of implementing this idea would be to provide a large set of story templates, whereby the user "customizes" the story such that it contains nouns and verbs that are meaningful to them. As long as each template had at least 4 items, and the user is asked to use at least two characters per item, the theoretical entropy of these passphrases might be less susceptible to attack than passphrases that users currently select. This sort of system must be carefully designed to ensure that the templates do not leak too much information, and that each user-defined noun or verb has more than a small number of likely possibilities, and could not be dramatically narrowed down by character frequency tables combined with the context.

Using persuasion to help users *choose* more random passwords is yet another possibility currently under investigation by Chiasson et al. [48, 23]. The general idea is that users are encouraged to choose passwords based on suggestions by the system, but are permitted to have the system suggest other possibilities if he/she does not like what the system last provided. For their example using CCP [23], the user is shown a viewport on a randomly placed part of the image, from which they are permitted to choose a click-point. If users do not like where the viewport is positioned, they can press a button to place the viewport over a new randomly selected part of the image. Forget et al. [48] also introduce the idea of using this concept for text passwords, whereby the user is displayed a password with some characters pre-assigned; the user then fills the remaining (blank) characters to complete the password.

Although the ideas discussed herein may prove useful if they are shown to improve both memorability and security, these methods are still susceptible to shoulder surfing and social engineering, and might suffer from users writing their passwords down (which in turn could lead to physical theft, e.g., through dumpster diving). To this end, we hope that our novel idea for user authentication called "pass-thoughts", described in Chapter 7 might prove less vulnerable to such problems, or motivate other solutions. There is an abundance of research required to determine how feasible this idea could be, as outlined in Section 7.8, which we hope to pursue in future work.

Bibliography

[1] M. Abadi, M. Burrows, C. Kaufman, and B. W. Lampson. Authentication and Delegation with Smart-cards. In *Theoretical Aspects of Computer Software*, pages 326–345, 1991.

[2] D. Asonov and R. Agrawal. Keyboard Acoustic Emanations. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 3–11, 2004.

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

[4] F. Attneave. Physical Determinants of the Judged Complexity of Shapes. *Journal of Experimental Psychology*, 53(4):221–227, 1957.

[5] L. Ballard, D. Lopresti, and F. Monrose. Forgery Quality and its Implications for Behavioral Biometric Security. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(5):1107–1118, 2007.

[6] L. Ballard, F. Monrose, and D. Lopresti. Biometric Authentication Revisited: Understanding the Impact of Wolves in Sheep's Clothing. In *Proceedings of the 15th Annual USENIX Security Symposium*, pages 29–41, 2006.

[7] BBC News. How Irises 'Reveal Personalities', 2007. `http://news.bbc.co.uk/1/hi/health/6375381.stm`.

[8] S. Bellovin and M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 72–84, 1992.

[9] F. Bergadano, B. Crispo, and G. Ruffo. High Dictionary compression for Proactive Password Checking. *ACM Transactions on Information and System Security*, 1(1):3–25, 1998.

[10] N. Bierbaumer, N. Ghanayim, T. Hinterberger, I. Iversen, B. Kotchoubey, A. Kubler, J. Perelmouter, E. Taub, and H. Flor. A Spelling Device for the Paralyzed. *Nature*, 398:297–298, 1999.

[11] N. Birbaumer and L. Cohen. Brain-Computer Interfaces: Communication and Restoration of Movement in Paralysis, March 2007.

[12] N. Birbaumer, A. Kubler, N. Ghanayim, T. Hinterberger, J. Perelmouter, J. Kaiser, I. Iversen, and B. Kotchoubey. The Thought Translation Device (TTD) for Completely Paralyzed Patients. *IEEE Transactions on Rehabilitation Engineering*, 8(2):190–193, 2000.

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

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

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

[16] P. Bourke. Determining Whether or Not a Polygon (2D) Has its Vertices Ordered Clockwise or Counterclockwise, 1998. `http://local.wasp.uwa.edu.au/~pbourke/geometry/clockwise/index.html`.

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

[18] X. Boyen. Halting Password Puzzles: Hard-to-break Encryption from Human-memorable Keys. In *16th USENIX Security Symposium*, 2007.

[19] Ian Britton (Reproduced by Permission of). Image Ref: 21-35-3. `http://www.freefoto.com`.

[20] S. Brostoff and A. Sasse. Are Passfaces more usable than passwords? A field trial investigation. In *Proceedings of HCI 2000*, pages 405–424, 2000.

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

[22] CERT Coordination Center. Vulnerabilities, Incidents, and Fixes. `www.cert.org`.

[23] S. Chiasson, A. Forget, R. Biddle, and P. van Oorschot. Influencing Users Towards Better Passwords: Persuasive Cued Click Points, 2007. School of Computer Science, Technical Report TR-07-16 `http://www.scs.carleton.ca/research/tech_reports/2007/download/TR-07-16.pdf`.

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

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

[26] L. S. Clair, L. Johansen, W. Enck, M. Pirretti, P. Traynor, P. McDaniel, and T. Jaeger. Password Exhaustion: Predicting the End of Password Usefulness. In *International Conference on Information Systems Security (ICISS)*, 2006.

[27] M. D. Corner and B. D. Noble. Zero-Interaction Authentication. In *International Conference on Mobile Computing and Networking*, pages 1–11, 2002.

[28] I. Corporation. Intel Core 2 Extreme Quad-Core Processor Qx6700 and Intel Core 2 Quad Processor Q6000 Sequence Datasheet, January 2007. `http://download.intel.com/design/processor/datashts/31559202.pdf`, site accessed March 25, 2007.

[29] L. Coventry. Usable Biometrics. In L. Cranor and S. Garfinkel, editors, *Security and Usability*, chapter 10, pages 97–120. O'Reilly, 2005.

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

[31] J. Daemen, R. Govaerts, and J. Vandewalle. Weak Keys for IDEA. In *Advances in Cryptology - CRYPTO 1993*, pages 224–231. Lecture Notes In Computer Science; Vol. 773, 1993.

[32] J. Daugman. How Iris Recognition Works. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):21–30, 2004.

[33] C. Davies and R. Ganesan. BApasswd: A New Proactive Password Checker. In *Proceedings of the 16th National Computer Security Conference*, pages 1–15, 1993.

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

[35] B. Defend and K. Fu. Personal communication. August 2007.

[36] D. Denning. *Cryptography and Data Security.* Addison-Wesley, 1982.

[37] J. Devore. *Probability and Statistics for Engineering and the Sciences.* Brooks/Cole Publishing Company, Pacific Grove, CA, USA, 4th edition, 1995.

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

[39] R. Dhamija, J. D. Tygar, and M. Hearst. Why Phishing Works. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2006)*, pages 581–590, 2006.

[40] A. Dirik, N. Memon, and J.-C. Birget. Modeling User Choice in the PassPoints Graphical Password Scheme. In *3rd Symposium on Usable Privacy and Security (SOUPS)*, 2007.

[41] E. Donchin, K. M. Spencer, and R. Wijesinghe. The Mental Prosthesis: Assessing the Speed of a P300-Based Brain-Computer Interface. *IEEE Transactions on Rehabilitation Engineering*, 8:174–179, 2000.

[42] M. Doppelmayr, W. Klimesch, T. Pachinger, and B. Ripper. Individual Differences in Brain Dynamics: Important Implications for the Calculation of Event-Related Brain Power, 1998.

[43] D. E. Duncan. Implanting Hope. *Technology Review: MIT's Magazine of Innovation*, 108(3):48–54, 2005.

[44] P. Dunphy and J. Yan. Do Background Images Improve Draw-A-Secret Graphical Passwords? In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS)*, 2007.

[45] T. Elbert, C. Pantev, C. Wienbruch, B. Rockstroh, and E. Taub. Increased Cortical Representation of the Fingers of the Left Hand in String Players. *Science*, 270:305–307, 1995.

[46] P. Felzenszwalb and D. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004. Code available from: `http://people.cs.uchicago.edu/~pff/segment/`.

[47] D. Florencio and C. Herley. A Large-Scale Study of Web Password Habits. In *Proceedings of the International World Wide Web Conference (WWW)*, 2007.

[48] A. Forget, S. Chiasson, and R. Biddle. Persuasion as Education for Computer Security. In *AACE World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (E-Learn)*, 2007.

[49] FreeImages.com (Photograph Courtesy of). Image ID: wm_asian_places_047. `http://www.freeimages.com`.

[50] FreeImages.com (Photograph Courtesy of). Image ID: wm_recreation_005. `http://www.freeimages.com`.

[51] Freeimages.co.uk (Photograph Courtesy of). Image ID: paperclips.jpg. `http://www.freeimages.co.uk`.

[52] Freeimages.co.uk (Photograph Courtesy of). Image ID: pcb04090023.jpg. `http://www.freeimages.co.uk`.

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

[54] J. Goldberg, J. Hagman, and V. Sazawal. Doodling Our Way to Better Authentication. In *Proceedings of the Conference on Human Factors and Computing Systems*, pages 868–869. ACM Press, 2002. CHI '02 extended abstracts on Human Factors in Computer Systems.

[55] P. Golle and D. Wagner. Cryptanalysis of a Cognitive Authentication Scheme (Extended Abstract). In *Proceedings of the IEEE Symposium on Security and Privacy*, 2007.

[56] L. Gong, M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting Poorly Chosen Secrets from Guessing Attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.

[57] S. Granger. Social Engineering Fundamentals, Part I: Hacker Tactics, 2001. `http://www.securityfocus.com/infocus/1527`, site accessed Mar. 22, 2005.

[58] g.Tec. g.MOBIlab, October 2007. Personal Communication. System description available at: `http://www.gtec.at/products/g.MOBIlab/gMOBIlab.htm`.

[59] J. A. Halderman, B. Waters, and E. W. Felten. A Convenient Method for Securely Managing Passwords. In *Proceedings of the 14th International World Wide Web Conference*, pages 471–479. ACM Press, 2005.

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

[61] G. Hayday. Security Nightmare: How Do You Maintain 21 Different Passwords?, 2002. `http://software.silicon.com/security/0,39024655,11036760,00.htm`.

[62] M. E. Hellman. A Cryptanalytic Time-memory Trade Off. *IEEE Transactions on Information Theory*, IT-26:401–406, 1980.

[63] N. J. Hopper and M. Blum. Secure Human Identification Protocols. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security – ASIACRYPT '01*. Springer-Verlag LNCS 2248, 2001.

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

[65] International Organization for Standardization (ISO). ISO 9241-11: Guidance on Usability, 1998.

[66] ISI Web of Knowledge, 2007. `http://isiwebofknowledge.com/`.

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

[68] D. P. Jablon. Strong Password-Only Authenticated Key Exchange. *ACM SIGCOMM Computer Communication Review*, 26(6):5–26, 1996.

[69] A. Jain, L. Hong, and S. Pankanti. Biometric Identification. *Communications of the ACM*, 43(2):90–98, 2000.

[70] A. Jain, A. Ross, and S. Prabhakar. Introduction to Biometric Recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):4–20, January 2004.

[71] W. Jansen, S. Gavrilla, V. Korolev, R. Ayers, and S. R. Picture password: A visual login technique for mobile devices. NIST Report - NISTIR7030, 2003.

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

[73] S. Jeyaraman and U. Topkara. Have the Cake and Eat it Too - Infusing Usability Into Text-Password Based Authentication Systems. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC)*, pages 473–482, 2005.

[74] M. Just and P. van Oorschot. Addressing the Problem of Undetected Signature Key Compromise. In *NDSS*, 1999.

[75] J. Kelsey, B. Schneier, C. Hall, and D. Wagner. Secure Applications of Low-Entropy Keys. *Lecture Notes in Computer Science*, 1396:121–134, 1997.

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

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

[78] K. Knopper. KNOPPIX - Live Linux Filesystem on CD. `http://www.knopper.net/knoppix/index-en.html`, site accessed March 12, 2007.

[79] P. D. Kovesi. MATLAB and Octave Functions for Computer Vision and Image Processing. Univ. Western Australia. Available from: `http://www.csse.uwa.edu.au/~pk/research/matlabfns/`.

[80] M. Kuhn and R. Anderson. Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations. In *Information Hiding: Second International Workshop*, 1998.

[81] M. Kumar, T. Garfinkel, D. Boneh, and T. Winnograd. Reducing Shoulder-surfing by Using Gaze-based Password Entry. In *3rd Symposium on Usable Privacy and Security (SOUPS)*, 2007.

[82] C. Kuo, S. Romanosky, and L. Cranor. Human Selection of Mnemonic Phrase-based Passwords. In *Proceedings of the 2nd Symposium on Usable Privacy and Security (SOUPS)*, pages 67–78, New York, NY, 2006. ACM Press.

[83] LC Technologies Inc. Eyegaze Systems. `www.eyegaze.com`, site accessed Mar. 22, 2005.

[84] H. Lei and V. Govindaraju. A Comparative Study on the Consistency of Features in On-Line Signature Verification. *Pattern Recognition Letters*, 26:2483–2489, 2005.

[85] S. Li and H.-Y. Shum. Secure Human-Computer Identification (Interface) Systems Against Peeping Attacks: SecHCI, 2005. Report 2005/268, `http://eprint.iacr.org/`.

[86] D. P. Lopresti and J. D. Raim. The Effectiveness of Generative Attacks on an Online Handwriting Biometric. In *Audio- and Video-Based Biometric Person Authentication (AVBPA)*, volume 3546/2005, pages 1090–1099. Springer-Verlag, 2005.

[87] S. J. Luck and E. K. Vogel. The Capacity of Visual Working Memory for Features and Conjunctions. *Nature*, 390:279–281, 1997.

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

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

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

[91] T. Matsumoto, H. Matsumoto, K. Yamada, and S. Hoshino. Impact of Artificial "Gummy" Fingers on Fingerprint Systems. In Rudolf L. van Renesse, editor, *SPIE Optical Security and Counterfeit Deterrence Techniques IV*, volume 4677, pages 275–289, Apr. 2002.

[92] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[93] J. R. Millan, J. Mourino, M. Franze, F. Cincotti, M. Varsta, J. Heikkonen, and F. Babiloni. A Local Neural Classifier for the Recognition of EEG Patterns Associated to Mental Tasks. *IEEE Transactions on Neural Networks*, 13(3):678–686, 2002.

[94] R. B. Miller. Response Time in Man-Computer Conversational Transactions. In *Proceedings of the AFIPS Fall Joint Computer Conference*, volume 33, pages 267–277, 1968.

[95] K. D. Mitnick. *The Art of Deception: Controlling the Human Element of Security*. John Wiley & Sons, 1 edition, 2001.

[96] W. Moncur and G. Leplâtre. Pictures at the ATM: Exploring the Usability of Multiple Graphical Passwords. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 887–894, New York, NY, USA, 2007. ACM Press.

[97] F. Monrose. *Towards Stronger User Authentication*. PhD thesis, NY University, 1999.

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

[99] F. Monrose, M. K. Reiter, Q. Li, and S. Wetzel. Cryptographic Key Generation From Voice. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2001.

[100] F. Monrose, M. K. Reiter, and S. Wetzel. Password Hardening based on Keystroke Dynamics. *International Journal of Information Security*, 1(1):69–83, 2001.

[101] M. Montoro. Cain & Abel. `http://www.oxid.it/cain.html`, site accessed Feb. 24, 2007.

[102] R. Morris and K. Thompson. Password Security: A Case History. *Communications of the ACM*, 22(11):594–597, 1979.

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

[104] J. Nakajima and M. Matsui. Performance Analysis and Parallel Implementation of Dedicated Hash Functions. In *Advances in Cryptology - EUROCRYPT 2002*, pages 165–180. Springer-Verlag LNCS 2332, 2002.

[105] D. Nali and J. Thorpe. Analyzing User Choice in Graphical Passwords. Tech. Report TR-04-01, School of Computer Science, Carleton University, Canada, http://www.scs.carleton.ca/research/tech_reports/2004/TR-04-01.pdf, 2004.

[106] A. Narayanan and V. Shmatikov. Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS)*, pages 364–372, 2005.

[107] Neurosky. Neurosky Home Page. http://www.neurosky.com, site accessed Oct. 31, 2005.

[108] Objectif Sécurité. Ophcrack. http://ophcrack.sourceforge.net, site accessed Feb. 24, 2007.

[109] P. Oechlin. Making a Faster Cryptanalytic Time-Memory Trade-Off. In *Advances in Cryptology - CRYPTO 2003*, pages 617–630. Springer-Verlag LNCS 2729, 2003.

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

[111] P. van Oorschot and S. Stubblebine. On Countering Online Dictionary Attacks with Login Histories and Humans-in-the-Loop. *ACM Transactions on Information and System Security*, 9(3):235–258, August 2006.

[112] P. van Oorschot and J. Thorpe. On the Security of Graphical Password Schemes (Extended Version), 2005. Tech. Report TR-05-11, School of Computer Science, Carleton University, Canada, http://www.scs.carleton.ca/research/tech_reports/2005/download/TR-05-11.pdf.

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

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

[115] Openwall Project. Wordlists, 2004. http://www.openwall.com/passwords/wordlists/, site accessed Jan.7 2004.

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

[117] R. Palaniappan and K. V. R. Ravi. A New Method to Identify Individuals Using Signals from the Brain. In *Proceedings of the International Conference on Information Computer Security*, pages 1442–1445, 2003.

[118] R. Paranjape, J. Mahovsky, L. Benedicenti, and Z. Koles. The Electroencephalogram as a Biometric. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, pages 1363–1366, 2001.

[119] passcracking.com. MD5 Online Cracking Using Rainbow Tables. `http://www.passcracking.com`, site accessed Feb. 24, 2007.

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

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

[122] Personal Photograph by the Author. Bee in Garden, 2006.

[123] M. Peters, B. Laeng, K. Latham, M. Jackson, R. Zaiyouna, and C. Richardson. A Redrawn Vandenberg and Kuse Mental Rotations Test: Different Versions and Factors That Affect Performance. *Brain and Cognition*, 28:39–58, 1995.

[124] Photobucket.com (Photograph Courtesy of). Image ID: 11_12_1_web. `http://i26.photobucket.com/albums/c136/hamm239/11_12_1_web.jpg`.

[125] B. Pinkas and T. Sander. Securing Passwords Against Dictionary Attacks. In *9th ACM Conference on Computer and Communications Security (CCS)*, pages 161–170. ACM Press, 2002.

[126] R. Plamondon and S. N. Srihari. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[127] A. Power, E. Lalor, and R. Reilly. Can Visual Evoked Potentials be used in Biometric Identification? In *Proceedings of the 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS '06)*, pages 5575–5578, 2006.

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

[129] Rainbowcrack-Online. `http://www.rainbowcrack-online.com`, site accessed Feb. 24, 2007.

[130] Rainbowtables.net. `http://www.rainbowtables.net`, site accessed Feb. 24, 2007.

[131] Real User Corporation. About Passfaces. `http://www.realuser.com`, site accessed May 24, 2004.

[132] Real User Corporation. The Science Behind Passfaces, 2004. `http://www.realuser.com/published/ScienceBehindPassfaces.pdf`, site accessed March 18, 2007.

[133] K. Renaud. Evaluating Authentication Mechanisms. In L. Cranor and S. Garfinkel, editors, *Security and Usability*, chapter 6, pages 97–120. O'Reilly, 2005.

[134] S. Riley. What Users Know and What They Actually Do. *Usability News*, 8(1), February 2006. `http://psychology.wichita.edu/surl/usabilitynews/81/Passwords.htm`, accessed March 10, 2007.

[135] A. R. Roddy and J. D. Stosz. Fingerprint Features - Statistical Analysis and System Performance Estimates. *Proceedings of the IEEE*, 85(9):1390–1421, 1996.

[136] V. Roth, K. Richter, and R. Freidinger. A PIN-Entry Method Resilient Against Shoulder Surfing. In *11th Conference on Computer and Communications Security (CCS)*, pages 236–245, 2004.

[137] RSA. RSA SecurID. `http://www.rsa.com/node.aspx?id=1156`, site accessed March 27, 2007.

[138] A. Salehi. Personal communication, September, 2007.

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

[140] C. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[141] Z. Shuanglei. Project RainbowCrack. `http://www.antsight.com/zsl/rainbowcrack`, site accessed Feb. 24, 2007.

[142] John Siracusa (Reproduced by Permission of). Clutter: About 130 Windows. `http://arstechnica.com/reviews/os/macosx-10.3.ars/5`.

[143] E. Spafford. Crisis and Aftermath (The Internet Worm). *Communications of the ACM*, 32(6):678–687, 1989.

[144] E. H. Spafford. OPUS: Preventing Weak Password Choices. *Computers and Security*, 11(3):273–278, 1992.

[145] A. Stubblebine and D. Simon. Inkblot Authentication, 2004. Microsoft Technical Report MSR-TR-2004-85.

[146] H. Tao. Pass-Go, a New Graphical Password Scheme. Master's thesis, School of Information Technology and Engineering, University of Ottawa, Canada, 2006.

[147] The Shmoo Group. Rainbow Tables. `http://rainbowtables.shmoo.com`, site accessed Feb. 24, 2007.

[148] Thomas Hawk (Photograph Courtesy of). Flickr Photo Download: Where I've Been Lately. `http://www.flickr.com/photo_zoom.gne?id=96968793&size=o`.

[149] J. Thorpe and P. van Oorschot. Graphical Dictionaries and the Memorable Space of Graphical Passwords. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[150] J. Thorpe and P. van Oorschot. Towards Secure Design Choices for Implementing Graphical Passwords. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC)*. IEEE, 2004.

[151] J. Thorpe and P. van Oorschot. Human-Seeded Attacks and Exploiting Hot Spots in Graphical Passwords. In *Proceedings of the 16th USENIX Security Symposium*, 2007.

[152] J. Thorpe and P. van Oorschot. Human-Seeded Attacks and Exploiting Hot Spots in Graphical Passwords, February 2007. Tech. Report TR-07-05, School of Computer Science, Carleton University, Canada, `http://www.scs.carleton.ca/research/tech_reports/2007/download/TR-07-05.pdf`.

[153] J. Thorpe, P. van Oorschot, and A. Somayaji. Pass-thoughts: Authenticating With Our Minds. In *Proceedings of the New Security Paradigms Workshop (NSPW)*, 2005.

[154] C. Tyler. Human Symmetry Perception. In C. Tyler, editor, *Human Symmetry Perception and its Computational Analysis*, pages 3–22. VSP, The Netherlands, 1996.

[155] U. Uludag and A. Jain. Attacks on Biometric Systems: A Case Study in Fingerprints. *Proc. SPIE-EI 2004, Security, Steganography and Watermarking of Multimedia Contents VI*, 5306:622–633, 2004.

[156] T. M. Vaughan, W. J. Heetderks, L. Trejo, W. Z. Rymer, M. Weinrich, M. M. Moore, A. Kubler, B. H. Dobkin, N. Birbaumer, E. Donchin, E. W. Wolpaw, and J. R. Wolpaw. Brain-Computer Interface Technology: A Review of the Second International Meeting, 2003.

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

[158] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems for Security. In *Advances in Cryptology - EUROCRPYT 2003*. Springer-Verlag LNCS 2656, 2003.

[159] L. von Ahn, R. Liu, and M. Blum. Peekaboom: A Game for Locating Objects in Images. In *Conference on Human Factors in Computing Systems (CHI)*, 2006.

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

[161] D. Weinshall. Cognitive Authentication Schemes Safe Against Spyware (short paper). In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 295–300, 2006.

[162] S. Wiedenbeck, J. Waters, J. Birget, A. Brodskiy, and N. Memon. Authentication Using Graphical Passwords: Basic Results. In *Human-Computer Interaction International (HCII)*, 2005.

[163] S. Wiedenbeck, J. Waters, J. Birget, A. Brodskiy, and N. Memon. Authentication Using Graphical Passwords: Effects of Tolerance and Image Choice. In *Proceedings of the 1st Symposium on Usable Privacy and Security (SOUPS)*, 2005.

[164] S. Wiedenbeck, J. Waters, J. Birget, A. Brodskiy, and N. Memon. PassPoints: Design and Longitudinal Evaluation of a Graphical Password System. *International Journal of Human-Computer Studies (Special Issue on HCI Research in Privacy and Security)*, 63:102–127, 2005.

[165] S. Wiedenbeck, J. Waters, L. Sobrado, and J.-C. Birget. Design and Evaluation of a Shoulder-Surfing Resistant Graphical Password Scheme. In *AVI '06: Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 177–184, 2006.

[166] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan. Brain-Computer Interfaces For Communication and Control. *Clinical Neurophysiology*, 113:767–791, 2002.

[167] T. Wu. A Real-World Analysis of Kerberos Password Security. In *Proceedings of the Network and Distributed System Security Symposium*, 1999.

[168] J. Yan. A Note on Proactive Password Checking. In *Proceedings of the New Security Paradigms Workshop (NSPW)*, 2001.

[169] J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password Memorability and Security: Empirical Results. *IEEE Security and Privacy Magazine*, 2(5):25–31, 2004.

[170] L. Zhuang, F. Zhou, and J. D. Tygar. Keyboard Acoustic Emanations Revisited. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS)*, 2005.

Chapter 9

Appendices

9.1 Appendix A - Results for DAS$_J$ Password Subset Sizes

Tables 9.1, 9.2, and 9.3 give sample results computed using the method outlined in Section 5.3.2 for various values of L_{max} and X, under different grid sizes. Values given are log_2(number of passwords). The size of the full DAS$_J$ password space (without a limitation on the number of strokes) was double-checked and essentially agrees with the results given in [72].

Tables 9.4 and 9.5 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 9.6 and 9.7 give results for the effect of having no dots or strokes ≤ 2 in a DAS password. The data in Tables 9.4 to 9.7 are for 5×5 grids.

L_{max} \ X	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 9.1: Bit-size of DAS$_J$ graphical password space. Values are given for total length at most L_{max} with at most X strokes on a 6×6 grid.

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

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

X	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
L_{max}																				
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 9.4: Bit-size of DAS$_J$ graphical password space when limited to strokes of length 1 and 2. Values are given for total length at most L_{max} with at most X strokes on a 5×5 grid.

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

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

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

9.2 Appendix B - Other Images Used in Lab Study

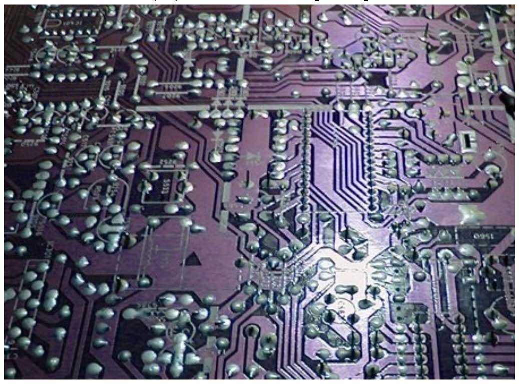
(a) *truck* [49]

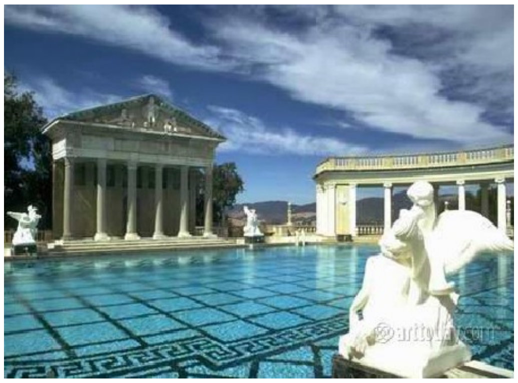
(b) *paperclips* [51]

(c) *bee* [122]

(d) *cdcovers* [142]

(e) *smarties* [124]

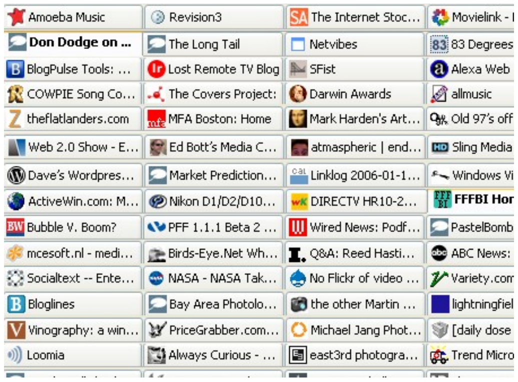

(f) *pcb* [52]


(g) *corinthian* [50]

(h) *tea* [163]

(i) *philadelphia* [163]

(j) *mural* [163]



Amoeba Music	Revision3	The Internet Stoc…	Movielink - I
Don Dodge on ...	The Long Tail	Netvibes	83 Degrees
BlogPulse Tools: …	Lost Remote TV Blog	SFist	Alexa Web
COWPIE Song Co…	The Covers Project:	Darwin Awards	allmusic
theflatlanders.com	MFA Boston: Home	Mark Harden's Art…	Old 97's off
Web 2.0 Show - E…	Ed Bott's Media C…	atmaspheric | end…	Sling Media
Dave's Wordpres…	Market Prediction…	Linklog 2006-01-1…	Windows Vi
ActiveWin.com: M…	Nikon D1/D2/D10…	DIRECTV HR10-2…	FFFBI Hor
Bubble V. Boom?	PFF 1.1.1 Beta 2 …	Wired News: Podf…	PastelBomb
mcesoft.nl - medi…	Birds-Eye.Net Wh…	Q&A: Reed Hasti…	ABC News:
Socialtext -- Ente…	NASA - NASA Tak…	No Flickr of video …	Variety.com
Bloglines	Bay Area Photolo…	the other Martin …	lightningfiel
Vinography: a win…	PriceGrabber.com…	Michael Jang Phot…	[daily dose
Loomia	Always Curious - …	east3rd photogra…	Trend Micro

(k) *icons* [148]

Figure 9.1: Subset of images used in the lab study. See Figure 6.4 for *cars* and *pool*. The remaining four images used (*citymap-nl*, *citymap-gr*, *faces*, and *toys*) are available from the author; we were not able to obtain permission to reproduce them herein.