Towards Stronger User Authentication

by

Newman Fabian Monrose

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

May 1999

Approved: ____________________________

Zvi Kedem

To Leah ("Peewee") & Nina.

Acknowledgements

First and foremost, I express my gratitute to Aviel D. Rubin for his friendship, guidance, and continual support throughout my studies. There were many times when I contemplated leaving prior to completing my studies, and without his support and encouragement, I am certain that I would not have reach thus far. I can not thank him enough.

I am also indebted to Zvi Kedem for providing me with assistance and direction whenever I needed it. I am grateful for his support of my research and the leadership he provided on numerous occasions. I am also thankful for his friendship.

Deciding to not return home immediately after completing my studies was a long and difficult process. I express special thanks to my mother and my sister for providing emotional support and believing in me all this time. I am especially appreciative of the support of my sister whose friendship I value tremendously. I am thankful for Leah, Nina, and Melanie who inspired me over the past few years. I am especially thankful for Nina who provided a sense of inner calm during difficult times.

I also thank the USENIX Scholars Program, Bell Communications Research, the Computer Science department, and my father for providing the financial support needed to complete my studies. I also express my gratitude to Michael K. Reiter and Alain Mayer of Bell Labs, Lucent Technologies, for their interests in my research. I would especially like to thank Mike for providing inspiration and direction, and for helping me focus on finishing my degree. I also thank my other committee members, Vijay Karamcheti and Arthur Goldberg.

I am thankful for the many friends with whom I share more than just an academic relationship. I cherish the friendships I share with each one of them. I extend a special thank-you to Ian Jermyn, who assisted me in more ways than one. Thanks, as well, to Richard Cole for listening to my concerns over the years and providing his assistance whenever possible. I would like to express my appreciation to Arash Baratloo, Karen Culler, Sabine French, Martin Garcia, Davi Geiger, Deepak Goyal, Raju Jawalekar, Bob Hummel, Hiroshi Ishikawa, Ayali Itzkovitz, Leslie Palanker, Laxmi Parida, Bill Pink, Madhu Nayakkankuppam, Tyrone McGhee, Fritz Orneas, Archisman Rudra, Peter Wyckoff, the Courant system administrators, and the Didier family. Warmest thanks to Jay Sachs for providing a LaTeX template compliant with NYU's thesis requirements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Weak passwords are a fact of life—Bellovin and Merrit.

Password-based authentication is the dominant mechanism for verifying the identity of computer users, even though it is well known that people frequently choose passwords that are vulnerable to dictionary attacks. This dissertation addresses the issue of improving the security of password-based authentication and presents authentication techniques that are more secure than traditional approaches against password guessing attacks.

The motivation for addressing the security and shortcomings of traditional password-based authentication is that users tend to choose passwords that are easy to remember, which in the case of textual passwords usually implies that they are easy to obtain by searching through a carefully formed "dictionary" of candidate passwords. For example, in one case study of over 14,000 UNIX^{TM} passwords, almost 25% of the passwords were found by searching for words in a dictionary of only 3×10^6 words [45]. This relatively high success rate is not unusual despite the fact that there are roughly 2×10^{14} 8-character passwords consisting of digits and upper and lower case letters alone.

1.1 Difficulty of User Authentication

The fact that people choose passwords that are susceptible to dictionary attacks has significant impact on the design of secure authentication schemes. In traditional authentication schemes a password is a string of characters (usually of length six or more) that a user, say Alice, is capable of committing to memory. When Alice logs into a host, demonstration of knowledge of the secret (i.e., the password) shared between Alice and the host, is accepted as proof of Alice's identity. In the classic UNIX method for authentication, the host validates an image of the password under a one-way function so that the password itself is never presented in the clear. Such authentication schemes that rely on time-invariant passwords are commonly referred to as providing *weak* (user) authentication.

Over the past 20 years the security of time-invariant passwords for authentication has been examined by numerous researchers, notably [63, 45, 25, 82, 90]. Without exception, these studies reiterate the fact that left to their own accord, people choose passwords that are easy to find by automated search. The basic attacks that authentication protocols need to guard against are therefore:

- Password guessing attack. Here, the attacker, Mallot, is assumed to have access to a dictionary containing some permutation of common choices of passwords. There are primarily two ways in which Mallot can use the dictionary:

On-line attack. Mallot repeatedly picks a password from the dictionary and tries to use it in order to impersonate a legitimate user. If the impersonation fails, the password is eliminated from the dictionary and Mallot tries again.

Off-line attack. Mallot records past communication or captures all information related to authentication (for example, gets hold of `/etc/password` on UNIX), and searches through the dictionary for a password which is consistent with the captured information.

- Man-in-the-middle attack. Mallot is an active participant in the protocol and can intercept and insert messages of his own.
- Replay attack. Mallot records messages and resends them at a later time.

In practice, the threat of on-line attacks can be reduced by limiting the number of failed logins that a user is allowed to have within a predetermined time window. Off-line guessing attacks, on the other hand, have been surprisingly effective (see [63, 45, 25, 82, 90]).

To thwart dictionary attacks, it is now common practice for system administrators to invoke reactive password checkers [73, 64] on an existing collection of user chosen passwords to identify weak password choices, or to use proactive checkers [8, 81] to filter out certain classes of poorly chosen passwords when the user inputs her password for the first time. Password checkers successfully increase the uncertainty (that is, the entropy rather than simply the length) of users' passwords, thus making them move beyond the reach of dictionary and exhaustive search attacks, but the resulting passwords are not necessarily easy for users to remember.

Password guessing attacks are still likely to succeed when conducted on a large collection of passwords, even if proactive and reactive checkers are being used. An approach to increasing further the work factor for off-line guessing attacks is to append an n-bit random number (called the *salt*) to a user's password and then to "encrypt" the concatenated string using a one-way hash function. Both the encrypted string and the salt are stored in a password table on the authentication server. When a user tries to authenticate and enters her password, the salt is retrieved from the password table, prepended to the password, and the concatenated string encrypted. The result is compared to that stored in the password table, and authentication succeeds if they match. The difficulty of exhaustive search on any particular user's password is unchanged by salting since the salt is given in clear-text in the password file. However salting increases the complexity of a dictionary attack against a large set of passwords simultaneously, by requiring any precomputed dictionary of hashed passwords to contain 2^n variations of each trial password. This imposes more time for the preparation, and larger memory requirements for the storage of the precomputed dictionary. Salting also ensures that users with the same textual password will have different encrypted entries in the password table.

The problem with password salting is that the hashed passwords are vulnerable to attack when the adversary captures all information relevant to login (for example, if the attacker obtains `/etc/password` on UNIX systems which includes the salt values). One way of eliminating this vulnerability is not to store the salt, but instead to have the login program exhaustively search for it [56]. During a login sequence the user inputs the password and the system searches for the n-bit random addition by trying all possible values for the salt. This variation increases the search space for the attacker by a factor of 2^n even if he captures all the stored information related to password authentication. Unfortunately, this results is a significant additional computational overhead on each login if n is large, and hence a longer login response time for the user.

The security of user-chosen passwords can be improved by allowing users to type in a phrase or sentence, called a pass-phrase, rather than a standard password. The use of pass-phrases allows for greater entropy without imposing too much burden on users' memory capacity. The intuition is that users can remember phrases more easily than random character sequences, and therefore, users should pick pass-phrases from a more uniform distribution. The pass-phrases are hashed down to fixed sized values and used as before. If the pass-phrase is long enough, the resulting hashed value will be random, but exactly what long enough means is open to interpretation. An additional drawback is that it can be difficult to type an entire pass-phrase during the login sequence with the echo turned off [77].

1.2 Our Work

In the general setting addressed in this dissertation, off-line guessing attacks are the major concern. Since the schemes put forth in this dissertation focus on user authentication via local login devices, our techniques, by design, are not resistant against replay and man-in-the-middle attacks. However, as shown later, our schemes can be combined in natural ways with existing strong authentication and key exchange protocols [9, 10, 31, 41, 53, 89] that defend against these attacks.

1.2.1 Keystroke Dynamics

The first approach that we propose for improving the security of standard password-based authentication directly addresses the problem of the additional computational overhead associated with salting techniques as the salt gets large, for example greater than 12 bits (i.e., the current size of the salt used in the conventional UNIX authentication). We propose a technique for strengthening the security of a textual password by augmenting it with biometric information. Biometric information such as the duration and latency of the keystrokes used during entry of the password is used to *strengthen* the password. Thereby, both the password and the user's typing pattern are used to corroborate the identity of the user. Intuitively, the technique presented can be used to improve salting by determining some or all of the salt bits using the user's keystroke features. In this way, some of the computational burden can be lifted from the login machine, or alternatively the size of the salt can be enlarged for greater security. Furthermore, our approach is complementary to the use of pass-phrases.

Specifically, the technique presented generates a strengthened password based on both the characters typed and the way the user types her password. This is done not by concatenating the password with bits generated from the user's features, but by using the password to decrypt a table of information. The user's keystroke features are then used to index into the table to select information from which the strengthened password is computed. The strengthened password can then be verified for login purposes by attempting to decrypt a file containing recognizable plaintext. Login succeeds if and only if the decryption is successful. Our technique imposes a multiplicative work factor on an attacker conducting an off-line dictionary attack, because upon decrypting the table with a guessed password, the attacker cannot immediately determine if the guessed password was correct. Instead, the attacker must sample elements of the table and generate corresponding strengthened passwords until he either finds the correct one or determines that the guessed password was incorrect. The work factor for an off-line attacker therefore grows with the number of keystroke features that the user reliably repeats. Moreover, an on-line attacker will fail to log if he cannot mimic the legitimate user's keystroke typing patterns.

The motivation for using keystroke features to strengthen password-based authentication comes from numerous research efforts that validate the hypothesis that certain keystroke features are highly repeatable and that significant variation exist between users [26, 86, 48, 49, 4, 43]. These studies, as well as our own (presented in Chapter 4), confirm that keystroke features are an effective means to distinguish users. This dissertation, however, is the first work that uses keystroke features to generate a repeatable key for use in authentication.

We argue that the use of features of a user's normal typing rhythm to improve password-based authentication is a natural choice for computer security. This argument stems from observations that similar neuro-physiological factors that make hand written signatures unique are exhibited in a user's typing pattern [86, 28, 59]. When a person types, the time intervals between successive keystrokes, the length of time that keystrokes are held down, and the applied pressure on the keys can be used to construct a unique "signature" for that individual. For well-known, regularly typed strings, such signatures have been shown to be consistent. Furthermore, recognition based on typing rhythm is not intrusive, making it applicable to computer access security as users will be typing at the keyboard anyway.

Challenges

The first and foremost challenge is to validate whether keystroke features can reliably be used for identification purposes across a diverse set of users. Prior work in this area [26, 86, 48, 49, 4, 43] has been conducted on relatively small sample sets and validating their results on a larger sample set taken from users of varying ages, nationalities and backgrounds is important. Moreover, much of the work on keystroke verification uses the key down-to-down time as the base unit of measure, but this measure may be further delineated into two components—the total time for which the first key is depressed (keystroke duration), and the time between the release of a key and the depression of the next one (keystroke latency). An analysis of these two components, with regard to the variation across the population, is important.

Given a set of features that a user reliably repeats (with some small degree of variation), the challenge remains to generate the correct strengthened password. At the same time, the attacker who captures system information used to generate or verify strengthened passwords should be unable to determine which features are relevant to generating a user's strengthened password, since revealing this information could reveal information about the characters related to that password feature. For example, suppose the attacker learns that the time between the first and second keystrokes is a feature that is reliably repeated by the user and thus is used in the authentication process. This may reveal information about the first and second characters of the password, since the layout of the keyboard makes some character pairs more amenable to reliable latency repetitions than others.

An important class of attack that must therefore be protected against is the adversary who captures all the information stored related to authentication, and then uses this information in an off-line effort to find the strengthened password or the correct string of characters of the password field as entered by the user. A basic requirement is that any such attack be at least as difficult as exhaustively searching for the password in a traditional UNIX setting where the attacker has a copy of `/etc/passwd`. In particular, if Alice's password is difficult for an attacker to find using a dictionary attack, then the strengthened password must be at least as secure. A more ambitious challenge is to increase by a considerable amount the work that the attacker must undertake, even if Alice chooses her password poorly, i.e., in a way that is susceptible to a dictionary attack.

As mentioned earlier, our scheme offers additional security against an on-line attacker who, for example, learns Alice's password (e.g., by observing her type it) and attempts to login to her account. Unless the attacker can type like Alice, the attacker will fail. However, this implies that if Alice types her password abnormally, her login will also fail. Therefore, it is important to analyze empirically not only the security of our scheme, but also the ability of legitimate users to reliably log in to their own accounts.

Contributions

Keystroke dynamics and password strengthening We present a set of template-based models for evaluating the thesis that users exhibit individualistic patterns in their typing behavior. Moreover, we present empirical results based on a diverse group of users that validate some of the prior work in this area, but that also point out some of the inherent limitations with using a behavioral trait (rather than an anatomical characteristic such as a fingerprint) as a sign of identity. We examine the more crucial of these limitations and discuss the ramifications for computer security.

Noting one of the more severe limitations of using keystroke dynamics as a method of authentication, that different keyboards may change a users' typing pattern significantly, we present a technique for improving the security of textual password-based authentication which performs well when restricted to logins using the same keyboard (for example, login to a laptop). The technique is significantly more secure against attacks than conventional password-based authentication. Moreover, our technique does not reveal which of a user's keystroke features are used to generate the corresponding strengthened password, and is the first to offer stronger security against *both* on-line and off-line attacks. Furthermore, the technique is unintrusive and works with off-the-shelf keyboards.

The scheme presented is initially as secure as a normal password scheme. Over time it adapts to the user's typing patterns, gradually strengthening the password with biometric information. Moreover, while fully able to adapt to gradual changes in a user's typing pattern, our scheme can be used to generate the *same* strengthened password indefinitely. In this regard, our scheme is the first to generate a repeatable secret based on a password and its keystroke dynamics that is stronger than the password itself, and that can be used in applications other than login (for example, file encryption).

A major contribution of this dissertation is in analyzing the security of our technique relative to conventional approaches. Through empirical analysis of user keystroke data, we show that our scheme improves upon the security of conventional passwords. Specifically, we show that finding a user's normal password is not made easier in our scheme than it is in a typical environment where login access is determined by testing the hash of the password against a previously stored hash value. More importantly, we show that the cost to an attacker of finding the strengthened password is generally greater, by a significant multiplicative factor, than that of finding the original password in a conventional scheme. An additional contribution is the empirical analysis of a user's ability to log in reliably to her own account when our scheme is used for authentication.

1.2.2 Graphical Passwords

Given the known limitations of (textual) passwords for authentication, it is only prudent to seek new directions in user authentication. The second approach presented here provides one such direction. We propose methods for achieving better security than conventional password schemes by exploiting the input features available with graphical devices such as Personal Digital Assistants (PDAs). Specifically, we propose and evaluate *graphical passwords*, which serve the same purpose as textual passwords, but consist of handwritten drawings, possibly in addition to text. Graphical passwords derive their strength from the fact that graphical input devices allow one to decouple the positions of the inputs from the temporal order in which those inputs occur. We use this independence to build new password-based authentication schemes that are convincingly stronger than conventional methods.

The most compelling reason for exploring the use of a picture-based password scheme is that human beings seem to possess a remarkable ability for recalling line drawings and real objects. The *picture effect*, that is, the superiority of pictorial and object representations on a variety of measures of learning and memory has been studied for decades [16, 80, 69, 83, 14]. Cognitive scientists and psychologists have shown that there is a substantial improvement of performance in recall and recognition with pictorial representations of to-be-remembered material than for abstract and concrete words. Numerous studies exhibiting strikingly high differences in memory recall of pictures over words can be found in the research literature (see for example [80, 83, 67, 13]). In this dissertation we explore an approach to user authentication that generalizes the notion of a textual password and that, in many cases, improves the security of user authentication over that provided by textual passwords. Furthermore our biometric technique for strengthening the security of textual password-based authentication can be combined in natural ways with password checkers and salting to improve the security of graphical passwords as well.

Challenges

The main challenge in developing graphical password schemes is to evaluate their security relative to that of textual password-based alternatives. An obvious approach is to calculate the information content of the graphical password space as the entropy of the probability distribution over that space, and compare that to the entropy of the textual password space. However, given that the distribution of graphical password choices is not known (since this is a new scheme), that approach is not possible. Hence, a challenge remains in modeling sets of *memorable* graphical passwords, and showing that these sets, or some subset of them, have cardinality greater than the dictionary of textual passwords from which users typically choose.

We argue that due to the dependence of the security of a scheme on the passwords that users choose in practice, a new password scheme can not be *proven* better than an old scheme, since there is no *a priori* knowledge of the distribution of passwords that will be chosen by users under the new scheme. Performing trials on users in order to learn the distribution of password choices for a new scheme is impractical for such large sample spaces. In the case of textual passwords, learning the knowledge that attackers routinely use would correspond to trying to learn the English dictionary given no prior knowledge of the types of letter combinations used in English, by having subjects type in 8-character passwords—a feat which we argue is not feasible. In the absence of such objective proof, the challenge is in presenting plausible arguments for reasoning about the security of graphical password schemes relative to conventional textual-based password alternatives.

Furthermore, given the ease with which graphical devices such as PDAs can be captured, an important criterion is that the passwords themselves not be stored on the device. This restriction is important as it protects both the password and any encrypted content from an attacker if the graphical device is captured.

Contributions

We analyze the security of graphical passwords[1] relative to that of textual passwords. To this end, we present two graphical password schemes that are more secure than textual passwords (and more secure than the scheme of [12]), and employ novel analysis techniques to make this argument.

The first graphical password scheme presented builds directly on textual password schemes, by enhancing the input of textual passwords using graphical techniques. In this case, if we assume the same underlying distribution of passwords in the old and new schemes, then the graphical password is at least as strong as the textual password scheme that underlies it, and even a conservative estimate of the variations introduced by the graphical input yields a substantial improvement in strength over the purely textual version.

We propose and implement a second scheme, called *Draw-a-Secret* (DAS), that is purely graphical: the user draws a secret design (the password) on a grid. To reason about improvements in security over textual passwords, we define a class of DAS passwords that, plausibly, are memorable for users. This class consists of those passwords that can be generated by a short program in a simple grid language. We do not argue that every memorable password has a short program to describe it, but that passwords described by short programs are memorable. We present the grammar for our language and show that even the set of memorable DAS passwords characterized by programs of small complexity has cardinality larger than the dictionaries of textual passwords to which a high percentage of

[1]Graphical passwords were first introduced by Blonder [12] but we considerably advance the concept in both theory and practice.

textual passwords belong.

1.3 Dissertation Outline

In Chapter 2 we examine elementary concepts in user authentication. In Chapter 3 we review some of the fundamental work that supports the ideas put forth in this dissertation. We give an overview of biometrics and examine why some biometrics, like fingerprints, are better candidates for identity verification than others.

The idea of using features of a user's typing rhythm to generate strengthened passwords came about as a direct result of preliminary work on building an authentication system using keystroke dynamics [61]. (Keystroke dynamics is the process of analyzing the way a user types at a terminal by monitoring the keyboard inputs thousands of times per second, and attempting to identify users based on habitual rhythmic patterns in the way they type.) In Chapter 4 we present empirical evidence which supports the use of keystroke dynamics for authentication in restricted environments, and examine some classification and identification strategies. We present our observations and findings and compare them with prior work in this area.

In Chapter 5 we present our technique for strengthening the security of textual passwords. The technique effectively hides information about which of a user's features are relevant to authentication, even from an attacker who captures all system information related to authentication. Furthermore, an on-line attacker will be unable to log in as a legitimate user unless he can mimic that user's typing pattern. We also show that the technique presented is viable in practice in the sense that a user can still log into her own account reliably.

In Chapter 6 we present an alternative technique for user authentication based on the use of graphical passwords. We model sets of graphical passwords, and show that the resulting password spaces are convincingly more difficult to attack than conventional textual-based based approaches. In Chapter 7 we make closing remarks and discuss areas of future work.

Chapter 2

User Authentication

People pick bad passwords, and either forget, write down, or resent good ones—Bellovin and Merritt.

When a user attempts to logs into a host computer, how does the host know who the user is? How does the host know that is it not an attacker impersonating a legitimate user? Traditionally, authentication techniques that rely on the possession of a physical object, such as an identification card, or something the user shares with the host, such as a password, have been used to solve this problem.

Authentication is the process whereby one party, the *verifier*, is assured (through acquisition of corroborative evidence) of the identity of a second party, the *claimant*, involved in a protocol and that the claimant has actually participated in the protocol [60]. Throughout this dissertation the terms authentication and identification are used interchangeably. Typically, the verifier is presented with, or presumes beforehand, the purported identity of the claimant, for example, that the claimant is Alice, and the goal is to corroborate that the identity of the claimant is indeed Alice. Either one or both parties may corroborate their identities to the other, providing *unilateral* or *mutual* identification, respectively.

The main difficulty in designing secure password mechanisms arises from the fact that the space of passwords from which most users tend to choose is small and much easier to attack by guessing than, for example, random cryptographic keys. Guessing attacks are most effective when a large number of guesses can be made automatically and each guess verified to see whether the guess was correct. A standard way to address this issue is to use salting as described in Chapter 1. However, this approach can be computationally intensive for the login system if the size of the salt is large.

2.1 Challenge-Response Schemes

More effective solutions to the problems associated with time-invariant passwords are those that involve the use of challenge-response mechanisms. The basic idea of challenge-response protocols used for authentication is that the password is used to compute the value of a function on a random challenge selected by the authentication server. The challenge changes with each session. Thus *freshness* of each authenticated session can be accomplished by providing a response to a time-variant challenge, such as a random number, where the response depends on both the claimant's secret and the challenge. For example, the verifier chooses a random element $n \in \mathbb{Z}^+$, encrypts it under the secret S shared between the claimant and the verifier, and then expects the claimant to respond with $f(n)$ encrypted under S, for some agreed upon function $f()$. Authentication is deemed successful if the claimant can correctly respond to the challenge.

Although challenge-response protocols provide freshness, they, unfortunately, leave the password open to password guessing attacks [33]. For example, Mallot can record an authentication session including the challenge from the authentication server and the corresponding response from Alice. Later, he tries a set of possible passwords on the challenge to see whether the same response is obtained. If so, then with high probability, Mallot has found Alice's password.

One variant of challenge-response mechanisms is *one-time* passwords [46, 34]. As the name suggests, once a one-time passwords is used it is no longer valid. One-time passwords are generated by applying a secure hash function, h, multiple times to a user's secret password. The secret, s, is never stored—instead, the first one-time password, $p_o = h^N(s)$, is stored on the authentication server. N represents the maximum number of one-time passwords allocated to the user. The next one-time password in the series (p_1) is generated by executing the hash function $N-1$ times. In general, p_i is calculated as $h^{N-i}(s)$. Verification of a user's one-time password is accomplished by requesting p_{i+1} (for the current value of i) from the user and checking that $p_i = h(h^{N-i-1}(s)) = h(p_{i+1})$ [34]. If this is the case, then the user is authenticated and the user's entry in the password file is updated accordingly. Otherwise, the authentication is unsuccessful.

Notice that even in challenge-response protocols such as one-time passwords, user-chosen passwords are used as keys to cryptographic functions. The security of these functions under such a small space of keys is clearly questionable. If the one-time passwords are derived from a user-chosen password, the latter is still vulnerable to password guessing attacks. The techniques we present in Chapter

5 strengthen the security of the derived cryptographic keys. Moreover, one-time password schemes are relevant primarily for network settings, to defend against the threat of a network eavesdropper capturing password information in transit between the user and a secure authentication server. Again, in the setting we consider in this dissertation, the concern is the capture and analysis of all stored information relevant to authentication. In this regard, the one-time password schemes of which we are aware offer no benefit against this type of attack over traditional passwords schemes.

2.2 Using Encryption for Authentication

Variants of challenge-response protocols are also used to provide mutual authentication. Typically they provide mutual authentication based on symmetric encryption (and involve the use of an online trusted party, for example [68, 66]), on public-key techniques [22, 66], or some combination of the two (see for example [9, 10, 31, 41, 53, 89]).

To our knowledge, the first work to propose the use of public-key techniques in conjunction with passwords for authentication was [30]. The authors suggested that by providing the authentication server with a private/public-key pair, one could protect user-chosen passwords against attacks by using public-key encryption. From this foundation, numerous techniques for frustrating guessing attacks have since been proposed. The basic principle to which these protocols adhere is that all data available to the attacker is made unpredictable enough to prevent off-line verification of whether a guess is successful or not; the only way that a guess can be verified involves interaction with some part of the system that is in a position to react to an excessive number of invalid login attempts.

Other influential work that proposed novel combinations of symmetric and asymmetric encryption is that of [9], in which they proposed the counter-intuitive use of a secret key (the user's password) to encrypt a randomly generated public-key, which is subsequently used to exchange secret information such as session keys. Their work resulted in numerous spinoffs [10, 41, 53, 89] protecting against different classes of attacks, the more complex of which have the desired properties of Zero-Knowledge (ZK) proofs [29]; in ZK proofs, the claimant, say Alice, demonstrates knowledge of a secret to the verifier, Bob, while revealing no information whatsoever (beyond what Bob was able to deduce prior to the protocol run) that is of use to him in conveying this knowledge to others.

This dissertation differs in its basic goals from the work outlined above: the techniques presented expand the password space that an off-line attacker must explore, but allow the attacker to confirm when the correct password has been found. These other works do the opposite: they do not expand the password space, but rather try to prevent the attacker from confirming a correct guess. Furthermore, our work is the first to offer stronger security against *both* online and off-line attackers, and is also the first to generate a repeatable secret based on a password and its keystroke dynamics that is stronger than the password itself and that can be used in applications other than login. Our approaches thus have the benefit of being applicable in more settings and in some cases can be combined with these other techniques to provide the benefits of both. Moreover, in some of these other works, for reasons of user acceptability, encryption keys are derived from a user-chosen password—there, our techniques may be combined in natural ways to improve overall system security.

Increasingly, attention is shifting to authentication techniques that encompass a third class of authentication—those that rely on characteristics of a person, for example, the way in which Alice signs her name. Security measures which rely on possession of a secret are inadequate because possession or knowledge may be compromised without discovery—the information or article may be extorted from its rightful owner. In the following section we review why biometrics have become so popular over the past few years, and examine why they make such ideal candidates for computer security.

2.3 Biometrics

Biometrics are the physical traits and behavioral characteristics that make each of us unique. They are a natural choice for identity verification because unlike keys or passwords, biometrics cannot be lost, stolen, or overheard, and in the absence of physical damage they offer a potentially foolproof way of determining a user's identity.

Indispensable to all biometric technologies is that they recognize a *living* person and encompass both anatomical and behavioral characteristics. Biometric technologies can be defined as automated methods of verifying or recognizing the identity of a living person based on an anatomical or behavioral characteristic [59]. Anatomical (i.e., *static*) characteristics, such as fingerprints and retinal prints, are good candidates for verification because they are stable (i.e., unalterable without causing trauma to the individual) and are unique across a large section of the population. Behavioral traits, on the other hand, have some anatomical basis, but also reflect a person's psychological makeup. Unique behavioral characteristics such as the pitch and amplitude in our voice, the way we sign our names, and even the way we type, form the basis of *non-static* biometric systems.

Biometrics technologies are gaining popularity because when used in conjunction with traditional methods for authentication they provide an extra level of security. However, for the foreseeable future, biometric solutions will not eliminate the need for identification cards, passwords and Personal Identification Numbers (PINs). Rather, the use of biometric technologies will provide a significantly higher level of identification and accountability than passwords and cards alone, especially in situations where security is paramount.

Biometric schemes generally rely on aspects of the body and its behavior. Slight changes in behavior are inevitable when dealing with non-static biometrics since they are influenced by both controllable actions and unintentional psychological factors. Therefore, biometric technologies need to be robust and adaptive to change—online signature verification systems, for example, update the reference template of a user on each successful authentication to the login device to account for slight variations in the signature. The scheme presented in Chapter 5 initially is as secure as a conventional password scheme and then adapts to the user's typing patterns over time, gradually strengthening the password with biometric information.

Some examples of biometrics being used for identification include hand geometry, thermal patterns in the face, blood vessel patterns in the retina and hand, finger and voice prints, and handwritten signatures (see, for example, [17, 20, 40, 91, 23, 44, 72]). Today, a few devices based on these biometric techniques are commercially available, some of which are non-intrusive. For example, the commercially available Face Access Control by Elemental Shapes (`FACES`) [72] is a passive monitoring system that uses the infrared emissions produced from an individual's face to uniquely identify her. The technology is inherently more accurate and more robust over varying lighting and environmental conditions than is the use of video images, but is more expensive.

Chapter 3

Preliminaries

3.1 Pattern Recognition

The biometric techniques presented in this dissertation, for the most part, rely on a fundamental concept—pattern recognition. Recognition is regarded as a basic attribute of human beings. Our ability to recognize people, even after being separated for some time, is remarkable. In this section we give a brief overview of the fundamental concepts related to pattern recognition.

The design of an automatic pattern recognition system involves *representation, extraction*, and *classification*. *Representation* of input data measures characteristics of the pattern or object to be recognized. When the measurements obtained yield information in the form of real numbers, it is often useful to think of a pattern vector as a point in an n-dimensional Euclidean space.

The *extraction* of characteristic features from the input data and the reduction of the dimensionality of the resulting pattern vectors is often referred to as the preprocessing and feature extraction problem. For example, we may choose to use only a selected number of measurements from the input, either because these features are enough to identify the individual (such as the latency between certain character pairs in Alice's password) or because the addition of extra features (applied pressure on the keyboard, for example) increases the computational complexity of the problem or yields no real benefit. The number of degrees of freedom of variation in the chosen index across the human population, their immutability over time and immunity to intervention, and the computational prospects for efficiently encoding and reliably recognizing the identifying pattern, must all be assessed during feature extraction.

Classification and *identification* involves the determination of optimum decision procedures. After the observed data from patterns to be recognized have been expressed in the form of measurement vectors in the pattern space, we want to decide to which pattern class these data belong [85, 24]. For example, given a number of user typing profiles, and an "unknown" reference template from a database of available users, we want to be able to identify the unknown individual with a specified level of certainty.

Biometric methods can never provide an absolutely certain identification because analysis of personal features has a natural range of variation. As such, biometric identification systems can fail in one of two ways; an authorized user may be rejected or an illegitimate user may be granted access to the system. Therefore, all biometric systems must allow for the control of error probabilities to some degree.

3.2 Secret Sharing and Polynomial Interpolation

Central to our technique for strengthening the security of textual passwords is the concept of secret sharing first described in [79, 11]. Secret sharing is the process of dividing some secret S into n pieces such that the secret is easily reconstructible from any k pieces, but even complete knowledge of $k-1$ pieces reveals absolutely no information about S. These schemes are commonly referred to as (k, n) *threshold* schemes, where the n pieces are called *shares.* The secret to be distributed is chosen by a special participant called the *dealer.*

Here we review the method of construction for a particular (k, n) *threshold* scheme, namely the **Shamir Threshold Scheme**, which was invented in 1979 by Adi Shamir [79]. In Shamir's scheme, the dealer constructs a random polynomial of degree at most $k-1$, $f(x) = a_0 + a_1 x +, \ldots, + a_{k-1} x^{k-1}$, where the *constant term is the secret* (thus $f(0) = a_0 = S$). Every participant involved in the sharing of the secret, obtains a point on this polynomial. By pooling their shares at a later time, any subset of k (or more) participants can reconstruct the secret by means of polynomial interpolation. The theorem of polynomial interpolation says, in effect, that a straight line can be passed through two points, a parabola through three, a cubic through four, and so on.

Moreover, given k points in the 2-dimensional plane $(x_1, y_1), \ldots, (x_k, y_k)$ with distinct x_i's, there is one and only one polynomial $f(x)$ of degree $k-1$ such that $f(x_i) = y_i$ for all i, since these points define k linearly independent equations in k unknowns [21]. Since the equations are linearly independent, there will be a unique solution, and the constant term, a_0, will be revealed as the key. Knowledge of $k-1$ of these shares is not sufficient to recover the secret S, since all values of S still remain equally probable. To see this, suppose you are told that $f(x)$ is a random polynomial of degree 2, and you are given only 2 points to reconstruct $f(x)$. There are infinitely many parabolas that pass through those two points, and without knowledge of a third point, the unique parabola defined by $f(x)$ will remain unknown.

The fundamental concept on which secret sharing relies is polynomial interpolation. One efficient method for interpolation is based on the Lagrange interpolation formula for polynomials [35]. The Lagrange interpolation formula is an explicit formula for the unique polynomial $f(x)$ of degree at most k. We review here the application of Lagrange interpolation for Shamir's scheme, as this is the scheme that we use in Chapter 5. The formula is as follows:

$$f(x) = \sum_{i=1}^{k} y_i \prod_{1 \le j \le k, j \ne i} \frac{x - x_j}{x_i - x_j}$$

Using the Lagrangian formula we have a polynomial of degree at most $k-1$ which contains the k ordered pairs $(x_i, y_i), 1 \le i \le k$. Any group of k participants can compute $f(x)$ by using the interpolation formula. However, since the participants do not need to know the entire polynomial (they only need to compute the constant term $S = f(0)$), the secret may be expressed as:

$$S = \sum_{i=1}^{k} c_i y_i,$$

where

$$c_j = \prod_{1 \le j \le k, j \ne i} \frac{x_j}{x_j - x_i}$$

is the standard Lagrange coefficient for interpolation. Hence, the key is a linear combination of the k shares. An efficient algorithm for Shamir's (k, n) *threshold* scheme can be found in [60].

Chapter 4

Keystroke Dynamics

It's not what you know, or what you have, but who you are.

This chapter focuses on our efforts to validate the thesis that keystroke features are repeatable and distinguishable among a diverse set of users. We present our results for an authentication system based on the use of keystroke dynamics. Keystroke dynamics is the process of analyzing the way users type at a terminal by monitoring the keyboard inputs thousands of times per second, and attempting to identify them based on habitual patterns in their typing rhythm. We present our data selection and extraction methods as well as our classification and identification strategies. Our observations and findings are discussed and compared with prior work in this area.

4.1 Prior Work

Keystroke identification techniques can be classified as either static or continuous. Static identification approaches analyze keystroke identification characteristics only at specific times, for example, during the login sequence. Static approaches provide more robust user identification than simple passwords, but do not provide continuous security—they can not detect a substitution of the user after the initial authentication. Continuous identification, on the contrary, monitors the user's typing behavior throughout the course of the interaction.

As early as 1980 researchers have been studying the use of habitual patterns in a users typing behavior for static identification schemes. To our knowledge, Gaines *et. al.* [26] were the first to investigate the possibility of using keystroke timings for authentication. Experiments were conducted with a very small population of seven secretaries. A test of statistical independence of their profiles was carried out using a standard technique, the *T-Test* [38], under the hypothesis that the means for character pairs (called digraphs) at different sessions were the same, but the variances were different. Similar experiments were conducted by Leggett *et. al.* [48, 49] with seventeen programmers but for the continuous approach to user identification. The authors report an identity verifier that validates the results of [26]—an identity verification system where legitimate users were rejected 5.5 percent of the time, and imposters were accepted approximately 5.0 percent of the time.

While the approaches of Gaines *et. al.* [26] and Leggett *et. al.* [48, 49] addressed a number of problems inherent to authentication via keystroke timings, there was considerable room for improvement. For example, [49] uses a 500 ms filter for all typists for the removal of data values for a variable that appear unusually large or small and out of place when compared with the other data values. (These erroneous data values are called outliers.) The rationale for this approach is that digraphs with abnormally long latencies are not likely to be representative of the authorized users' typing, and as such should be considered as outliers. While this seems like a reasonable assumption it was subsequently shown by [28] that the use of one filter value (500 ms) for all typists for the removal of outliers can be problematic. Gentner's studies suggests the median inter-key latency of expert typists is approximately 96 ms, while that of novice typists is near 825 ms, and as such, a single filter excludes many keystrokes typical of a novice typists, while at the same time, includes many keystrokes which are not representative of an expert typist. (This observation, however, plays an important role in the password strengthening technique presented in section 5.8.1.)

Some neural network approaches, [39, 4, 5, 6, 15, 2] have also been undertaken in the last few years. While the techniques used yield favorable performance results on small collection of users, neural network approaches have a fundamental limitation in that each time a new user is introduced into the database, the network must be retrained. For applications such as access control, the training requirements are prohibitively expensive and time consuming. Furthermore, in situations where there is a high turnover of users, the down time associated with retraining can be significant.

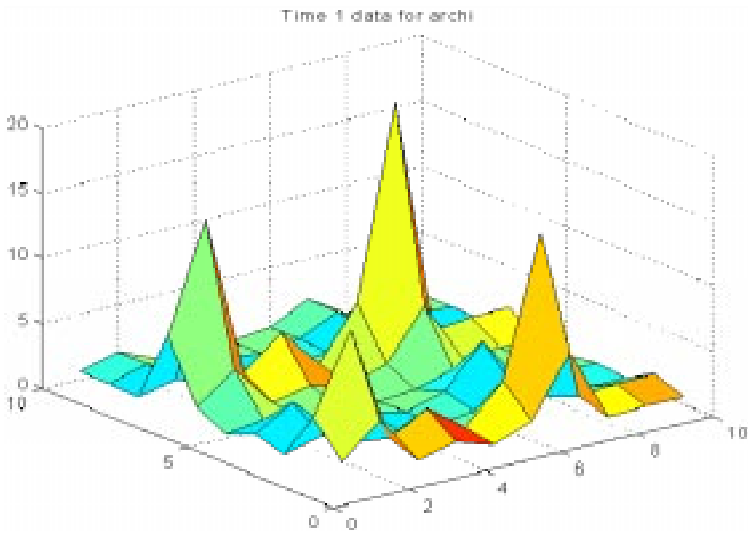
4.2 Our Experiments

A promising research effort in applying keystroke dynamics as a static authentication method is the work of Joyce and Gupta [43]. Their approach is relatively simple and yields impressive results. Our work extends that of [43]. We present our data selection and representation techniques, review the classifier of [43], and introduce our own classifiers in subsequent sections.

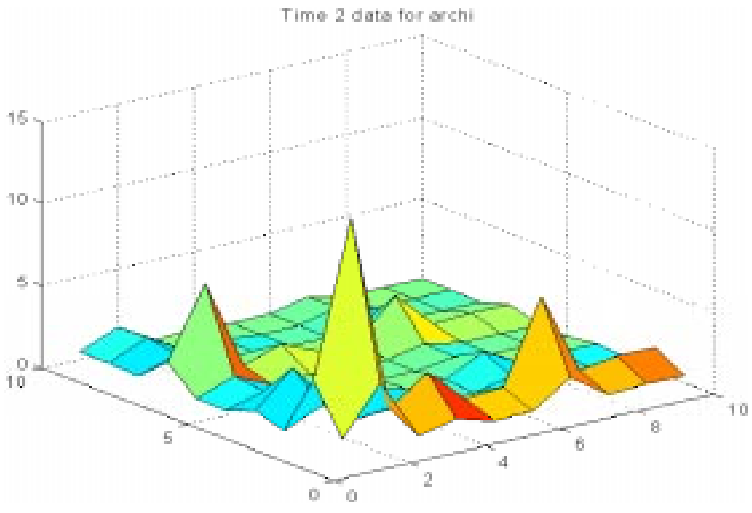
4.2.1 Data Selection and Representation

The performance results reported here are based on a database collected sporadically over a total period of approximately 7 months from 52 users on a variety of Sun workstations. Two types of measures were taken in these experiments: keystroke duration and latency between keystrokes. Typing proficiency was not a requirement in this study, although almost all participants were familiar with computers. At least two samples were collected from each user, with the time interval between sessions being anywhere from one to six weeks. During each session, users were asked to retype a few sentences from a list of available phrases and/or to type a few sentences on the spur of the moment. The text to be typed was placed in the top half of the display and the participants keystrokes were displayed in the bottom half. It was not necessary for participants to shift their focus between the text to be typed and their keystrokes.

Unlike previous studies in which the observers had complete control over the collection of the data [26, 49, 4, 5, 6], participants ran the experiment at their convenience. The results were automatically `uuencoded` and electronically mailed to a designated address. Figure 4.1 depicts a representation of the joint frequency distribution (for latencies and durations) for a set of digraphs typed by different users over different time intervals. Notice that the same peaks and overall structure can be observed between plots in Figure 4.1, but the plot of the joint distribution for the user whose typing pattern best matches that of (a), shown in Figure 4.2, is very distinct from plots (a) and (b).

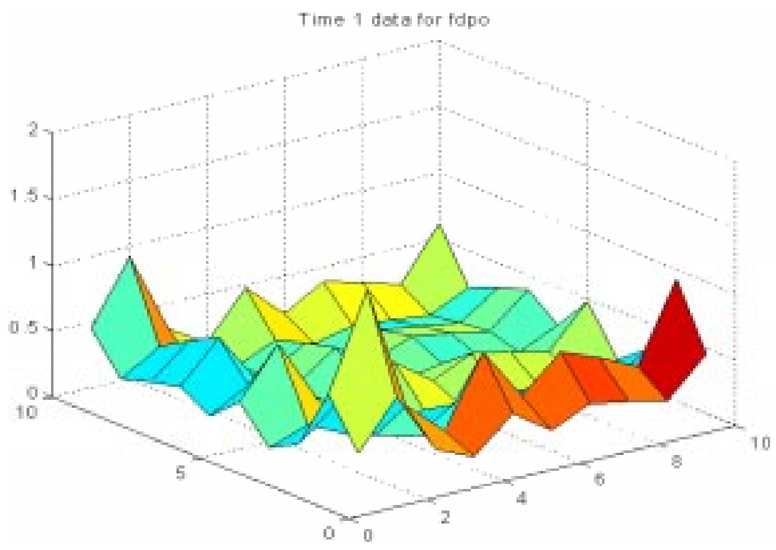


(a)



(b)

Figure 4.1: Joint frequency distribution for the same user at two differrent time intervals.



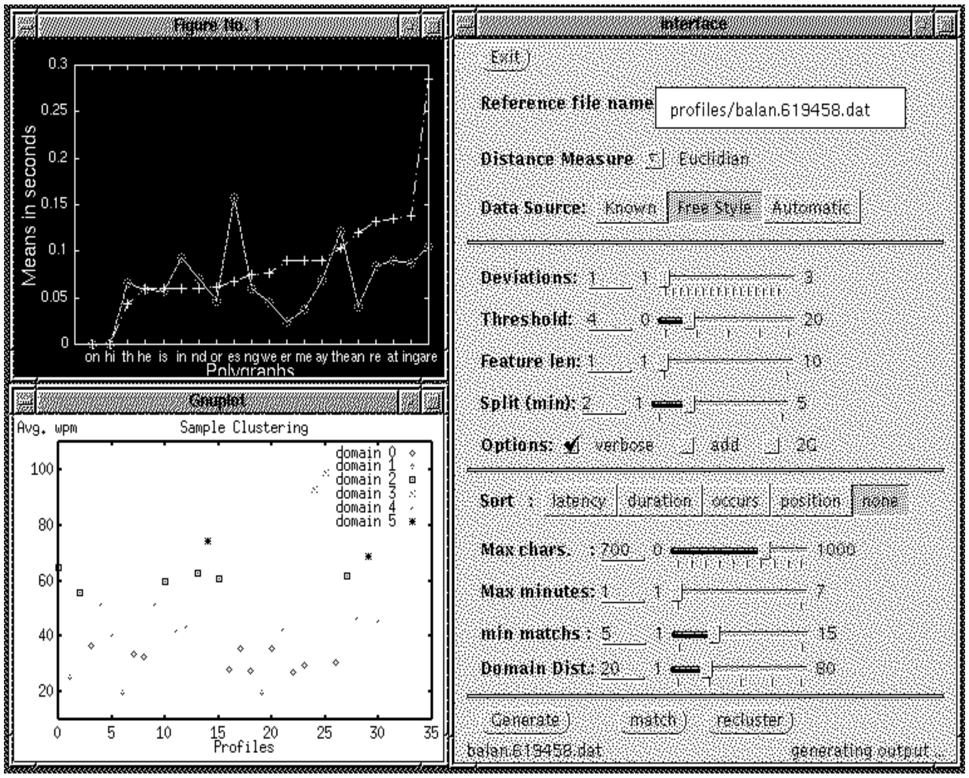
(c)

Figure 4.2: Joint frequency distribution for a different user over the same set of digraphs used in Figure 4.1.

4.2.2 Data Extraction

To evaluate the behavior and performance of each of the classifiers presented in Section 4.2.3 we developed a **C++** toolkit for analyzing the data. The toolkit was built using the `xview` library routines, and serves as a front-end to the main recognition engine. The toolkit (shown in Figure 4.3) aides in diagnosing system behavior and generates graphical output for both the `Matlab`$^{\text{TM}}$ and `Gnuplot`$^{\text{TM}}$ systems.

The data extraction toolkit provides a quick way to establish rough properties on the data set by partitioning the users into distinct groups. Our clustering criterion represents a heuristic approach that is guided by intuition—users are clustered into groups comprising of (possibly) disjoint feature sets in which the



Reference file name profiles/balan.619458.dat

Distance Measure Euclidian

Data Source: Known [✓ Free Style] Automatic

Deviations: 1 1 3

Threshold: 4 0 20

Feature len: 1 1 10

Split (min): 2 1 5

Options: ✓ verbose add 2G

Sort : latency duration occurs position [✓ none]

Max chars. : 700 0 1000

Max minutes: 1 1 7

min matchs : 5 1 15

Domain Dist.: 20 1 80

balan.619458.dat generating output ...

Figure 4.3: To automate the data selection and extraction process a system toolkit was designed to assist in the visualization, tuning, and overall analysis of the data. A graphical user interface with various tunable options allow the operator to diagnose the performance of each of the classifiers in detail. The above is a snapshot from the main panel of the interface.

features in each set are pairwise correlated.

Feature sets are determined through Factor Analysis (FA) [19]. Factor analysis seeks a lower dimensional representation that accounts for the correlation among features. This idea partitions the database of users into subsets whose in-class members are "similar" in typing rhythm over a particular set of features and whose cross-class members are dissimilar in the corresponding sense. For example, members of group i may exhibit strong individualistic typing patterns for features in the set $S = \{th, ate, st, ion\}$, whereas members of group j may be more distinctive over the features $S = \{ere, on, wy\}$. K-Nearest Neighbor [24] is used as the clustering algorithm. The net result is a hierarchical cluster that assists in user identification.

4.2.3 Classification and Identification

The problem of recognizing a given pattern as belonging to a particular person either after exhaustive search through a large database, or by simply comparing the pattern with a single authentication template can be formulated within the framework of statistical decision theory. By this approach one can convert the problem of pattern recognition into a much more expedient task, which involves the execution of tests of statistical independence. The approaches described in the following paragraphs adhere to this model.

The classification technique used by Joyce and Gupta [43] represents the mean reference signature for a given user as $\mathcal{M} = \{\mathcal{M}_{\text{username}}, \mathcal{M}_{\text{password}}, \mathcal{M}_{\text{firstname}}, \mathcal{M}_{\text{lastname}}\}$. Verification is performed by comparing the test signature $\mathcal{T}$ (acquired at login time) with $\mathcal{M}$ and determining the magnitude of difference between the two profiles. Given $\mathcal{M} = (m_1, m_2, ..., m_n)$ and $\mathcal{T} = (t_1, t_2, ..., t_n)$ where n is the total number of latencies in the signature, the verifier computes the magnitude of difference using an L_1 norm. Positive identification is declared when this difference is within a threshold variability of the reference signature. The mean and standard deviation of the norms $||M - S_i||$, where S_i is one of the eight training signatures, are used to decide the threshold for an acceptable difference vector between a given $\mathcal{T}$ and $\mathcal{M}$.

Although these absolute verification rates are encouraging, Joyce and Gupta tested using a replacement methodology, which means that the distribution of the training set is necessarily representative of the learning set. The use of separate data sets, recorded at different times, would be more reliable. Therefore, we investigated the performance of classifiers based on studies where users were allowed to participate in experiments conducted at varied times under no supervision. The reference profiles collected were represented as N-dimensional feature vectors and processed in a manner similar to that of [43]. The data was split into learning and testing sets. We used the following classifiers for identification:

- Euclidean Distance Measure: "similarity" is based on the Euclidean distance between the pattern vectors. Let $R = [r_1, r_2, ..., r_N]$ and $U = [u_1, u_2, ..., u_N]$ then the Euclidean distance between the two N dimensional vectors U and R, is defined as:

$$D(R, U) = \left[\sum_{i=1}^{N} (r_i - u_i)^2 \right]^{1/2}$$

  For an "unknown" U (i.e., from the testing set) the pairwise Euclidean distances $D(R_i, U), i = 1, 2, ..., n$ where n is the number of patterns vectors in the database, were rank ordered and the profile with the minimum distance to U was chosen.

- Non-Weighted Probability: Let U and R be N-dimensional pattern vectors as defined previously. Furthermore, let each component of the pattern vectors be the quadruple $\langle \mu_i, \sigma_i, o_i, X_i \rangle$, representing the mean, standard deviation, number of occurrences, and data value for the i^{th} feature. Assuming that each feature for a user is distributed according to a Normal distribution, we calculate the score between a reference profile R and unknown profile U as:

$$S(R,U) \; = \; \sum_{i=1}^{N} s_{u_i}$$

where

$$s_{u_i} \; = \; \frac{1}{o_{u_i}} \Big[\sum_{j=1}^{o_{u_i}} P\Big(\frac{X_{i_j}^{(u)} - \mu_{r_i}}{\sigma_{r_i}} \Big) \Big]$$

and $X_{i_j}^{(u)}$ is the j^{th} occurrence of the i^{th} feature of U.

In other words, the score for each u_i is based on the probability of observing the value u_{ij} in the reference profile R, given the mean (μ_{r_i}) and standard deviation (σ_{r_i}) for that feature in R. Intuitively we assign higher probabilities to values of u_i that are close to μ_{r_i} and lower probabilities to those further away. The "unknown" vector is then associated with the *nearest neighbor* in the database, i.e., to the person who maximizes the probability of the feature vector.

- Weighted Probability Measure: Some features are more reliable than others simply because they come from a larger sample set or have a relatively higher frequency in the written language; example in English "er", "th", and "re" should constitute greater weights than *qu* or *ts*. Thus, the notion of weights was incorporated, and the score between profiles R and U was computed as:

$$S(R,U) = \sum_{i=1}^{N} \left(s_{u_i} w_{u_i} \right)$$

where the weight of feature u_i is the ratio of its occurrences relative to all other features in the pattern vector U. Features that are based on many occurrences are considered more reliable and weighted higher than those features that come for a smaller sample set. Assuming that each feature for a user is distributed according to a Normal distribution, a likelihood score between a reference profile R and unknown profile U is calculated based on the probability of observing a feature value in the reference profile R, given the mean and standard deviation for that feature in R. Scores are weighted an the "unknown" profile is then associated with the *nearest neighbor* in the database, .i.e, the person who maximizes the score of the feature vector.

4.2.4 Results and Observations

The correct identification rate using the weighted probabilistic classifier was approximately 87.18% on a dataset of 52 users,[1] which represents improvement with respect to the performance of the Euclidean distance (83.22%) and the non-weighted scoring approach (85.63%). (In Chapter 5 we examine these results in further detail.)

While these results are encouraging, we believe that they reflect some of the problems with using keystroke dynamics for user recognition—different keyboards

[1]The results presented reflect a larger sample set than that reported in [61].

Table 4.1: Performance of classification templates.

Classifier	Identification (%)
Euclidean	83.22
Non-Weighted	85.63
Weighted	87.18

can significantly affect the typing pattern of the user. The poor recognition results reported here (in contrast to the performance of static biometrics such as fingerprints that have been shown to achieve correct classification and identification rates in the 99th percentile (see for example [40])), may be attributed to the nature of the data gathering process; the data was collected in an unrestricted setting under no supervision. Under such conditions we argue in favor of the use of structured text instead of allowing users to type arbitrary text (i.e., text-independent or "free-text"). While recognition based on free-text may be more desirable, free-text recognition simply did not perform as well as recognition based on fixed-text (interested readers are referred to [61]). Recognition based on free-text may be expected to vary greatly under operational conditions in which the user may be absorbed in a task or involved in an emotionally charged situation. The fact that the input is unconstrained, that the user may be uncooperative, and that environmental parameters are uncontrolled impose limitations on what can be achieved with free-text recognition.

4.3 Summary

In this chapter we address the thesis of using keystroke dynamics as a biometric for authenticating access to workstations. Keystroke dynamics is the process of analyzing the way users type by monitoring keyboard inputs and authenticating them based on habitual patterns in their typing rhythm. Although the use of a behavioral trait (rather than an anatomical characteristic) as a sign of identity has inherent limitations, for example, that different keyboards may affect the normal typing behavior of a user, we argue that when implemented in conjunction with other forms of identification (for example, identification based on knowledge of a password), keystroke dynamics may allow for the design of robust authentication systems.

Our empirical results suggest that the level of verification accuracy attained with static biometrics such as fingerprints (where false-alarm rates of less than 1% are not uncommon) is probably not achievable with a behavioral characteristic alone. In that regard, we believe that it is unlikely that successful user recognition systems based on keystroke dynamics can be built, unless their application is restricted to more confirmed environments than those under which our experiments were conducted. Though in the past years additional studies [76, 75] have been presented that support the hypothesis that different individuals exhibit characteristics in their typing rhythm that are strikingly individualistic, and that these characteristics can be successfully exploited and used for identification purposes, we believe that the strength of keystroke dynamics can be better realized when it is used in conjunction with, and not as opposed to, standard techniques for authentication.

In Chapter 5 we examine how individualistic characteristics in a user's typing pattern as she inputs her password can be used to build authentication schemes that are more secure than traditional password-based alternatives. There, we present a password strengthening technique which extends from the work presented here, but unlike [48, 49, 39, 4, 43, 15, 55, 2, 76, 75] we do not try to *recognize* users based on keystroke typing rhythm. Instead, once a claimed identity is presented, the user's typing features as she inputs her password are used to generate a repeatable key that is tested for access to a login device. Futhermore, the technique is restricted to logins using the same keyboard. Therefore, our approach involves a much more specific application of the use of keystroke dynamics, but one which can have significant impact in practice.

Chapter 5

Password Strengthening using Keystroke Dynamics

In a world of smart cards, hand-held authenticators, and zero-knowledge proofs, it seems pointless to be worrying about poorly-chosen passwords. Were the world like that, we might agree. Today, it is not—Bellovin and Merritt.

5.1 Introduction

In this Chapter we propose a technique for improving the security of password-based authentication. Here, keystroke dynamics is used not to *recognize* who is attempting to gain access to a device, but rather to *corroborate* the identity of the claimant, given knowledge of a password shared between the authentication server and the claimant. Specifically, we factor password typing patterns into the password authentication process, so that not only are the characters typed during the login sequence relevant to password authentication, but the manner in which they are typed is also relevant. Therefore, we rely on knowledge of a secret which the user types frequently. Intuitively, biometric information is used to *strengthen* the password so that an attacker who learns a legitimate user's password must still mimic the user's normal typing pattern to log in as that user. Moreover, we show that an attacker who obtains the stored system information used for password authentication (the analog of the `/etc/passwd` file in a typical UNIX environment) is faced with a more difficult task to exhaustively search for the password than in a traditional password-based scheme.

There are several challenges to realizing this goal. The first is to identify features of a user's typing patterns (e.g., the latency and duration of keystrokes while typing the password) that the user reliably repeats (approximately), and to translate these features into "secret bits" that can be used to strengthen the password. The second is to identify those features when the user logs in, and use them in the verification of the password. At the same time, however, the attacker who captures system information used to verify passwords should be unable to determine which particular features are relevant to each user's authentication process, since revealing this information could reveal information about what the characters related to that password feature are. For example, suppose the attacker learns that the latency between the first and second keystrokes is a feature that is reliably repeated by the user and thus is used in the authentication process. Then this may reveal information about the first and second characters of the password, since due to keyboard dynamics, some digraphs are more amenable to reliable latency repetitions than others.

The scheme we propose effectively hides information about which of a user's typing features are relevant to authentication, even from an attacker that captures all system information related to password authentication. At the same time, it employs novel techniques to impose an additional (multiplicative) computational overhead on the attacker who attempts to exhaustively search the password space. The overhead grows with the number of features that the user reliably repeats, which we show empirically will be large for the average user. Our empirical studies also demonstrate that our scheme provides sufficient ease of use to be viable in practice.

The main limitation of our scheme is that a user whose typing patterns change substantially between consecutive logins may be unable to login. The most common circumstance in which this could happen is if the user attempts to log in to her account using a different style keyboard than her regular one, which can cause a dramatic change in her typing patterns. Thus, our scheme is most suitable for use in, for example, logins to laptops, and not for use in remote logins where the user's keyboard varies depending on her location. A second limitation of our scheme is that it generally precludes sharing an account among multiple users, since one user's keystroke patterns will be different from another's.

5.2 Related Work

As described in Section 5.1, the use of keystroke features to strengthen user passwords is similar to salting, a technique that goes back at least twenty years [63].

Though an analogy to salting is an effective tool for understanding the goals of our scheme, the mechanics of our scheme differ fundamentally from salting. In our scheme, a strengthened password is generated from a textual password and the user's typing features, but not by concatenating the password with bits generated from the user's features. Rather, the password is used to decrypt a table of information, and the user's typing features are used to index into the table to select information from which the strengthened password is computed. This imposes an additional multiplicative computational overhead on an attacker conducting an off-line dictionary attack, because upon decrypting the table with a guessed password, the attacker cannot immediately determine if the guessed password is correct. Instead, the attacker must sample elements of the table and generate corresponding strengthened passwords until he either finds the correct one or determines that the guessed password was incorrect.

We reiterate that the technique presented here differs in its basic goals from the works of [9, 10, 31, 41, 53, 89] that protect against off-line guessing attacks by using public-key techniques. Our technique expand the password space that an off-line attacker must explore, but allow the attacker to confirm when the correct password has been found. These other works on the other hand, do the opposite: they do not expand the password space, but rather try to prevent the attacker from confirming a correct guess. Furthermore, our work is the first to offer stronger security against *both* online and off-line attackers, and is also the first to generate a repeatable secret based on a password and its keystroke dynamics that is stronger than the password itself and that can be used in applications other than login.

5.3 Preliminaries

For the rest of this Chapter, we denote by pwd_a the correct string of characters for the password field when logging into account a. That is, pwd_a denotes the correct text password as used in computer systems today. In our architecture, typing pwd_a is necessary but not sufficient to access a. Rather, the login program combines the characters typed in the password field with keystroke features to form an *strengthened password* that is tested to determine whether login is successful. The correct strengthened password for account a is denoted spwd_a. The login program will fail to generate spwd_a if either something other than pwd_a is entered in the password field or if the user's typing patterns significantly differ from the typing patterns displayed in previous successful logins to the account. Here we present our scheme in a way that maintains spwd_a constant across logins, even despite gradual shifts in the user's typing patterns, so that spwd_a can also be used for longer-term purposes (e.g., file encryption). However, the scheme can be easily tuned to change spwd_a after each successful login.

5.3.1 Features

In order to generate spwd_a from pwd_a and the (correct) user's typing patterns, the login program measures a set of features whenever a user types a password. Empirically we examine the use of keystroke duration and latency between keystrokes as features of interest, but other features (e.g., force of keystrokes) could be used if they can be measured by the login program. Abstractly, we define a feature by a function $\phi : A \times \mathbb{N} \to \mathbb{R}^+$ where $\phi(a, \ell)$ is the measurement of that feature during the ℓ-th (successful or unsuccessful) login attempt to account a. For example, if the feature ϕ denotes the latency between the first and second keystrokes, then $\phi(a, 6)$ is that latency on the sixth attempt to login to a. Let m denote the number of features that are measured during logins, and let $\phi_1, \ldots, \phi_m$ denote their respective functions.

For each feature ϕ_i, let $t_i \in \mathbb{R}^+$ be a value such that the mean of $\phi_i(a, \ell)$ over all $a \in A$ and all ℓ is roughly t_i. Intuitively, t_i is a threshold such that when a particular user shows regularity in a feature, the mean value of that user's measurements on that feature instance is equally likely to fall above or below t_i, in the absence of information about the user's password or typing patterns.

Central to our scheme is the notion of a *distinguishing feature*. For each feature ϕ_i, let $t_i \in \mathbb{R}^+$ be a fixed parameter of the system. Also, let μ_{a_i} and σ_{a_i} be the mean and standard deviation of the measurements $\phi_i(a, j_1), \ldots, \phi_i(a, j_h)$ where $j_1, \ldots, j_h$ are the last h successful logins to the account and $h \in \mathbb{N}$ is a fixed parameter of the system. We say that ϕ_i is a distinguishing feature for the account (after these last h successful logins) if $|\mu_{a_i} - t_i| > k\sigma_{a_i}$ where $k \in \mathbb{R}^+$ is a parameter of the system. If ϕ_i is a distinguishing feature for the account a, then either $t_i > \mu_{a_i} + k\sigma_{a_i}$, i.e., the user consistently measures below t_i on this feature, or $t_i < \mu_{a_i} - k\sigma_{a_i}$, i.e., the user consistently measures above t_i on this feature. The value t_i should be chosen so that users' distinguishing features fall above or below t_i with roughly equal likelihood. In Section 5.8, we give empirically generated values for t_i for features based on keystroke durations and keystroke latencies.

5.3.2 Security Goals

In our login architecture, the system stores information per account that is accessed by the login program to verify attempts to login. This information is necessarily based on pwd_a and spwd_a, but will not include either of these values themselves. This is similar to UNIX systems, for example, where the `/etc/passwd` file contains the salt for that password and the result of encrypting a fixed string using the password and salt. In our login architecture, the information stored per account will be somewhat more extensive, though will still be relatively small.

The first attacker with which we are concerned is an attacker who captures this information stored in the system and then uses this information in an off-line effort to find spwd_a (and pwd_a). A first and basic requirement is that any such attack be at least as difficult as exhaustively searching for pwd_a in a traditional UNIX setting where the attacker only has `/etc/passwd`. In particular, if the user chooses pwd_a to be difficult for an attacker to find using a dictionary attack, then spwd_a will be at least as secure in our scheme.

A more ambitious goal of our scheme is to increase the work that the attacker must undertake by a considerable amount, even if pwd_a is chosen poorly, i.e., in a way that is susceptible to a dictionary attack. The amount of additional work that the attacker must undertake generally grows with the number of distinguishing features for the account (when the attacker captured the system information). On one extreme, if there are no distinguishing features for the account and the attacker knows this,[1] then the attacker can find pwd_a and spwd_a in roughly the

[1]Our scheme hides the number of distinguishing features for an account. However, if the attacker captures system information at a point in time when he knows that a certain account has not yet been

same amount of time as the attacker would take to find pwd_a in a traditional UNIX setting. On the other extreme, if all m features are distinguishing for the account, then the attacker's task can be slowed by a multiplicative factor up to 2^m. In Section 5.8, we provide an empirical analysis of what this slowdown factor is likely to be in practice. In addition, we show how our scheme can be combined with other standard salting techniques, either with a stored salt [63] or one that is not stored [56]. Thus, the slowdown factor that our scheme achieves is over and above any benefits that conventional salting techniques offer.

A second attacker that we defend against with our scheme is an "online" attacker who learns pwd_a (e.g., by observing it being typed in) and then attempts to log in using it.

5.4 Overview

We begin with a high-level explanation of our technique. When an account a is initialized, the initialization program chooses the value of spwd_a at random from $\mathbb{Z}_q^*$, where q is a sufficiently large prime number (e.g., $|q| = 160$ bits). The initialization program then creates a table of $2m$ shares of the secret spwd_a using Shamir's secret sharing [79] (See section 3.2) as follows: it chooses a random polynomial $f_a \in \mathbb{Z}_q^*[x]$ of degree $m - 1$ such that $f_a(0) = \mathsf{spwd}_a$, and creates a two-column "instruction table"

used, then he would know that there are no distinguishing features for that account.

	$< t_i$	$\geq t_i$
1	$f_a(2)$	$f_a(3)$
2	$f_a(4)$	$f_a(5)$
$\vdots$	$\vdots$	$\vdots$
m	$f_a(2m)$	$f_a(2m+1)$

The initialization program encrypts each element of both columns with pwd_a. This (encrypted) table is stored in the system. In the ℓ-th login attempt to a, the login program uses the entered password text pwd' to decrypt the columns of the table (which will result in the previously stored values only if $\mathsf{pwd}_a = \mathsf{pwd}'$). For each feature ϕ_i, the value of $\phi_i(a, \ell)$ indicates which of the two values in the i-th row should be used in the interpolation to find $f_a(0)$ ($= \mathsf{spwd}_a$): if $\phi_i(a, \ell) < t_i$, then the value in the first column is used, and otherwise the value in the second column is used. In the first logins after initialization, the value in either the first or second column works equally well. However, as distinguishing features ϕ_i for this account develop over time, the login program perturbs the value in the second column of row i if $\mu_{a_i} < t_i$ and perturbs the value in the first column of row i otherwise. So, the proper interpolation to find $f_a(0)$ in the future will succeed only when future measurements of features are consistent with the user's previous distinguished features.

5.5 Stored Data Structures and Initialization

Let q be a prime number as described above, and let G be a pseudorandom function family [74] such that for any key K and any input x, $G_K(x)$ is a pseudorandom element of $\mathbb{Z}_q^*$.[2] In practice, a likely implementation of G would be $G_K(x) = F(K, x)$ where F is a one-way function, e.g., SHA-1 [78].

There are two data structures stored in the system per account.

- A plaintext *instruction table*, which contains "instructions" regarding how feature measurements are to be used to generate spwd_a. More specifically, this instruction table contains an entry of the form

  $$<i, \alpha_{a_i}, \beta_{a_i}>$$

  for each feature ϕ_i. Here,

  $$\begin{aligned}\alpha_{a_i} &= y^0_{a_i} G_{\mathsf{pwd}_a}(2i) \bmod q \\ \beta_{a_i} &= y^1_{a_i} G_{\mathsf{pwd}_a}(2i+1) \bmod q\end{aligned}$$

  and $y^0_{a_i}, y^1_{a_i}$ are elements of $\mathbb{Z}_q^*$. Initially (i.e., when the user first chooses pwd_a), all $2m$ values $\{y^0_{a_i}, y^1_{a_i}\}_{1 \le i \le m}$ are chosen such that all the points $\{(2i, y^0_{a_i}), (2i+1, y^1_{a_i})\}_{1 \le i \le m}$ lie on a single, random polynomial $f_a \in \mathbb{Z}_q^*[x]$ of degree $m-1$ such that $f_a(0) = \mathsf{spwd}_a$.

- An encrypted, constant-size *history file*, which contains the measurements for all features over the last h successful logins to a for some fixed parameter h. More specifically, if since the last time pwd_a was changed, the login attempts $j_1, \ldots, j_\ell$ to a were successful, then this file contains $\phi_i(a, j)$ for each $1 \le i \le m$ and $j \in \{j_{\ell-h+1}, \ldots, j_\ell\}$. In addition, enough redundancy is added to this

[2]That is, a polynomially-bounded adversary not knowing K cannot distinguish between $G_K(x)$ and a randomly chosen element of $\mathbb{Z}_q^*$, even if he is first allowed to examine $G_K(\hat{x})$ for many $\hat{x}$'s of his choice and is allowed to even pick x (as long as it is different from every $\hat{x}$ he previously asked about).

file so that when it is decrypted with the key under which it was previously encrypted, the fact that the file decrypted successfully can be recognized.

When a user first chooses her password pwd_a or changes pwd_a, this file is initialized with all values set to 0, and then is encrypted with $\mathsf{spwd}_a = f_a(0)$ (see above) using a symmetric cipher. The size of this file should remain constant over time (e.g., must be padded out when necessary), so that its size yields no information about l.

5.6 The Login Protocol

The login program takes the following steps whenever the user attempts to login to a. Suppose that this is the ℓ-th attempt to login to a, and let pwd' denote the sequence of characters that the user typed. The login program takes the following steps.

1. For each ϕ_i, the login program uses pwd' to "decrypt" α_{a_i} if $\phi_i(a,\ell) < t_i$, and uses pwd' to "decrypt" β_{a_i} otherwise. Specifically, it assigns
$$(x_i, y_i) = \begin{cases} (2i,\ \alpha_{a_i} G_{\mathsf{pwd}'}(2i)^{-1} \bmod q) & \text{if } \phi_i(a,\ell) < t_i \\ (2i+1,\ \beta_{a_i} G_{\mathsf{pwd}'}(2i+1)^{-1} \bmod q) & \text{if } \phi_i(a,\ell) \geq t_i \end{cases}$$
The login program now holds m pairs $\{(x_i, y_i)\}_{1 \leq i \leq m}$.

2. The login program sets
$$\mathsf{spwd}' = \sum_{i=1}^{m} y_i \lambda_i \bmod q$$
where
$$\lambda_i = \prod_{1 \leq j \leq m, j \neq i} \frac{x_j}{x_j - x_i}$$

is the standard Lagrange coefficient for interpolation (e.g., see [60, p. 526]). It then decrypts the history file using spwd'. If this decryption yields a properly-formed plaintext history file, then the login is deemed successful. (If the login were deemed unsuccessful, then the login procedure would halt here.)

3. The login program updates the data in the history file, computes the standard deviation σ_{a_i} and mean μ_{a_i} for each feature ϕ_i over the last h successful logins to a, encrypts the new history file with spwd' (i.e., spwd_a), and overwrites the old history file with this new encrypted history file.[3]

4. The login program generates a new random polynomial $f_a \in \mathbb{Z}_q^*[x]$ of degree $m-1$ such that $f_a(0) = \mathsf{spwd}'$ (i.e., spwd_a).

5. For each distinguishing feature ϕ_i, i.e., $|\mu_{a_i} - t_i| > k\sigma_{a_i}$, the login program chooses new random values $y^0_{a_i}, y^1_{a_i} \in \mathbb{Z}_q^*$ subject to the following constraints:

$$\begin{aligned} \mu_{a_i} < t_i &\;\Rightarrow\; f_a(2i) = y^0_{a_i} \wedge f_a(2i+1) \neq y^1_{a_i} \\ \mu_{a_i} \geq t_i &\;\Rightarrow\; f_a(2i) \neq y^0_{a_i} \wedge f_a(2i+1) = y^1_{a_i} \end{aligned}$$

   For all other features ϕ_i—i.e., those for which $|\mu_{a_i} - t_i| \leq k\sigma_{a_i}$, or all features if there have been fewer than h successful logins to this account since initialization (see Section 5.3.1)—the login program sets $y^0_{a_i} = f_a(2i)$ and $y^1_{a_i} = f_a(2i+1)$.

6. The login program replaces the instruction table with a new table with an

[3]For maximum security, steps 2 and 3 should be performed without writing the plaintext history file to disk. Rather, the login program should hold the plaintext history in volatile storage only.

entry of the form

$$<i, \alpha'_{a_i}, \beta'_{a_i}>$$

for each feature ϕ_i. Here,

$$\begin{aligned}\alpha'_{a_i} &= y^0_{a_i} G_{\mathsf{pwd}'}(2i) \bmod q \\ \beta'_{a_i} &= y^1_{a_i} G_{\mathsf{pwd}'}(2i+1) \bmod q\end{aligned}$$

where $y^0_{a_i}, y^1_{a_i}$ are the new points generated in the previous step.

Step 4 above is particularly noteworthy for two reasons. First, due to this step, the polynomial f_a is changed to a new random polynomial during each successful login. This ensures that an attacker viewing the instruction table at two different times will gain no information about which features switched from distinguishing to non-distinguishing and vice-versa during the interim logins. That is, each time the attacker views an instruction table for an account, either all values will be the same since the last time (if there were no successful logins since the attacker last saw the table) or all values will be different. Another important aspect of Step 4 is that f_a, though generated randomly, is chosen so that $f_a(0) = \mathsf{spwd}_a$. This is what ensures that spwd_a remains constant across multiple logins.

Step 5 is also noteworthy, since it shows that whether each feature is distinguishing is recomputed in each successful login. Hence, a feature that was previously distinguishing can become undistinguishing and vice-versa. This is the mechanism that enables our scheme to naturally adapt to gradual changes in the user's typing patterns over time.

We reiterate that our scheme can be easily combined with standard salting techniques to further improve security. A natural place to include an additional salt is just before using spwd_a to decrypt the history file. For example, when spwd_a is generated during a login, the key used to decrypt the history file is $G_{\mathsf{spwd}_a}(s)$ where s is a salt. The salt can be stored as is typically done today, or may not be stored so that the system must exhaustively search for it [56]. In this case, the extra salt results in an additional multiplicative factor that the attacker must overcome.

5.7 Security

In this section we discuss the security of our scheme. First, it is immediate that our scheme helps defend against an on-line attacker who learns (or tries to guess) pwd_a and then attempts to log into a using pwd_a. Unless this attacker can mimic the measurements for the account's distinguishing features, the attacker will fail in logging into the account.

We now turn to the off-line attacker who obtains account a's history file and instruction table, and attempts to find the value of spwd_a. Presuming that the encryption of the history file using spwd_a is secure, since the values $y^0_{a_i}, y^1_{a_i}$ are effectively encrypted under pwd_a, and since pwd_a is presumably chosen from a much smaller space than spwd_a, the easiest way to find spwd_a is to first find pwd_a. Thus, to argue the benefits of this scheme, we have to show two things. First, we have to show that finding pwd_a is not made easier in our scheme than it is in a typical environment where access is determined by testing the hash of the password against a previously stored hash value. Second, we have to show that our scheme offers security benefits, i.e., that the cost to the attacker of finding

spwd_a is generally greater by a significant multiplicative factor.

That searching for pwd_a is not made easier in our scheme is obvious. The attacker has available only the instruction table with one entry per feature, and the encrypted history file. Since the features listed in the instruction table are all possible features (and not just those that are distinguishing for account a), and since the contents of each row are pseudorandom values, the rows reveal no information about pwd_a. And, all other data available to the attacker is encrypted with spwd_a.

The more interesting security consideration in this scheme is how much security it achieves over a traditional password scheme, i.e., how much harder it is to find spwd_a than it is to find pwd_a. Suppose that the attacker captured the history file and instruction table after $\ell \geq h$ successful logins to a, and let d be the number of distinguishing features for this account in the ℓ-th login. When guessing a password pwd', the attacker can decrypt each field α_{a_i} and β_{a_i} using pwd' to yield points $(2i, \hat{y}^0_{a_i})$ and $(2i{+}1, \hat{y}^1_{a_i})$, respectively, for $1 \leq i \leq m$. Note that $\hat{y}^0_{a_i} = y^0_{a_i}$ and $\hat{y}^1_{a_i} = y^1_{a_i}$ if and only if $\mathsf{pwd}' = \mathsf{pwd}_a$. Therefore, there exists a bit vector $b \in \{0,1\}^m$ of length m such that $\{(2i + b(i), \hat{y}^{b(i)}_{a_i}\}_{1 \leq i \leq m}$ interpolates to a polynomial $\hat{f}$ with $\hat{f}(0) = \mathsf{spwd}_a$, if and (a.s.) only if $\mathsf{pwd}' = \mathsf{pwd}_a$. (Here, $b(i)$ denotes the i-th bit of b.) Consequently, one "naive" approach that the attacker can take is to enumerate through all vectors $b \in \{0,1\}^m$ and, for each $\hat{f}$ thus computed, see if $\hat{f}(0) = \mathsf{spwd}_a$ (i.e., if $\hat{f}(0)$ will decrypt the history file). This approach slows down the attacker's search for spwd_a (and pwd_a) by a multiplicative factor of 2^m.

However, the attacker has potentially more powerful attacks at his disposal using error-correcting techniques [32]. In [62] we extend the scheme presented here to confound the application of such error-correcting techniques. Since the techniques available to the attacker are outperformed by the "naive" approach for the range of m we are concerned with here, i.e., $m \leq 15$ for an 8-character password, we do not present that variation.

5.8 Empirical Analysis

In this section we examine our proposal in light of empirical typing data presented in Chapter 4. Obviously this data will enable only a rough and, we believe, pessimistic analysis of our proposal, because a user tends to type her password more frequently than every one to six weeks. Indeed, we expect that once or more per day is typical. Therefore, a password should be much more "practiced" than the sentences used in this study and thus will yield much better results than the data here does. Nevertheless, this data already suggests that our scheme should be viable in practice, and will often result in a significant improvement in security.

Due to space limitations, here we provide an analysis for our scheme only for the case $k = 1$, where a feature ϕ_i is distinguishing if $|\mu_{a_i} - t_i| > k\sigma_{a_i}$ (see Section 5.3.1). In general, a lower value of k increases the number of distinguishing features per user and thus increases the sensitivity of login to user typing patterns. On the other hand, a higher value of k makes it easier for the user to log in, but tends to decrease the number of distinguishing features per user. As we show here, the value $k = 1$ seems to offer a good balance between security and ease of use.

5.8.1 Choosing t_i

The strength of our proposal is enhanced if the number d of distinguishing features for a user is large relative to the number m of features overall. In this section we choose values of t_i for features based on duration and latency, and based on our data set, we show that in fact the ratio of d to m is likely to be high on average.

Let C denote the set of characters typed in these experiments, and let U be the set of users in these experiments. For each $u \in U$ and $c \in C$,[4] let $\mu_{\mathsf{dur}}(u, c)$ be the mean duration of the keystrokes in which u typed c, and let $\sigma_{\mathsf{dur}}(u, c)$ be the standard deviation of these measurements. Similarly, for each $u \in U$, $c_1, c_2 \in C$,[5] let $\mu_{\mathsf{lat}}(u, c_1, c_2)$ be the mean latency between c_1 and c_2 whenever u typed c_1 followed by c_2, and let $\sigma_{\mathsf{lat}}(u, c_1, c_2)$ be the standard deviation of these measurements. Given values t_{dur} and t_{lat}, we define the following values:

- $\mathsf{low}_{\mathsf{dur}}$ is the percentage of all (u, c) pairs such that $t_{\mathsf{dur}} > \mu_{\mathsf{dur}}(u, c) + \sigma_{\mathsf{dur}}(u, c)$
- $\mathsf{high}_{\mathsf{dur}}$ is the percentage of all (u, c) pairs such that $t_{\mathsf{dur}} < \mu_{\mathsf{dur}}(u, c) - \sigma_{\mathsf{dur}}(u, c)$
- $\mathsf{low}_{\mathsf{lat}}$ is the percentage of all (u, c_1, c_2) triples such that $t_{\mathsf{lat}} > \mu_{\mathsf{lat}}(u, c_1, c_2) + \sigma_{\mathsf{lat}}(u, c_1, c_2)$
- $\mathsf{high}_{\mathsf{lat}}$ is the percentage of all (u, c_1, c_2) triples such that $t_{\mathsf{lat}} < \mu_{\mathsf{lat}}(u, c_1, c_2) - \sigma_{\mathsf{lat}}(u, c_1, c_2)$

Based on our data, reasonable values of t_{dur} and t_{lat} to support our scheme are shown in Table 5.1. These values imply that $\mathsf{low}_{\mathsf{dur}} \approx \mathsf{high}_{\mathsf{dur}}$ and $\mathsf{low}_{\mathsf{lat}} \approx \mathsf{high}_{\mathsf{lat}}$.

[4]Only pairs (u, c) for which we had at least 8 samples were counted.

[5]Only triples (u, c_1, c_2) for which we had at least 6 samples were counted.

Thus, if in the scheme of Section 5.4, we set $t_i = t_{\mathsf{dur}}$ for each feature ϕ_i that is some keystroke's duration (e.g., ϕ_i is the duration of the second keystroke), then the attacker has little *a priori* basis to predict whether the user's mean is above or below t_i, in the case that ϕ_i is a distinguishing feature for that user. The attacker is in a similar situation with respect to the latency-based features ϕ_i if we set $t_i = t_{\mathsf{lat}}$. The uncertainty faced by the attacker due to these choices of t_{dur} and t_{lat} is further analyzed in Section 5.8.2.

Table 5.1: Choices for t_{dur} and t_{lat}

$t_{\mathsf{dur}} = 0.088709\text{s}$	$t_{\mathsf{lat}} = 0.121367\text{s}$
$\mathsf{low}_{\mathsf{dur}} = 37.1\%$	$\mathsf{low}_{\mathsf{lat}} = 32.9\%$
$\mathsf{high}_{\mathsf{dur}} = 30.9\%$	$\mathsf{high}_{\mathsf{lat}} = 31.6\%$

The fact that $\mathsf{low}_{\mathsf{dur}} + \mathsf{high}_{\mathsf{dur}} < 100\%$ means that for various u and c, it was the case that $|\mu_{\mathsf{dur}}(u, c) - t_{\mathsf{dur}}| \leq \sigma_{\mathsf{dur}}(u, c)$. This implies that when such a user adopts such a character for a particular position in her password, the duration of that character position is not likely to be a distinguishing feature for the user. However, note that for the majority of (u, c) pairs—68% of them—choosing the character c for a position in u's password yields a distinguishing feature for the user. A similar situation holds for latencies between characters, where in this case the majority is 64.5%.

5.8.2 Entropy of Distinguishing Features

Section 5.8.1 demonstrated choices of t_{dur} and t_{lat} that yielded a large number of distinguishing features for the average account, based on the user population in our experiments. This alone, however, is not enough to conclude that our scheme adds a significant amount of security. To see why, suppose for an extreme case that all users could be partitioned into "slow typists" and "fast typists": slow typists have the property that for any of their distinguishing features ϕ_i, $\mu_{a_i} > t_i$ (where a is the user's account), and analogously fast typists have the property that $\mu_{a_i} < t_i$ for all of their distinguishing features ϕ_i. Then, even if *all* of an account's features are distinguishing, the off-line attacker only needs to examine two possibilities upon guessing a password pwd': either the values in the first column of the (decrypted) instruction table, or the values in the second column. If neither of these columns can be used to generate spwd_a, then the attacker can conclude with high probability that $\mathsf{pwd}' \neq \mathsf{pwd}_a$, after examining only 2 of the 2^m possibilities.

Ideally, users would type "randomly", in the sense that for any of an account's distinguishing features, whether that feature is "high" or "low" is uncorrelated to other distinguishing features for that account. Of course, one cannot expect this to be the case in reality. Therefore we measure this degree of correlation. We need the following definitions. For a user u and character c, we say that c is a *distinguishing character* for u if $|\mu_{\mathsf{dur}}(u,c) - t_{\mathsf{dur}}| > \sigma_{\mathsf{dur}}(u,c)$. Similarly, for a user u and a character pair (c_1, c_2), we say that (c_1, c_2) is a *distinguishing character pair* for u if $|\mu_{\mathsf{lat}}(u, c_1, c_2) - t_{\mathsf{lat}}| > \sigma_{\mathsf{lat}}(u, c_1, c_2)$. A *distinguishing event* e for user u is either a distinguishing character or distinguishing character pair for u. For a distinguishing event e, we define $\mu(u,e) = \mu_{\mathsf{dur}}(u,c)$ and $\sigma(u,e) = \sigma_{\mathsf{dur}}(u,c)$ if e denotes the distinguishing character c for u, and we define $\mu(u,e) = \mu_{\mathsf{lat}}(u,c_1,c_2)$ and $\sigma(u,e) = \sigma_{\mathsf{lat}}(u,c_1,c_2)$ if e denotes the distinguishing character pair (c_1,c_2) for u. Finally, $t_e = t_{\mathsf{dur}}$ if e is a distinguishing character, and $t_e = t_{\mathsf{lat}}$ if e is a distinguishing character pair.

Now consider the following experiment. For a fixed number of distinguishing features d, $0 \leq d \leq 15$, consider the bit string $b \in \{0,1\}^d$ generated by (i) choosing a user u uniformly at random from U, (ii) choosing a sequence of distinguishing events $e_1, \ldots, e_d$ uniformly at random (allowing repetition) from all distinguishing events for u, such that at most 7 events are distinguishing character pairs and at most 8 are distinguishing characters, and (iii) for each e_i, if $\mu(u,e) < t_e$ then set $b(i) = 0$ and otherwise set $b(i) = 1$. Then, computing the entropy on the distribution of b characterizes the degree of correlation among a user's distinguishing features for 8-character passwords. On one extreme, if all users could be partitioned into "fast" or "slow" typists, then the entropy of this distribution would be one, regardless of d. On the other extreme, if each user types "randomly" in the sense described above, then the entropy of this distribution is d.

The entropy of this distribution based on our data, as well as the entropy of the distributions when restricted only to 8 distinguishing characters (duration) or 7 character pairs (latency), are shown in Figure 5.1. This figure demonstrates that though users are far from the ideal case of "random", additional distinguishing features do contribute entropy, at least in the range of d that we are likely to encounter for 8-character passwords. Consequently, the job of the off-line attacker does become more difficult as the number of distinguishing features increases.

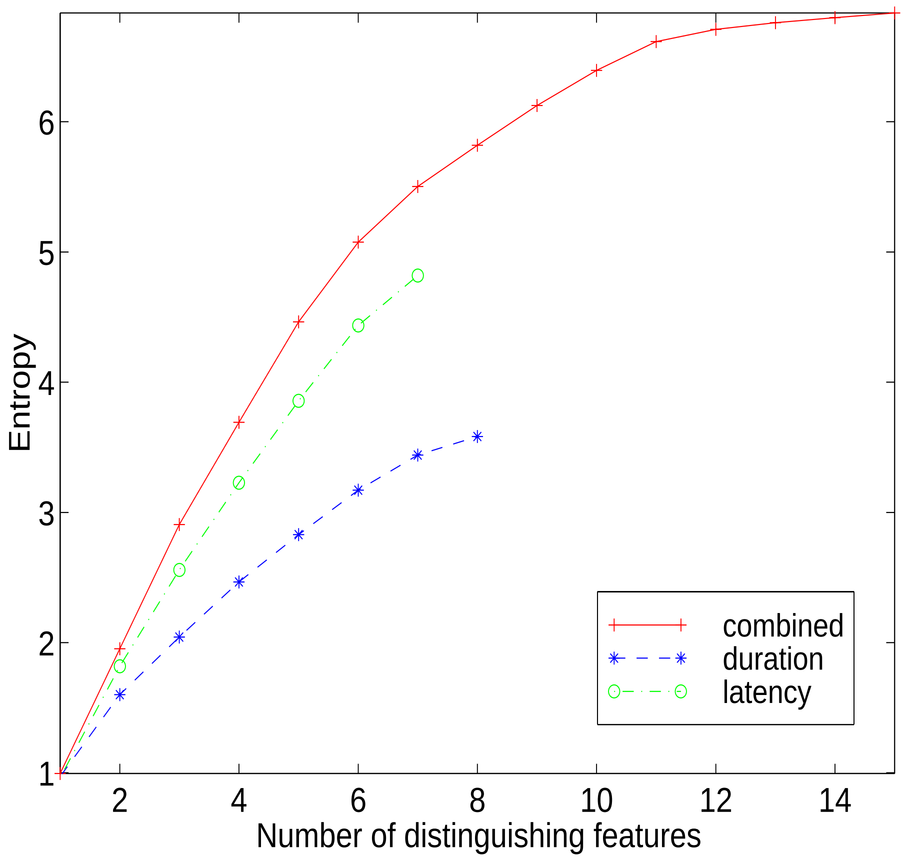


Figure 5.1: Approximate entropy of d distinguishing features

If the results of this calculation turn out to be representative in general, as does our derived expected value of $d \approx 0.66m$ from Section 5.8.1, then this suggests that the average user will have roughly 10 distinguishing features for an 8-character password and that these 10 distinguishing features will impose an additional multiplicative work factor of roughly $2^{6.39} \approx 84$ on the off-line attacker's search for spwd_a (or pwd_a). Moreover, Figure 5.1 does not capture the full uncertainty that the attacker faces, since the attacker does not know the features that are distinguishing for any given account or even the number of distinguishing features for that account. The data presently available to us does not permit a more precise calculation of this full uncertainty, though clearly the entropy reflected in Figure 5.1 is but one component of the uncertainty that the attacker faces.

We caution the reader that the experiment measured in Figure 5.1 is only a coarse characterization of the entropy due to distinguishing features. This is partly due to the nature of the data available to us, as already discussed. Moreover, our experiment ignores certain constraints that might affect the entropy for high values of d. For example, a valid experiment for the case $d = 15$ would require that any two distinguishing characters c_1 and c_2 at adjacent positions in the password be separated by the distinguishing character pair (c_1, c_2), i.e., consisting of the *same* two characters. Due to insufficient data to conduct an experiment under the full set of such constraints, we have waived certain constraints here. Thus, Figure 5.1 should not be viewed as a precise characterization of entropy due to distinguishing features, but rather primarily as evidence that entropy due to distinguishing features does increase as the number of distinguishing features grows.

5.8.3 Reliability of Password Entry

As discussed in Section 5.7, our scheme offers additional security against an online attacker who learns pwd_a (e.g., by observing it being typed) and attempts to log into a. Unless the attacker can type like the legitimate user, then the attacker will fail. However, this implies that if the legitimate user types her password abnormally, her login will also fail. Thus, it is important to empirically analyze not only the security of our scheme, but also the ability of the legitimate user to reliably log into her own account.

To measure this reliability, we split our records of user typing patterns into two groups, namely "training" data and "test" data. For each user u, we computed the distinguishing characters and character pairs for that user *based on the training data only.* Now consider the following experiment: For a fixed number of distinguishing features d, $1 \leq d \leq 15$, consider the bit strings $b_{\mathsf{train}}, b_{\mathsf{test}} \in \{0,1\}^d$ generated as follows:

1. Choose a user u uniformly at random from U.
2. Choose a sequence of distinguishing events $e_1, \ldots, e_d$ for u (allowing repetition) uniformly at random, where at most 7 are distinguishing character pairs and at most 8 are distinguishing characters.
3. For each e_i, if $\mu(u, e_i) < t_e$ then set $b_{\mathsf{train}}(i) = 0$ and otherwise set $b_{\mathsf{train}}(i) = 1$.
4. For each e_i: If e_i is a distinguishing character c, then choose an occurrence of u typing c from the test data uniformly at random and set $b_{\mathsf{test}}(i) = 0$ if the duration of that occurrence was less than t_{dur} and $b_{\mathsf{test}}(i) = 1$ otherwise. If e_i is a distinguishing character pair (c_1, c_2), then choose an occurrence of u typing (c_1, c_2) from the test data set uniformly at random and set $b_{\mathsf{test}}(i) = 0$ if the latency of that occurrence was less than t_{lat} and $b_{\mathsf{test}}(i) = 1$ otherwise.

Intuitively, when $b_{\mathsf{train}} = b_{\mathsf{test}}$ in the above experiment, this corresponds to a successful login by the user, where the login attempt is generated from the test data. The fraction of cases in which $b_{\mathsf{train}} = b_{\mathsf{test}}$ is shown in Figure 5.2.

As expected, Figure 5.2 shows that the reliability of password entry drops as the number of distinguishing features grows. However, even if all 15 features of an

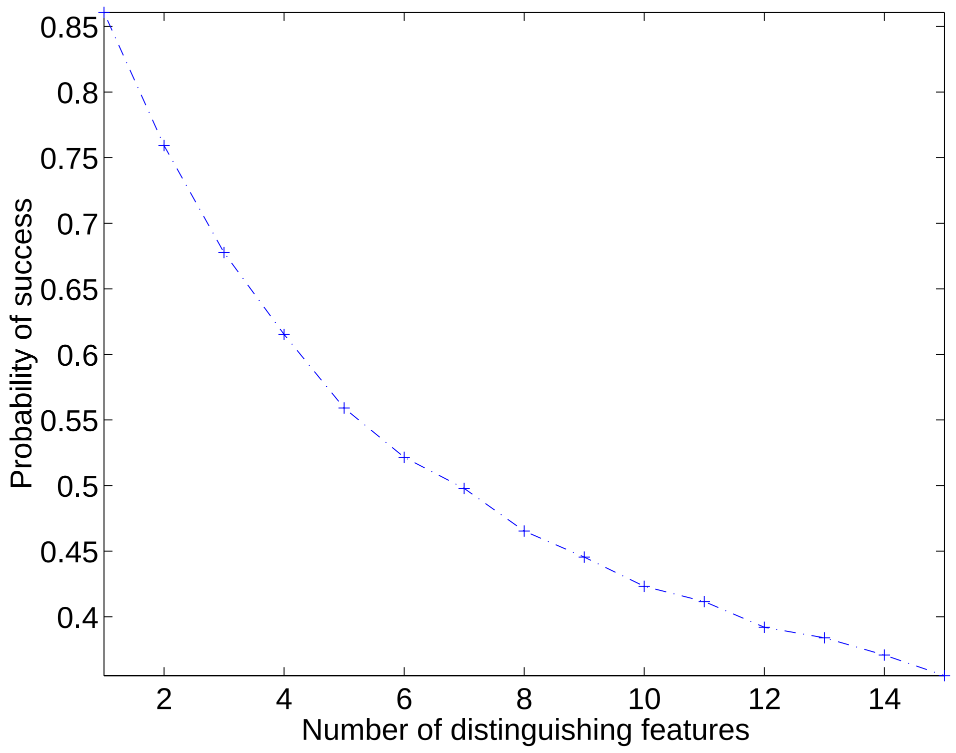


Figure 5.2: Approximate probability of successful password entry

8-character password are distinguishing for the user, our data still suggests that each of the user's login attempts succeeds with probability greater than 1/3 (among those attempts in which the user types the correct password pwd_a for the account a). That is, the expected number of login attempts before the user successfully logs in is less than 3. We thus believe that our scheme offers good reliability for user login. We reiterate that we believe our data to present a pessimistic view of our scheme, and consequently in practice, we expect that our scheme will offer even better login reliability. Moreover, login reliability can be improved by increasing the value of k, at the cost of decreasing the number of distinguishing features for the average user.

5.9 Summary

Given that our scheme is at least as secure as typical password schemes in use today, perhaps the most cogent reason to not use our scheme is the risk that a user, due to some a sharp change in her typing patterns, finds herself unable to log in after repeated attempts. As discussed in Section 5.1, to minimize this risk we further recommend that our scheme be used primarily to support local (console) logins on the same keyboard, e.g., for file encryption or virtual private network access on laptops, which are frequently stolen.

In addition, an appropriate recovery channel can be established for the event that a user finds herself unable to gain access to her account a after repeated attempts even though she remembers pwd_a. A recovery program can easily be derived from the login program described in Section 5.4: the recovery program decrypts all instruction table entries using the password pwd_a (provided by the user) and then exhaustively searches to find spwd_a (within time proportional to 2^m). However, this recovery program should not simply be used as an alternative login program, since it would enable an attacker who learns pwd_a to log into a without having to recreate the legitimate user's keystroke dynamics. Rather, the use of this recovery program for logging in should be under tighter controls, e.g, an administrator's.

Another possible recovery channel is to employ a more cumbersome but more secure technique for recovering spwd_a. For example, spwd_a could additionally be stored encrypted under a secret passphrase that is required to be substantially longer than 8 characters, under a secret key that can be accessed only with a secure physical token, or, in case a graphical input device is available, under a key generated from a graphical password that is more secure than conventional passwords [42]. This alternative could even be made available for login under normal circumstances, so that the user can log in if she either types her shorter password with her typical keystroke dynamics, or if she can remember the more secure passphrase or graphical password, or has the required token available. However, doing so might hinder the ability of our scheme to gradually adapt to changes in the user's typing patterns, if fewer user logins take place using our scheme.

Chapter 6

Graphical Passwords

*When people choose their own keys, they generally choose poor ones. This is not always due to poor security practices; "Barney" is easier to remember than "*9 (hH/A"*—Bruce Schneider.

In this Chapter we propose and evaluate new authentication schemes that exploit features of graphical input displays to achieve better security than that of traditional schemes. Graphical input devices enable the user to decouple the *position* of inputs from the *temporal order* in which those inputs occur, and we show that this decoupling can be used to generate password schemes with substantially larger (memorable) password spaces. In order to evaluate the security of one of our schemes, we devise a novel way to capture a subset of the "memorable" passwords that, we believe, is itself a contribution.

We explore an approach to user authentication that generalizes the notion of a textual password and that, in many cases, improves the security of user authentication over that provided by textual passwords. We design and analyze *graphical* *passwords*, which can be input by the user to any device with a graphical input interface. A graphical password serves the same purpose as a textual password, but can consist, for example, of handwritten designs (drawings), possibly in addition to text. The devices by which we are primarily motivated are personal digital assistants (PDAs) such as the Palm Pilot$^{\mathrm{TM}}$, Apple Newton$^{\mathrm{TM}}$, Casio Cassiopeia E-10$^{\mathrm{TM}}$, and others, which allow users to provide graphics input to the device via a stylus. More generally, graphical passwords can be used whenever a graphical input device, such as a mouse, is available.

To the best of our knowledge, the notion of a graphical password is due to Blonder [12]. That work proposed a password scheme in which the user is presented with a predetermined image on a visual display and required to select one or more predetermined positions ("tap regions") on the displayed image in a particular order to indicate his or her authorization to access the resource. Beyond this proposal, however, [12] did not further explore the power of graphical passwords or argue security for its particular proposal.

In this Chapter we considerably advance the theory and practice of graphical passwords. We take as a main criterion the need to evaluate graphical passwords' security relative to that of textual passwords. We design two graphical password schemes that we believe to be more secure than textual passwords (and more secure than the scheme of [12]), and we employ novel analysis techniques to make this argument. Moreover, we describe our implementation of one of our graphical password schemes on the Palm Pilot$^{\mathrm{TM}}$.

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

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

Throughout this Chapter we focus on graphical passwords that are exactly repeatable by the user. This distinguishes our work from all works on graphical pattern recognition of which we are aware (see Section 6.1), where it suffices for the device to recognize an input as being sufficiently similar to—but not necessarily the same as—a previously stored input. Because pattern recognition schemes require the storage of (some representation of) the plaintext password on the device, the password is vulnerable to an attacker who captures and probes the device. In contrast, because graphical passwords are repeatable, our schemes can derive a secret key, e.g., to encrypt and decrypt files, without need to store the password on the device. This protects both the password and the encrypted content from the attacker if the device falls into the attacker's hands.

The rest of this Chapter is outlined as follows: In Section 6.1 we overview other password schemes, and put our work in context. In Section 6.2, we present textual passwords with graphical assistance. In Section 6.3, we proceed to purely graphical passwords with a scheme called "draw-a-secret" (DAS). Section 6.3.2 shows our design and implementation of a memo pad encryption scheme based on DAS. Section 6.3.3 proposes novel ways to analyze and estimate the security of DAS and graphical passwords in general. Finally, Section 6.4 concludes.

6.1 Prior Work

There is a considerable amount of prior work on authenticating users via graphical inputs to a device, particularly handwritten signatures (see, e.g., [52, 47, 65]). None of these works strive for exact repeatability by the user, and therefore, a model of the user's graphical input is stored on the device and used to ascertain whether a new input is sufficiently similar to the previously-stored one to grant access. This renders it essential to protect the device's (PDA's) storage from probes by an attacker. In contrast, repeatability is achieved in our schemes, thereby enabling designs in which the device, if captured, is of little help to the attacker.

The techniques in this Chapter can be combined in natural ways with the those discussed in Chapter 2 for strengthening textual passwords—i.e., proactive and reactive password checking, and salting—to improve the security of graphical passwords, as well.

6.2 Textual Passwords with Graphical Assistance

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

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

$\pi(6) = \mathsf{o}$.

Now suppose that the user is presented with a simple graphical input display consisting of, say, eight positions into which to enter a textual password, as illustrated in Figure 6.1. In this figure, step 0 is the initial row of blanks, and steps 1–6 indicate the temporal order in which the user fills in the blanks; i.e., $\pi(i)$ is entered in row i. The password can be placed in the "normal", left-to-right positions as shown in Figure 6.1a. Due to the graphical nature of the input interface, however, the user could enter the password in other positions, as well. For example, Figure 6.1b shows a modification in which the user enters the password in a left-to-right manner, but starting from a different initial position than the leftmost. Figure 6.2c shows entering the password in an "outside-in" strategy. And, of course, these variations can be combined in the obvious way, as shown in Figure 6.2d.

Formally, a k-character graphical password in this scheme can be defined by a total function $\pi' : \{1, \ldots, k\} \rightarrow \mathcal{A} \times \{1, \ldots, m\}$, where $m \geq k$ is the number of positions into which characters can be entered ($m = 8$ in Figure 6.1). If $\pi'(i) = (c, j)$, then this means that the i-th entry (temporally) is the character c in position j. A conventional textual password π, entered in the standard left-to-right way, can be expressed in this scheme as a graphical password π' where $\pi'(i) = (\pi(i), i)$. But as shown in Figure 6.2, more generally we can have variations π' in which $\pi'(i) = (\pi(i), j)$ and $i \neq j$. In fact, it is easy to see that each k-character conventional password π yields $m!/(m-k)!$ graphical passwords π', and indeed this is the factor by which the size of the graphical password space exceeds the k-character conventional password space. This can be a relatively large number: e.g., for $k = 8$

0. _ _ _ _ _ _ _ _	0. _ _ _ _ _ _ _ _
1. t _ _ _ _ _ _ _	1. _ _ _ _ _ _ _ t
2. t o _ _ _ _ _ _	2. o _ _ _ _ _ _ t
3. t o m _ _ _ _ _	3. o m _ _ _ _ _ t
4. t o m a _ _ _ _	4. o m a _ _ _ _ t
5. t o m a t _ _ _	5. o m a t _ _ _ t
6. t o m a t o _ _	6. o m a t o _ _ t
(a) Left-to-right	(b) Rotated left

Figure 6.1: Variations on inputting `tomato`. The word `tomato` can be input in the "normal" left to right manner as shown in (a). Step 0 is the initial row of blanks, and steps 1–6 indicate the temporal order in which the user fills in the blanks. In addition, however, the user can vary the position of the letters in `tomato`. Figure (b) demonstrates shifting the input left by one.

and $m = 10$, this factor is approximately 2×10^6.

Of course, there are far fewer than 2×10^6 variations of each 8-character password that are memorable for human users. However, it is easy to derive a convincing lower bound on the improvement this achieves over a conventional password scheme. It is conservative to assume that the m positional rotations of a password, plus perhaps a handful of others (e.g., reversal, outside-in, inside-out, evens-then-odds, odds-then-evens), and combinations thereof, are memorable, because the choices of position involved in these cases can be derived from simple algorithms

```
0. _ _ _ _ _ _ _ _          0. _ _ _ _ _ _ _ _

1. t _ _ _ _ _ _ _          1. _ _ _ _ _ _ _ t

2. t _ _ _ _ o _ _          2. _ _ _ _ o _ _ t

3. t m _ _ _ o _ _          3. m _ _ _ o _ _ t

4. t m _ _ a o _ _          4. m _ _ a o _ _ t

5. t m t _ a o _ _          5. m t _ a o _ _ t

6. t m t o a o _ _          6. m t o a o _ _ t
```

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

Figure 6.2: Other strategies for inputing `tomato`. Figure (c) represents an outside-in input strategy, and figure (d) is the combination of (a)-(c).

that are more memorable than the positions themselves. (We will return to this characteristic of memorability in the next section.) The attacker's work load will thus be increased by a factor of at least m. An important feature of this scheme is that it is at least as strong as the initial textual password that was chosen by the user, assuming that users do not reduce the size of the space of character sequences that they choose in response to the need to remember a positional order.

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

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

6.3 The Draw-a-Secret (DAS) Scheme

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

The most compelling reason for exploring the use of a picture-based password scheme is that humans seem to possess a remarkable ability for recalling pictures (i.e., line drawings and real objects). The "picture effect", that is, the effect of pictorial and object representations on a variety of measures of learning and memory has been studied for decades [16, 80, 69, 83, 14]. Cognitive scientists and psychologists have shown that there is a substantial improvement of performance in recall and recognition with pictorial representations of to-be-remembered material than for verbal representations.

Superiority in recall of objects over words in immediate recall and over short retention intervals has been demonstrated through a number of experiments. Empirical evidence of the power of pictures over words dates back to the early 1800s; experiments performed by Calkins [16] showed the recall of words declining by 50% or more over a 72 hour retention interval, and recall of objects dropping by less than 20% over the same period. Studies exhibiting strikingly high differences in memory recall of pictures over words have since been replicated on numerous occasions [80, 83, 67, 13]. Some theories that have been proposed to explain these experimental results are outlined in Appendix 6.3.3.

6.3.1 Password Selection and Input

Consider an interface consisting of a rectangular grid of size $G \times G$. Each cell in this grid is denoted by discrete rectangular coordinates $(x, y) \in [1, \ldots, G] \times [1, \ldots, G]$. Suppose that the the user is given a stylus with which she can draw a design on this grid. The drawing is then mapped to a sequence of coordinate pairs by listing the cells through which the drawing passes in the order in which it passes through them, with a distinguished coordinate pair inserted in the sequence for each "pen up" event, i.e., whenever the user lifts the stylus from the drawing surface. For example, consider the drawing in Figure 6.3. Here, the coordinate sequence generated by this drawing is

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

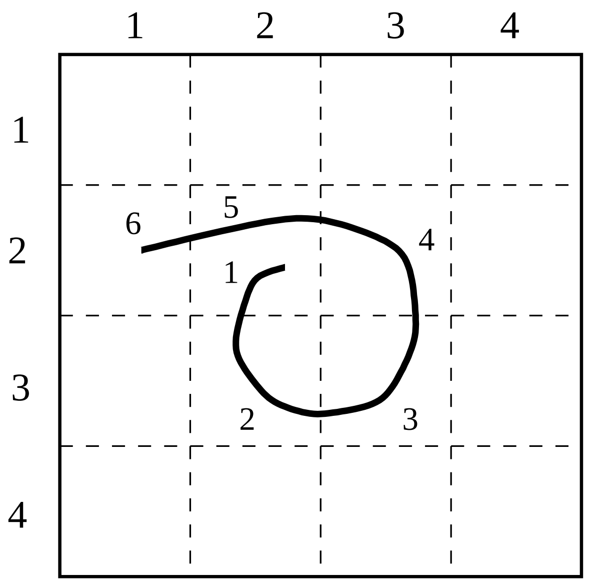


Figure 6.3: Input of a graphical password on a 4×4 grid. The drawing is mapped to a sequence of coordinate pairs by listing the cells in the order which the stylus passes through them, with a distinguished coordinate pair inserted in the sequence whenever the stylus is lifted from the drawing surface.

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

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

As with the scheme of Section 6.2, this scheme is most viable if the user's strokes are echoed as curves while they are drawn. Again we appeal to the maneuverability of the devices we are targeting (i.e., PDAs) to support the restriction that the user must shield the input display from onlookers.

Our requirement of repeatability constrains the parameters of this scheme. As long as the user's current drawing lies in the same equivalence class as the original drawing, she has successfully repeated a chosen password. In general, this gives the user sufficient tolerance when (involuntarily) varying the drawing, provided that the cells of the grid are not too small. Indeed, this was the purpose of separating the drawings into equivalence classes to begin with. Difficulties might arise however, when the user chooses a drawing that contains strokes that pass too close to a grid-line. In those cases, the user might vary the drawing in such a way as to change the resulting sequence of coordinates. We consider the following two possibilities to address this problem: (1) The user is offered to view the internal representation, depicting the path of cells, when she chooses a password so that she can confirm which cells were actually touched by the drawing. (2) The system does not accept a drawing which contains strokes that are located "too close" to a grid line. In the implementation, described in Section 6.3.2, we offer both alternatives.

6.3.2 Application of DAS: An Encryption Tool for a PDA

Our graphical password schemes are motivated primarily by PDAs that offer graphical input capabilities. We now describe our implementation of a memo pad encryption tool for the Palm Pilot that uses a user-input graphical password to derive the encryption key. Either of the schemes of Sections 6.2 and 6.3 could be used to enter the password. Here we illustrate our tool using the DAS scheme, which we have implemented and use regularly.

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

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

An adversary who captures the PDA can presumably obtain all of the ciphertext encrypted under k, and since p is either public or stored in plaintext on the device, the adversary has at least one known plaintext/ciphertext pair with which to attack E. For a strong encryption scheme E, however, the best bet for the attacker remains to guess the original password, which, as we will show in Section 6.3.3, on average is likely to be much harder than if the attacker were faced with attacking a textual password.

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

The interface for our DAS implementation is shown in Figure 6.4. Our application shares the database of the `memopad` application, and allows a user to encrypt/decrypt records in the database based on a user specified drawing. Our implementation conforms to the methodology outlined in Section 6.3.2, with SHA-1 as the cryptographic hash function and 3DES (based on Ian Goldberg's crypto library for his port of SSLeay for the Pilot. See `http://www.isaac.cs.berkeley.edu/pilot`) as the symmetric encryption scheme.

6.3.3 Security of the DAS Password Scheme

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

High information content renders a password scheme more or less invulnerable. For example, if users did in fact choose passwords uniformly from the space of all textual passwords, successful attacks would be extremely unlikely. What is it that renders such attacks successful in practice? There are two factors. The first is that in reality users do not choose their passwords uniformly. If we assume that the data collected in Klein's study [45] is representative of the general population, then users in fact use only 10^{-8} of the possible passwords 25% of the time. Such a distribution is highly peaked, and the information content of the textual password space is correspondingly reduced.

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

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

The Size of the Password Space

First we consider the raw size of the password space, or in other words, its information content assuming users are equally likely to pick any element as their password. The raw size is an upper bound on the information content of the distribution that users choose in practice. We need some way to delimit the password space in order to obtain a finite answer, or in probabilistic terms, a way to ascribe probability zero to an infinite subset of passwords, leaving a finite subset which we will count. We will assume that all passwords of total length (as defined in Section 6.3.1) greater than some fixed value have probability zero. We compute the size $\Pi(L_{max}, G)$ of the space of passwords of total length less than or equal to L_{max} on a grid of size $G \times G$. Π is defined in terms of the number of passwords with total length equal to L, $P(L, G)$ by:

$$\Pi(L_{max}, G) = \sum_{L=1}^{L_{max}} P(L, G) \tag{6.1}$$

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

$$P(L, G) = \sum_{l=1}^{l=L} P(L - l, G) N(l, G) \tag{6.2}$$

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

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

$$N(l, G) = \sum_{(x,y) \in [1,\ldots,G] \times [1,\ldots,G]} n(x, y, l, G) \tag{6.3}$$

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

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

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

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

Putting the pieces together, we can calculate the size of the password space. The results for different upper bounds on total password length on a 5×5 grid are given in Table 6.1.

Table 6.1: Number of passwords of total length less than or equal to L_{max} on a 5×5 grid.

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

The data in Table 6.1 shows that the raw size of the graphical password space surpasses that of textual passwords for reasonable password configurations. While these numbers are encouraging, in practice not all graphical passwords are equally likely to be chosen by users, rendering a uniform distribution overly optimistic. For example, although the number of passwords of length greater than or equal to 12 is already greater than the number of textual passwords of 8 characters or less constructed from the printable ASCII codes ($95^8 \approx 2^{53}$), this includes all possible combinations of twelve isolated dots.

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

Modeling User Choice

We assume that the reason that users choose from such a small subset of textual passwords is that the passwords in that set are more memorable than those outside it. That lack of imagination on the part of the user is not the cause for the lack of variety is supported by the fact that system-generated passwords have been so unsuccessful [7]. By making the same assumption about DAS passwords, we can "reduce" our task to that of modeling the set of "memorable" graphical passwords. If we can show that this set, or some subset of it, has cardinality larger than the dictionary of textual passwords from which users typically choose, we can plausibly claim that as far as information content goes, DAS is more secure than conventional textual password schemes. Here, we identify two such subsets using different criteria of memorability, and show that the cardinalities of these sets do indeed satisfy the above criterion.

What constitutes a memorable password? In the textual case, one obvious component is semantic content. If the sequence of characters has a meaning for the user, the password is more likely to be memorable [58, 80, 13]. This semantic definition is extremely hard, if not impossible, to characterize in the abstract. It is only because the semantic content of many character combinations has been established by the common use of a written language that we can talk about such content at all. In the DAS scheme, there are obvious password components that have meaning, but it is impossible *a priori* to identify exactly which passwords will have semantic content, and to how many users, precisely because it is not a representation with meanings established by common use.

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

$$R(G) = \binom{G}{2}^2 = \frac{1}{4}\, G^2(G-1)^2 \tag{6.4}$$

Each of these rectangles can be generated in many ways. For example, the starting point of a stroke can be at any of the corners, and the stroke direction can be clockwise or counter-clockwise. This yields 8 possibilities for each rectangle. In addition, one can choose whether to close the rectangle by returning to the starting cell or not, again doubling the possibilities. On a 5×5 grid, this amounts to 1600 possible strokes. Two such strokes in succession gives 2.56×10^6 passwords, already roughly the size of the textual dictionary that contained the passwords of 25% of users in Klein's study [45]. Clearly we can generate a much larger set of passwords by considering variations on the theme of rectangles, or by considering other Gestalt forms [88].

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

- Common-code theory: this view of memory and recall theorizes that pictures and words access semantic information in a single conceptual system that is neither word-like or picture-like. This theory hypothesizes that pictures and words both require analogous processing before accessing semantic information, but pictures require less time than words for accessing the common conceptual system. Common-code theorists attribute better picture recall to differences in the encoding of pictures and words: pictures share fewer common perceptual features among themselves and therefore need to be discriminated from a smaller set of possible alternatives than words. The greater number of dictionary meanings or the greater lexical complexity of words create uncertainty and confusion, and hence poorer recall.
- Dual-code theory: unlike the common-code approach, this theory postulates that language and knowledge of worlds are represented in functionally distinct verbal and nonverbal memory systems. The verbal system is specialized for dealing with linguistic information whereas the non-verbal stores perceptual information. The most evident examples of dual process theory can be found in experiences that we have all had at some time or the other: we meet someone, know that person to be familiar but do not know who they are; we recognize a melody, but fail to remember its name or when or where we heard

it before; we read a line of a poem, know it, but do not know where we have read it before, much less the title or author of the poem. In all these cases, we experience a sense of familiarity, but have—at least at first— no access to any contextual or conceptual information [57].

Dual code theory suggests that there are qualitative differences between the ways words and pictures are processed during memory and hypothesizes that the reason for superior picture memory is that pictures automatically engage multiple representations and associations with other knowledge about the world, thus encouraging a more elaborate encoding than occurs with words [69, 70].

- Abstract-propositional theory: in contrast to the dual-code approach, this theory rejects any notion of sophisticated distinctions between verbal and nonverbal modes of representation, but instead describes representations of experience or knowledge in terms of an abstract set of relations and states, called propositions. This theory postulates that better free recall with pictures may be due to even more elaborative encoding effects than those suggested by dual-code theorists. Propositional theorists view the involvement of *abstractive and interpretive* processes in picture memory as the explanation for the picture effect [54]. Therefore, a series of line drawings will be poorly remembered if a subject is unable to interpret the drawings in a meaningful way, whereas memory for the same drawings, presented in the same way will be much better if a conceptual interpretation is provided, and it this interpretive process which is responsible for better picture memory recall.

While the strongest evidence thus far for the picture effect can be best explained by dual-code theory (see [57]), an understanding of picture memory and the means by which we acquire and maintain information about the visual environment is still an ongoing challenge. Nonetheless, the research to date provides strong arguments in terms of the memorability of drawings over words in recognition tasks and hence its applicability to computer security.

Memorability based on short algorithms The second set of passwords that we describe is suggested by the discussion of text-based graphical passwords in Section 6.2, which pointed toward a different definition of memorability. There, a memorable sequence of positions seemed characterized by the fact that there existed a short algorithm to describe the sequence. It is this definition of memorable that we wish to apply here, since it can be characterized in precise terms. We do not argue that every memorable password has a short algorithm to describe it, but that passwords describable by short algorithms are memorable. We will show that the cardinality of this subset of memorable passwords is already larger than the dictionary of character sequences from which users most often draw their passwords, and that therefore, following the argument above, the DAS password scheme should be harder to crack in practice than the conventional textual scheme.

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

Before describing the results of this endeavor, we give some details of the language in which we generated the programs. The grammar of the language is as follows:[1]

$$
\begin{array}{rcl}
\text{program} & \rightarrow & \text{digit digit block} \\
\text{block} & \rightarrow & \text{statement block} \\
\text{statement} & \rightarrow & \text{instr} \mid \textbf{repeat} \;\; \text{digit} \;\; \text{block} \;\; \textbf{end} \\
\text{instr} & \rightarrow & \textbf{up} \mid \textbf{down} \mid \textbf{right} \mid \textbf{left} \mid \textbf{penup} \mid \textbf{pendown} \\
\text{digit} & \rightarrow & \textbf{1} \mid \textbf{2} \mid \textbf{3} \mid \textbf{4} \mid \textbf{5}
\end{array}
$$

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

To calculate the complexity for a given program, we assign a complexity to each

[1]Those readers old enough to remember the APPLE II will recognize that our language bears a striking resemblance to Turtle Graphics, the children's language based on LOGO (see, e.g., [1]).

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

The results of using the above described procedure for counting the number of DAS passwords of a given complexity on a 5×5 grid are shown in Figure 6.2. As expected, this data shows that the number of DAS passwords grows exponentially as a function of the maximum complexity of the program. What is more interesting, however, is that by extrapolation[2] we see that the number of DAS passwords generated by programs of only complexity 12 far surpasses the dictionary size of

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

approximately 3×10^6 used in Klein's password-cracking studies [45]. As a point of comparison, even just tracing the outermost cells of a 5×5 grid to make a square already requires a program of complexity at least fifteen in our simple language. And, obviously this design and many other, more complex ones will fall in the realm of memorable for most users. We believe that this is compelling evidence that DAS passwords, of which those generated by programs of complexity at most twelve are but a very small subset, will be significantly harder to crack in practice than textual passwords.

Table 6.2: Number of DAS passwords generated by programs of short complexity on a 5×5 grid.

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

Lack of Knowledge of the Distribution

Given the size of typical password spaces, knowledge of the distribution of user passwords is essential to an adversary. Without such knowledge the adversary has no way of directing her search toward more probable passwords, and is no better off than if users really did pick their passwords uniformly from the set of possibilities [18].

Where did the knowledge of the distribution come from in the case of textual passwords? For the most part, dictionaries have been compiled by using reasonable assumptions about likely choices. The assumptions stem from the use of a shared language, and shared knowledge of the semantic content of words. For example, in the work of Klein [45] the sources for likely passwords included the St. James Bible, the UNIX dictionary, and many other sources of English words that were available to the author precisely because they are a part of our language. If these assumptions had turned out to be incorrect, textual password schemes would be extremely difficult to break in practice.

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

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

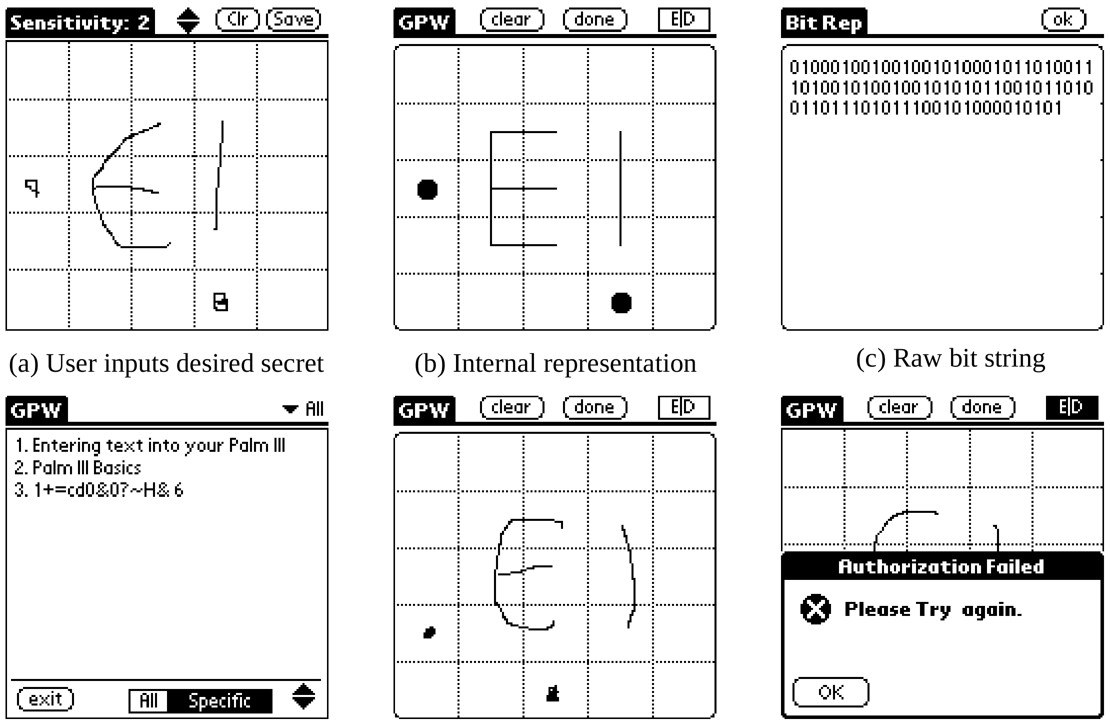
6.3.4 Summary

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

6.4 Conclusions

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

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



(a) User inputs desired secret

(b) Internal representation

```
010001001001001010001011010011
101001010010010101011001011010
011011101011100101000010101
```

(c) Raw bit string

1. Entering text into your Palm III
2. Palm III Basics
3. 1+=cd0&0?~H& 6

(d) Interface to database

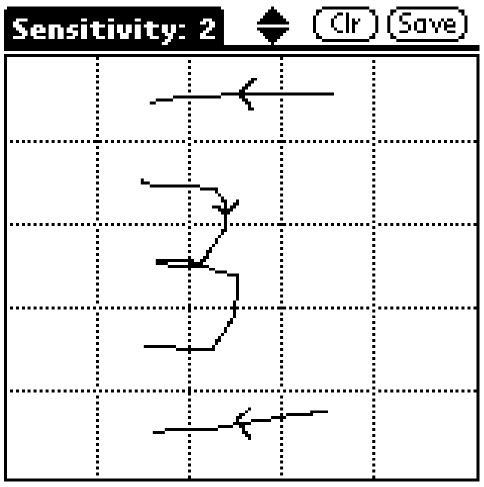
(e) Re-entry of secret

Authorization Failed

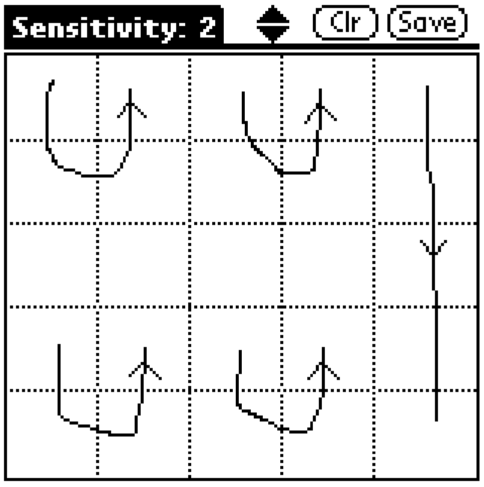
Please Try again.

(f) Authorization failed

Figure 6.4: A password is created by drawing the secret on the display as shown in (a). Both the internal representation of the input password showing the cells covered by the user's drawing and the derived key are depicted in (b) and (c) respectively. To apply a symmetric cryptographic function to records in the database (shown in (d)), the user selects the records and then re-inputs the DAS password. If the encryption of a known clear-text with the input password matches the stored ciphertext created during initialization, then the symmetric cryptographic routine, $E_k(x)$, is applied to the selected records. Otherwise, the user is prompted to re-enter the DAS secret.



```
2 2
pendown
repeat 2
right
down
left
end
penup
right
right
down
pendown
left
left
penup
repeat 4
up
end
right
right
pendown
left
left
penup
```



```
1 1
repeat 2
pendown
down
right
up
penup
left
repeat 3
down
end
pendown
down
right
up
penup
repeat 3
up
end
right
end
pendown
repeat 4
down
end
penup
```

Figure 6.5: Example DAS passwords and the shortest programs that generate them are shown above. The complexities are 24 and 26 respectively.

Chapter 7

Summary

We presented techniques for strengthening the security of password-based authentication. Specifically, we introduced a novel approach for improving the security of password-based authentication that exploits patterns in a user's normal typing rhythm as she inputs her password to the login device. Our technique enables the generation of a long-term secret (that is significantly larger than the password itself) that has many potential uses, e.g., file encryption. Our method increases substantially the time for an off-line attacker to exhaustively search for this secret (or the password itself) and, where appropriate, can be used in conjunction with other techniques, such as salting, to slow the attacker even further. In addition, our approach improves security against an on-line attacker who learns a user's password (e.g., by observing it being typed) and attempts to log in as the legitimate user. Unless the attacker can mimic the keystroke behavior of the legitimate user—which [48, 49, 39, 4, 43, 6, 15, 55, 2, 61, 75] suggest is unlikely—the attempted login will fail.

We have also shown that our technique for improving the security of password-based authentication is viable in other ways:

- It adapts to gradual changes in a user's typing pattern over time, while generating the same long-term secret across logins. Since we use a behavioral characteristic as part of the process of corroborating a user's identity, adapting to changes in typing behavior is important because as a user becomes more familiar with her password, her typing patterns will change over time.
- It allows for control of error probabilities. A trade-off between security and usability can be met by adjusting the number of standard deviations away from the mean typing value for feature instances used in calculating a user's set of distinguishing features. Increasing this system parameter makes it easier for a legitimate user to log in reliably to her account, but it also increases the probability of success for an on-line attacker.
- It is usable in practice. Using empirical analysis of user keystroke data we showed that our scheme is easy to use by the average user. There remains a risk in our scheme that due to a sudden shift in typing behavior, a user will be unable to log into her account. We discussed a number of recovery mechanisms for this event. To minimize this risk we suggest that our technique be restricted to local logins using the same keyboard.

In addition, we presented an alternative approach to user authentication that is based on exploiting the input capabilities of graphical devices such as Personal Digital Assistants. We presented two graphical authentication schemes that are more secure than traditional approaches, and used novel analysis techniques to make this argument. The first graphical password scheme presented builds directly on textual password schemes, by enhancing the input of textual passwords using graphical techniques. We showed that even a conservative estimate of the variations introduced by the graphical input yields a substantial improvement in strength over the purely textual version.

The second scheme presented is based on a purely graphical approach called Draw-a-Secret (DAS), in which a user's password consist of line drawings on a rectangular grid. To reason about improvements in security over textual passwords, we defined a class of graphical passwords that, plausibly, are memorable for users. We considered the class of passwords that can be generated by a short program in a simple grid language and showed that even the set of DAS passwords generated by very short programs already has cardinality larger than the dictionaries used to crack a high percentage of the textual passwords chosen by users.

7.1 Future Directions

A possible direction for future work is to pursue a more thorough analysis of the security improvements offered by the techniques presented herein. In particular, although the technique presented for strengthening the security of password-based authentication is provably at least as secure as conventional password-based authentication, we believe that the data used in our empirical analysis was not particularly well-suited to analyzing our scheme. We believe the results of our analysis to be pessimistic and therefore an interesting direction of future work is to conduct more targeted studies to see if they yield more optimistic results than the (already positive) results of Section 5.8. We conjecture that this will indeed be the case.

With regard to our work on graphical passwords, different schemes for modeling the memorability of DAS passwords can be explored. For example, a model that captures high-level structure more intuitively than our current models may lead to a better understanding of the difficulties that an off-line attacker would face. Here, the idea would be to capture the concept of organized drawings, in which the view of an entire object is more than just the sum of the individual parts that constitute it—one can view a square as an object in itself and not simply as an arrangement of the individual lines from which it is composed. In this way, it is possible to define a set of primitive structures from which all memorable drawings can be derived using meta-level compositions of these primitives. We hypothesize that even a reduced set of DAS passwords (for some reasonable number of primitives) will constitute a much larger space than that of textual-based passwords, and as such, will be significantly harder to crack in practice.

Finally, we note that though we presented our technique for strengthening the security of password-based authentication in the context of keystroke dynamics, other biometric information could obviously be used in place of (or in addition to) keystroke durations and latencies. Graphical input displays supporting handwritten character recognition, e.g., as offered by the Palm Pilot, offer a wealth of possibilities, where handwriting dynamics such as slant of characters and speed of writing could be used in conjunction with the techniques presented herein.

Chapter 8

Bibliography

[1] H. Abelson, J. Bamberger, I. Goldstein and S. Papert. *Logo Progress Report.* MIT AI memo no. 356, 1975.

[2] T. J. Alexandre. Biometrics on smart-cards: an approach to keyboard behavioral signature. In *Proceedings of the* 2^{nd} *Smart Card Research & Advanced Applications Conference*, 1996.

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

[4] S. Bleha, C. Slivinksy and B. Hussein. Computer access security systems using keystroke dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990.

[5] S. Bleha. Dimensionality reduction and feature extraction applications in identifying computer users. *IEEE Transactions on System, Man and Cybernetics*, 1991.

[6] S. Bleha and M. S. Obaidat. Computer users verification using the Perceptron algorithm. *IEEE Transactions on System, Man and Cybernetics*, 1993.

[7] M. Bishop. Password management. In *Proceedings of the International Computer Conference*, 1991.

[8] M. Bishop. Improving system security via proactive password checking. *Computers & Security*, 1995.

[9] S. M. Bellovin and M. Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1992.

[10] S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise. In *Proceedings of the* 1^{st} *ACM Conference on Computer and Communications Security*, 1993.

[11] G. R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the National Computer Conference*, 1979.

[12] G. Blonder. *Graphical passwords.* United States patent no. 5559961, 1996.

[13] M. A. Borges, M. A. Stepnowsky and L. H. Holt. Recall and recognition of words and pictures by adults and children. *Bulletin of the Psychonomic Society*, 1977.

[14] G. H. Bower, M. B. Karlin and A. Dueck. Comprehension and memory for pictures. *Memory and Cognition*, 1975.

[15] M. Brown and S. J. Rogers. User identification via keystroke characteristics of typed names using neural networks. *International Journal of Man-Machine Studies*, 1993.

[16] M. W. Calkins. Short studies in memory and association from the Wellesley College Laboratory. *Psychological Review*, 1898.

[17] R. Chellappa, C. L. Wilson and S. Sirohey. Human and machine recognition of human face images: a survey. In *Proceedings of the IEEE*, 1995.

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

[19] E. Cureton. *Factor analysis, an Applied Approach.* Erlbaum Associates, New Jersey, 1983.

[20] J. G. Daugman. High confidence visual recognition of persons by a test of statistical independence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1993.

[21] P. J. Davis. *Interpolation and Approximation.* Blaisdell, London, 1963.

[22] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 1976.

[23] G. R. Doddington. Speaker recognition—identifying people by their voices. *Proceedings of the IEEE*, 1985.

[24] R. Duda. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[25] D. Feldmeier and P. Karn. UNIX password security—Ten years later. *Lecture Notes in Computer Science*, 1990.

[26] R. Gaines, W. Lisowski, S. Press and N. Shapiro. Authentication by keystroke timing: some preliminary results. *Rand report R-256-NSF.* Rand Corporation, 1980.

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

[28] Gentner. Keystroke timing in transcription typing. *Cognitive aspects of skilled typewriting*, 1993.

[29] S. Goldwasser, S. Micali and C. Rackoff. The knowledge complexity of interactive proof systems. In *Proceedings of the* 17^{th} *ACM Symposium on Theory of Computing*, 1985.

[30] L. Gong, T. M. Lomas, R. M. Needham and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 1993.

[31] L. Gong. Optimal authentication protocols resistant to password

guessing attacks. In *Proceedings of the* 8^{th} *IEEE Computer Security Foundations Workshop*, 1995.

[32] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. In *Proceedings of the* 39^{th} *IEEE Symposium on Foundations of Computer Science*, 1998.

[33] S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. In *Proceedings of the* 5^{th} *ACM Conference on Computer and Communications Security*, 1998.

[34] N. Haller. The s/key$^{\text{TM}}$ one-time password system. In *Proceedings of the Network and Distributed System Security Symposium*, 1994.

[35] F. B. Hildebrand. *Introduction to Numerical Analysis.* McGraw Hill, London, 1956.

[36] D. Hoover. Software smart cards via cryptographic camouflage. To appear in *Proceedings of the IEEE Symposium on Security and Privacy*, 1999.

[37] G. Horng. Password authentication without using a password table. *Information Processing Letters*, 1995.

[38] P. J. Huber. *Robust Statistics.* Wiley, 1981.

[39] B. Hussien, R. McLaren and S. Bleha. An application of fuzzy algorithms in a computer access security system. *Pattern Recognition Letters*, 1989.

[40] D. K. Isenor and S. G. Zaky. Fingerprint identification using graph matching. *Pattern Recognition*, 1986.

[41] D. Jablon. Strong password-only authenticated key exchange. *ACM Computer Communications Review*, 1996.

[42] I. Jermyn, A. Mayer, F. Monrose, M. K. Reiter and A. D. Rubin. The design and analysis of graphical passwords. To appear in *Proceedings of the* 8th *USENIX Security Symposium*, 1999.

[43] R. Joyce and G. Gupta. Identity authorization based on keystroke latencies. *Communications of the ACM*, 1990.

[44] L. G. Kersta. Voiceprint identification. *Nature*, 1962.

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

[46] L. Lamport. Password identification with insecure communications. *Communications of the ACM*, 1981.

[47] F. Leclerc and R. Plamondon. Automatic signature verification: the state of the art. *International Journal on Pattern Recognition and Artificial Intelligence*, 1994.

[48] G. Leggett and J. Williams. Verifying identity via keystroke characteristics. *International Journal of Man-Machine Studies*, 1988.

[49] G. Leggett, J. Williams and D. Umphress. Verification of user identity via keystroke characteristics. *Human Factors in Management Information Systems*, 1989.

[50] R. E. Lennon, S. M. Matyas and C. H. Meyer. Cryptographic authentication of time-invariant quantities. *IEEE Transactions on Communications*, 1981.

[51] C. H. Lin, C. C. Chang, T. C. Wu and R. C. Lee. Password authentication using Newton's interpolating polynomials. *Information Systems*, 1991.

[52] G. Lorette and R. Plamondon. Dynamic approaches to handwritten signature verification. *World Scientific*, 1990.

[53] S. Lucks. Open key exchange: how to defeat dictionary attacks without encrypting public keys. In *Proceedings of the Workshop on Security Protocols*, 1997.

[54] S. Madigan. Picture memory. *Imagery, Memory and Cognition.* Lawrence Erlbaum Associates, 1983.

[55] D. Mahar, R. Napier, M. Wagner, W. Laverty, R. Henderson and M. Hiron. Optimizing digraph-latency based biometric typist verification systems: inter and intra typists differences in digraph latency distributions. *International Journal of Human-Computer Studies*, 1995.

[56] U. Manber. A simple scheme to make passwords based on one-way functions much harder to crack. *Computers & Security*, 1996.

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

[58] G. A. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 1956.

[59] B. Miller. Vital sings of identity. *IEEE Spectrum*, 1994.

[60] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1997.

[61] F. Monrose and A. D. Rubin. Authentication via keystroke dynamics. In *Proceedings of the* 4^{th} *ACM Conference on Computer and Communications Security*, 1997.

[62] F. Monrose and M. K. Reiter. Password strengthening using keystroke dynamics. *Submitted for publication*, 1999.

[63] R. Morris and K. Thompson. Password security: a case history. *Communications of the ACM*, 1979.

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

[65] V. S. Nalwa. Automatic on-line signature verification. *Proceedings of the IEEE*, 1997.

[66] R. M. Needham and M. D. Schroeder. Using encryption for authenti-

cation in large networks of computers. *Communications of the ACM*, 1978.

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

[68] C. Neuman and T. Ts'o. Kerberos: an authentication service for computer networks. *IEEE Communications*, 1994.

[69] A. Paivio, T. B. Rogers and P. C. Smythe. Why are pictures easier to recall than words? *Psychonomic Science*, 1968.

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

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

[72] F. J. Prokoski, R. B. Riedel and J. S. Coffin. Identification of individuals by means of facial thermography. *IEEE Computer*, 1992.

[73] T. Raleigh and R. Underwood. CRACK: a distributed password advisor. In *Proceedings of the* 1^{st} *USENIX Security Workshop*, 1988.

[74] R. L. Rivest. Cryptography. In *Handbook of Theoretical Computer Science*, Elsevier Science, 1990.

[75] J. A. Robinson, V. M. Liang, J. A. Chambers and C. L. MacKen-

zie. Computer user verification using login string keystroke dynamics. *IEEE Transactions on System, Man, and Cybernetics*, 1998.

[76] W. de Ru and Jan H. Eloff. Enhanced password authentication through fuzzy logic. *IEEE Expert*, 1997.

[77] B. Schneier. *Applied Cryptography, 2nd edition.* John Wiley & Sons, 1996.

[78] FIPS. Secure hash standard. *Federal Information Processing Standards Publication 180-1*, U.S. Department of Commerce, 1995.

[79] A. Shamir. How to share a secret. *Communications of the ACM*, 1979.

[80] R. N. Shepard. Recognition memory for words, sentences and pictures. *Journal of Verbal Learning and Verbal Behavior*, 1967.

[81] E. Spafford. Preventing weak password choices. In *Proceedings of the 14th National Computer Security Conference*, 1991.

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

[83] L. Standing. Learning 10,000 pictures. *Quarterly Journal of Experimental Psychology*, 1973.

[84] M. Steiner, G. Tsudik and M. Waidner. Refinement and extension of encrypted key exchange. *ACM Operating Systems Review*, 1995.

[85] J. T. Tou and R. C. Gonzalez. *Pattern Recognition Principles.* Addison-Wesley, 1981.

[86] D. Umphress and G. Williams. Identity verification through keyboard characteristics. *International Journal of Man-Machine Studies*, 1985.

[87] J. E. Wells. Encoding and memory for verbal and pictorial stimuli. *Journal of Experimental Psychology*, 1972.

[88] M. Wertheimer. *Laws of organization in perceptual forms.* Routledge & Kegan Paul, London, 1938.

[89] T. Wu. The secure remote password protocol. In *Proceedings of the Network and Distributed System Security Symposium*, 1998.

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

[91] J. Zhang, Y. Yan and M. Lades. Face recognition: eigenface, elastic matching and neural nets. *Proceedings of the IEEE*, 1997.

Towards Stronger User Authentication

by

Newman Fabian Monrose

Advisor: Zvi Kedem

Password-based authentication is the dominant mechanism for verifying the identity of computer users, even though it is well known that people frequently choose passwords that are vulnerable to dictionary attacks. This dissertation addresses the issue of improving the security of password-based authentication, and presents authentication techniques that are more secure than traditional approaches against off-line attacks.

We present a technique for strengthening the security of a textual password by augmenting it with biometric information such as the duration and latency of keystrokes during entry of the password. Thereby, both the password and the user's typing pattern are used to corroborate the user's identity. The technique presented adapts to gradual changes in a user's typing pattern while maintaining the same strengthened password across multiple authenticated sessions. Moreover, our technique does not reveal which of a user's keystroke *features* are used to generate the corresponding strengthened password. This knowledge is hidden even from an attacker who captures all the system information used by the authentication server, and we show that our technique increases significantly the amount of work such an attacker must perform.

Additionally, we present an alternative technique for user authentication which exploits features of graphical input devices. We propose and evaluate *graphical passwords*, which serve the same purpose as textual passwords, but consist of handwritten drawings, possibly in addition to text. Graphical passwords derive their strength from the fact that graphical input devices allow one to decouple the positions of inputs from the temporal order in which these inputs occur. We use this independence to build new password-based authentication schemes that are convincingly stronger than conventional methods.