User Authentication Using Keystroke Dynamics

Robert B. Gramacy
University of California
at Santa Cruz

Paul McKenna
University of California
at Santa Cruz

November 29, 2001

Abstract

Keystroke dynamics is examined as a means of user verification. Letter digraph latencies are measured and combined into a signature vector. In order to successfully login into the system a user must enter the correct login data and match the latency signature vector with his/her keyboard input. Preliminary results suggest the possibility of a practical implementation using keystroke dynamics and suggests a number of avenues of further research.

1 Introduction

Passwords play a central role in computer security. A username in combination with a well chosen password is the standard means of user authentication, and can provide a high degree of security. The fact remains however that in real world applications users often employ poor judgment when picking passwords. People often need to remember several different passwords for a variety of applications. Secret pass-codes are required for bank accounts, online stock transactions, premium web services, ISP accounts as well as work and school computer accounts.

Overwhelmed by the shear number of secret codes they must remember, users frequently choose easy to remember but easy to guess passwords such as names, dates, or common words. A simple dictionary attack on a computer password file is likely to break a significant number of account passwords.\(^1\)

\(^1\)In one case study of 14,000 passwords almost 24% were

Biometrics has been presented as a means of bolstering or replacing password authentication. Biometrics involves positively identifying an individual based on some physical characteristic. Authentication based on fingerprint, face recognition, and iris scanning have all been examined and to some extent implemented. The problem with such methods however is that they all require special hardware which is impractical in many situations.

2 Keystroke Dynamics in User Authentication

In contrast to other biometric authentication systems, keystroke dynamics offers the promise of high confidence user identification without the need of special hardware. Keystroke dynamics consists of constructing a unique signature for a user based on his/her unique style of typing. In contrast to other types of biometrics, no hardware other than a standard keyboard is needed. Various keystroke dynamics have been tried for creating such a signature.

The obvious source of keyboard metrics that can serve for creating a signature involve a wealth of different timing schemes. A person’s typing speed can be measured; the time between keystrokes recorded; duration of key-presses noted; and a record kept of special key uses such as the backspace key. Any or all of these metrics can be combined to create a so-called signature for a user. It has been observed that certain key-pairs (or digraphs) are much more common than

1


others [4]. For instance ‘zx’ never occurs but ‘ea’ is fairly common. It can be argued that monitoring for these likely key-pairs and recording statistics such as keystroke duration might yield a useful metric.

For our project, we have chosen keystroke latency for all key-pairs as our metric. We record the time between keystrokes for a fixed text chosen by the user during training. The latency information is then used to create the user’s signature.

Assuming a reliable signature can be constructed, the question then becomes how the signature can be used in the user authentication process. One approach seeks to authenticate a user based solely on his/her typing signature with no other verification (e.g. password) [4, 7]. Various attempts have been made to generate the high degree of confidence required by this method. Using complicated signature monitoring and dynamic learning algorithms noteworthy accuracy has been claimed. But while impressive at first glance, none of these attempts have approached the 100% user authentication accuracy required if the signature is to be the only source of verification.

A second approach involves so-called password hardening [9]. With this method a user types in his/her user name and password as usual. However instead of simply checking whether the password is correct, the login program encrypts the password using keystroke dynamical features to produce a hardened password. A match is then attempted with the previously encrypted password on record. It is argued that encryption using the keystroke signature provides a higher degree of security. Furthermore, it is argued that an attacker will not have access to a user’s keystroke signature and as such will not have any ready means to crack the password file. In the more familiar case of a normally encrypted UNIX password file, an attacker who gains access to the file stands a fairly good chance of cracking certain passwords using simple, known attacks.

A third method involves using the keystroke signature of a user in combination with a password for user authentication. In this approach, the user must enter the correct string(s) for authentication as well as enter them in a way that matches the keystroke signature for that user. In this way, the keystroke signature acts much as a secondary password. A user must correctly enter both the primary password and the secondary (keystroke) password to gain access. This is the path we will explore.

3 Implementation

We have chosen to use keystroke latency as our metric for user authentication. We created an artificial login environment where the user is prompted for a username, password, full name, and a static phrase of their choosing. If the data is entered correctly and the keystroke latencies match, the user is allowed access. If not, the user may repeat the login process.

Accurate authentication relies on a unique latency signature for each user. Keystroke latency, or any other keystroke metric, needs a training period in order to develop a model of the user’s characteristic signature. In our implementation, we have chosen a straight-forward training method in which the user initially enters all his/her input data several times. Each input, such as username, is monitored for key-pair latency and the data is recorded in a latency vector. The result after training is a set of latency vectors for each of the user-name, password, full name, and phrase tokens. A string along with its digraph latency information is referred to as a token. The vectors for each token (i.e. username, phrase, etc.) are combined into a signature. The mean and standard deviation for each signature is calculated. These means and standard deviations are collected into two additional vectors which fully characterize the user’s signature. That is, as of now the mean and standard deviation for each signature are the only statistics used for validation.

Figure 1 contains sample login signatures from two user profiles. As mentioned above, the user profile is comprised of four such signatures (username, password, full name, and a phrase) each of which is a collection of latency vectors summarized by mean and standard deviation statistics. From the figure it is

\[\text{in practice, no latency information should be collected for the password. Keystroke latency for a password could provide an attacker with valuable information to crack the password.}\]
clear that keystroke latencies follow a pattern – perhaps with an occasional outlier.

For subsequent logins, latency information is collected from the keyboard and compared with the signature on file. For a given key pair, we chose to accept any input within 1.0 standard deviations of the mean as a match for that key pair. The percentage of the pairs that match (i.e. login likelihood) is a measure of the difference between the login tokens and signatures in a user’s profile. If at least 60% of the pairs (from all four tokens) match, we assume that the user is authenticated. (Other values are being experimented with – see the Results & Discussion section.) If the login likelihood is below the authentication threshold (e.g. 60%), the user is prompted to re-enter the information. If however the user enters the correct information with the correct latency signature, access is granted. For logins which are particularly high confidence, the login statistics might then be incorporated into the signature (see Section 4). This experimental system allows an unlimited number of attempts to gain access. In practice there would likely be some kind of maximum number of attempts before the user would be locked out. Additionally, in a real world system all attempts at access would be logged for later scrutiny.

Previous work in this area [7] used similar pairwise validation criterion (60%) but required that in order for a pair to be counted it must fall within a stricter threshold of 0.5 standard deviations. However, profiles from the previous study contained fewer prototypes (roughly eight) with outliers pruned. Our profiles contain many more prototypes with no explicit pruning, which yields much larger standard deviations in the signature. We will discuss this later in the paper when we examine our results and the topic of adaptive learning.

Figures 2 and 3 show the validation decision of two different users trying to login into the same account. In each figure the latencies given at the login prompt are compared against each of the four signatures’ mean and standard deviation statistics. Figure 2 shows a successful login by “p.mckenna” whereas in the figure 3 an intruder (“bobby”) unsuccessfully attempts to break into the same account. In the case of the successful login the statistics are as follows:

<table>
<thead>
<tr>
<th>Valid Tokens</th>
<th>Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>login</td>
<td>87.5%</td>
</tr>
<tr>
<td>password</td>
<td>100%</td>
</tr>
<tr>
<td>full name</td>
<td>76.9%</td>
</tr>
<tr>
<td>phrase</td>
<td>92.9%</td>
</tr>
<tr>
<td>total</td>
<td>89.3%</td>
</tr>
</tbody>
</table>

For the intrusion attempt we have:

Figure 1: (Top) User login “p.mckenna” latencies (29) and (Bottom) “bobby” (25) with mean and standard deviation statistics superimposed.
Figure 2: A successful login to the account “pmckenna”. The latencies associated with login, password, full-name, and phrase are compared with the mean and standard deviation statistics stored in the profile. Notice how the latency line *jumps through* most of the mean/sd loops.

Figure 3: A unsuccessful intrusion attempt on the account “pmckenna” by user “bobby”. The latencies associated with login, password, full-name, and phrase are compared with the mean and standard deviation statistics in the stored profile. Notice how the latency line *misses* most of the mean/sd loops.

<table>
<thead>
<tr>
<th>Invalid Tokens</th>
<th>Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>login</td>
<td>37.5%</td>
</tr>
<tr>
<td>password</td>
<td>25%</td>
</tr>
<tr>
<td>fullname</td>
<td>23.1%</td>
</tr>
<tr>
<td>phrase</td>
<td>21.4%</td>
</tr>
<tr>
<td>total</td>
<td>26.8%</td>
</tr>
</tbody>
</table>

In this example we see from the figures and the tables a clear discrimination between a valid login and an intrusion attempt. The distinction may not always be so clear. Many factors may affect the login pattern of a legitimate user or the robustness of a user’s signature. Fatigue, illness, and keyboard variations all may affect the ability of a user to generate the correct latency signature. In addition, an unskilled typist is likely to exhibit a broad range of latencies for the various digraphs. Such variability may generate a broad standard deviation which is easier for an attacker to match by chance. We present in the next section a discussion of learning which may in fact solve or at least mitigate some of these issues. However other issues surrounding keystroke authentication may not be so easily remedied.

4 Learning: refining signatures over time

The latency signature as described so far is static. A pattern is established during the training period and this pattern persists for the life of the account or until retraining. This may be acceptable for an expert typist with a consistent typing pattern but may not be optimal for the less experienced or erratic typist. For such a user, it is desirable that the signature not be static but rather that it evolve with time. Slight changes in the user’s typing pattern should be re-
fected in an evolution of the latency signature. How to incorporate additional information into the user’s latency signature is a complicated question. We have chosen to use a simple method of combining high confidence authentications into the existing signature statistics. The measured login latencies are combined with the corresponding signature means to produce an updated mean with the appropriately revised standard deviation. It is hoped that this simple combination would result in a learning process which produces a more robust yet adaptive signature.

In addition, other parameters can be learned over time. Just as the standard deviation and means of signatures migrate over time, so might it be acceptable that the standard deviation and validate thresholds also migrate. If we agree that a user’s signature solidifies over time (becomes more consistent) it then makes sense to apply a more strict validation criterion as latency fluctuations diminish. The resulting constrained window of acceptance should result in an increase in security.

On the other hand, if it becomes clear that a valid user could be frustrated by validation criterion which are too strict, we should be willing to loosen the acceptance somewhat to allow for easier logins with less false negatives. Although we have not implemented standard deviation migration over time, it would be interesting to experiment with different implementations and the resulting gain (or loss) of security. We will come back to these ideas again in the following sections.

5 Results & Discussion

As hinted at above, one of the most prominent results of our experiments is that different users have varying success with our evolved login process. This is likely due to the fact that the user is actively involved with the authentication process unlike with other biometric techniques (fingerprint, iris scan, etc). Typing four fields in consistently can be quite taxing; and for the novice at the keyboard, quite frustrating.

Nonetheless our results are quite encouraging. However, most of our data was gathered from users with a fair amount of typing experience.

In general, our results show that as more logins are incorporated into a user’s profile both the false negatives (unsuccessful valid logins) and false positives (successful intrusion attempts) become less likely. The reason for this is likely twofold.

1. The profile signatures are more robust as their statistics are comprised of more prototypes.

2. The user has become more familiar with the login process and their own biometric. In other words, they consciously work to create a consistent style.

For some users it is has been shown that large standard deviation thresholds lead to fewer false negatives without a notable increase in false positives. In general the false negative rate is low. However, sufficient experiments to support these results (or a functional relationship) have yet to be carried out. Joyce and Gupta [7] have noted similar results.

Our results also indicate that longer training periods can do a great deal to compensate for lack of consistency between a user’s logins. We will try to summarize our findings below. Unfortunately, due to time constraints most of our experiments are informal, so most of the following results should be taken as preliminary. The subjects in our experiments are close friends of ours (not a random sample). There were a total of seven subjects, some contributing much more than others. Some informal statistics follow (number of login attempts (after training), login successes (after training), number incorporated (latencies in profile), percent of successful intrusion attempts.) Each user had approximately 8 rounds of training.

<table>
<thead>
<tr>
<th>username</th>
<th>att</th>
<th>suc</th>
<th># inc.</th>
<th># int.</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmckenna</td>
<td>60</td>
<td>45</td>
<td>29</td>
<td>2%</td>
</tr>
<tr>
<td>bobby</td>
<td>50</td>
<td>43</td>
<td>33</td>
<td>0%</td>
</tr>
<tr>
<td>jmefford</td>
<td>25</td>
<td>15</td>
<td>8</td>
<td>2%</td>
</tr>
<tr>
<td>fire</td>
<td>15</td>
<td>13</td>
<td>10</td>
<td>0%</td>
</tr>
<tr>
<td>mark</td>
<td>30</td>
<td>15</td>
<td>8</td>
<td>0%</td>
</tr>
<tr>
<td>eloewick</td>
<td>20</td>
<td>17</td>
<td>12</td>
<td>0%</td>
</tr>
<tr>
<td>sean_griffing</td>
<td>30</td>
<td>10</td>
<td>5</td>
<td>5%</td>
</tr>
</tbody>
</table>
User “sean_griffing” was notably frustrated with the process, but also admitted to being tired. User “mark” admitted that he was trying too hard, and would like to try again soon. All other users had notable success, especially “pmcken na”, “bobby” and “fire”. By and large, most of the failed login attempts were in the first ten rounds. As users developed a profile with more prototypes their login success improved. In addition, intrusion attempts were entirely unsuccessful on users who had profiles with many prototypes, and who had high rate of login success. Our current setup gives abundant feedback in the form of graphs and likelihood statistics after each login which is used for debugging and experimentation. This makes intrusion attempts much more likely to succeed. An attacker can use this information to reconstruct the profile of a user. Thus the low imposter success rate above is even more impressive. Such information should clearly not be available in a practical setting. In fact no feedback other than success/failure should be given in a real application.

To illustrate the effect of refining signatures over time Figure 4 shows the migration of the mean and standard deviation statistics of a digraph. This figure shows the history of the password signature digraph ‘en’ from “penguins” over approximately 50 incorporated logins. The first 8 or so are from training. Notice that these eight logins are the least consistent, and that the signature evolves into something much different as the user proceeds with login trials. These 50 attempts span about seven days. This migration illustrates the difference between our extended training period (migration over time) and those models employed by Joyce and Gupta [7], and others who reproduced their experiments. Clearly eight rounds of training would not be sufficient to capture the migration of the digraph ‘en’ in “penguins” over time.

Most of our experimentation was centered around finding good static standard deviation and validation thresholds. It didn’t seem to make much sense to have a validation threshold of less than 60%. Therefore, most of our experimentation left this threshold fixed while varying the standard deviation threshold. This threshold seemed (at first glance) to give the best results when set somewhere between 1.0 to 1.25. More consistent users were comfortable logging in with an standard deviation threshold of 1.0. Intrusion attempts on these accounts yielded very low likelihood overall. Less consistent users had more success logging in with slightly higher standard deviation thresholds. However, the likelihood of an intrusion was increased.

Since it is clear that some users have more success with our extended login system than others the usefulness of this process in a general setting is suspect. However, just because the system breaks down in some special cases doesn’t mean that users who show success should not be able to reap the rewards of the system. Many modifications of the system are possible in order to better accommodate inconsistent user signatures.

Each user can have their own set of validation thresholds. These can be set by the user themselves or an administrator. There is also the possibility of tuning these parameters by incorporating a learning algorithm. In either case, the lower the user’s thresholds the more carefully the user must be monitored (by a security official). In this way the security of the system is no worse than the traditional login/password authentication scheme. In the case
where a user’s signature does not solidify over extended training periods a usefulness index for the user can be incorporated into the profile. The usefulness index can be a function of the users current thresholds, and/or the user’s login track record (number of successes and failures).

Even though our preliminary results suggest that our extended login process becomes more robust over time our experiments have not run long enough yet to accurately determine if there is a possibility of overfitting. A user that is very consistent will have a small latency deviation making an attack on his/her account less likely over time. However, with more tunable parameters (thresholds, etc – a more complex hypothesis space) Machine Learning theory suggests that there is a better chance of over-fitting and subsequent degradation of performance. The idea is that a learning algorithm would tune the threshold parameters, etc, so as to strike a balance between security and usability.

Another possibility to make the login process more adaptive, and custom to the user, is to give unequal weight to the four signatures in the login process when calculating the overall (mean) likelihood. That is, each signature (username, password, full name, phrase) can be assigned an associated a weight. For validation a weighted average can be taken instead of a straight average. Those signatures which are more consistent can be given a larger portion of the total weight, and these weight parameters can be adjusted over time. Unfortunately, this is also on our to-do list.

6 Further Work

Our modest investigation of user authentication along with the existing body of research indicates that keystroke dynamics may be able to provide increased security by boosting the confidence of user authentication. In particular it provides a means of subverting the problem of poor user password choice which is a known real-world problem. Our preliminary investigation has yielded a number of interesting areas open to further research. Several of the most intriguing are listed below.

1. There are many parameters which could be varied or implemented to improve user authentication. For instance keystroke latency could be combined with other metrics such as keystroke duration. In addition there is much variation possible in thresholding. Several different experiments could be performed over time with various incorporate and authenticate thresholds to find some optimum combination.

It would be interesting to investigate the performance of these various techniques using various text lengths and familiarity. For instance it has been suggested that the keystroke pattern of typing a familiar string such as a user’s own name may be quite distinctive [5, 7]. This is an obvious area of investigation and experimentation.

2. We discussed the need for a dynamic signature which can evolve with the user over time. We implemented a simplistic scheme which we hoped could provide some learning capability. It would be quite interesting to experiment with other approaches to the learning problem. There are undoubtedly any number of simple to sophisticated statistical methods for combining keystroke vectors to simulate learning. It would be of great interest to compare the performance of various statistical methods to results obtained from learning algorithms.

For example, it would be interesting to apply the expert learning framework [6, 1] (etc) of Manfred Warmuth e.t al. Although just how is as of yet unclear, this setting may be helpful both in identifying impostors and learning changes in a user’s signature over time. Each learned latency can be considered as an expert and the validation scheme can use the best (weighted) combination of experts when analyzing a new login. The weights of the experts are adjusted based on how like the new tokens they are, and whether or not the user should be admitted. The new latency can initially share a fixed portion of the weight or a weight proportional to its likelihood. These thoughts are very preliminary.

3. A dynamic signature does present at least one
obvious problem in practice. If a user’s signature changes enough over time and if that signature is used in temporally persistent applications such as encrypting or archiving files, it may happen that given a long period of time a user’s signature may vary enough that he would be unable to decrypt old files. What possible solutions might be implemented for this and other similar problems?

4. We have not investigated the problems of implementation as of yet. For instance exactly how should our method or any of the variations be implemented across platform. Timing is likely to be architecture dependent and as such a signature generated on one architecture may not work on another architecture.

Another question is can any of the keyboard type metrics be implemented across a network? It would seem that network delay would create problems in the user authentication process which limits the wide usefulness of keystroke authentication. Our informal experiments suggest that in a stable LAN environment, keystroke dynamics could play a role in remote user authentication. If network delays are small compared to the latencies measures, it should (and appears) not have a large impact on the success of the technique.

Lastly, we are in general uncertain how accurate our latency measurements are. We use the C/C++ itimer (Interval Countdown Timer) in order to get wall-clock measurements in milliseconds. This makes our code very portable, but not necessarily very accurate. Assembly code for interrupting a hardware clock may yield more accurate results. This could definitely take some work!

5. We plan to experiment with validation thresholds which are learned during training, and are refined over time. That is, as a user becomes more consistent (the ratio of successful logins over logins total approaches 1) the standard deviation and validate thresholds become more strict. Of course it doesn’t make much sense to automatically loosen a threshold when a user has a series of unsuccessful logins since it is unclear whether or not these logins were given by an imposter. Work will need to be done in this area. Determining that a user is positively an imposter is a harder problem than determining that the user is valid.

6. We plan to gather other useful digraph latency information from a user in a more general setting to establish a more robust profile. The advantage of having the user type static strings of their choosing is that it establishes good signatures. However it gives little information about the typing characteristics the user exhibits in a general setting. That is, rather than just learning a signature we might try to learn a user’s handwriting. Of course, this should still be done in controlled environment. Therefore we consider augmenting the login process to present the user with random (or seemingly random) phrases to type as a final stage before logging in. The profile might then keep track of a latency history for all pairs of digraphs. Those digraphs which the user is good at (have small standard deviation) can be taken as characteristic of that user. When a user is next prompted to type in a random phrase the phrase can actually be chosen such that the true valid user will exemplify these characteristics. If a user is good at typing ‘es’ or ‘st’ consistently then phrases will be chosen with these digraphs. Other less useful digraphs in the phrase might be ignored. The latency information for these digraphs can be used to aid in the validation process. How exactly is still unclear. One thing is for sure, a much longer training period will be necessary in order for these individual digraph statistics to be useful. The user’s keystrokes for these random phrases will be much more inconsistent. The hope is that even in this more stressful final login step the user can still be consistent for certain digraphs.

7. Another possibility is to look at letter trigraphs (groups of three), etc. Or, instead of using the mean and standard deviation of individual digraph latencies for validation we might look at
the slope of the lines connecting digraph latencies (see Figure 1) [7]. This way the signature will be translation invariant (i.e. less prone to architecture / LAN timing issues).

7 Conclusion

We were able to implement a basic user authentication scheme based on keystroke dynamics showing some interesting preliminary results. Our work revealed a number of areas open to research and experimentation. In particular, the learning process holds interest and may prove a focus for us for future work. If investigations yield a strong enough need we are considering implementing and experimenting with one or more full blown authentication systems.

References


