

# Using Alert Levels to enhance Keystroke Dynamic Authentication

Alex Andersen  
alex@aleander.no

Simen Hagen  
simen.hagen@iu.hio.no

Oslo University College

## Abstract

Authentication and verification of users in computer security are areas which gains a lot of attention. A reason for this is the high number of inside attacks, where already authenticated user accounts are used to gain access to prohibited information or privileges. Session hijacking, password stealing/guessing or perimeter possession are examples of areas where ordinary authentication has been known to fail. A secret password and public username is the most widespread authentication and verification scheme used. This research will purpose to add layers of software biometrics into the authentication and verification process to increase security. This will be done using keystroke dynamics, which is a way to distinguish a users pattern of typing, giving or removing privileges. Making use of *Alert levels*, can decrease the *Mean Error Rate* and especially the *False Rejection Rate*, when accepting some human behavior anomalies.

## 1 Introduction

Authentication is the process of validating a user, in order to set privileges based on a policy. This process is often conducted before a session is created, using a username and password combination. To strengthen security, implementations of biometrics can help the process of user validation. However this is an expensive technical and administrative process, and might introduce ethical issues. Classic biometric procedures includes physical identification via fingerprints, retina scans or other matching processes. This has proved to be effective for validating physical entry, but electronically data can be crafted without any system being able to detect it.

To provide sufficient confidentiality and integrity in data and information, different identification techniques are used to authenticate users; typically, and historically, through username and password schemes. One user is given one username and one password. Traditionally the username has been used as the user identification credential. Privileges in the system (such as access to files and services) is defined for each username (or user group). The username usually remains open to everyone. To authenticate a user the password is kept secret, and entered as part of the validation.

---

*This paper was presented at the NIK-2007 conference; see <http://www.nik.no/>.*

In a situation where the password is no longer secret, through password sniffing, stealing or guessing, the system could allow unauthorized access. This would enable a malicious user to access information or privileges, and breach the intended security policy. To avoid this, an approach to add keystroke dynamic behavior to the verification could stop an attacker from accessing the system.

This paper is abstracted from master thesis [1].

## 2 Background

In computer security, authentication is known to be the identification and verification process before a user attains privileges on a system. Often this is done through a log in process or exchanging secrets. Once a user is authenticated, the access to specific resources will be given based on his credentials. Once authenticated, there exists next to no widespread system to make sure the user is the real person claimed through the login process. Some few applications have periodical authentication forced upon the user, other attempt to learn the behavior of the specific user, to automatically detect unusual patterns deviating from the historical behavior. If the deviation is strong enough, reactions to this can be put into place, such as forced logout, quarantine user or privilege removal.

Another attempt that grew from the ideas mentioned is the keystroke dynamics technique. Keystroke dynamics is the process of analyzing the way a user types at a terminal, by monitoring the keyboard inputs thousands of times per seconds, in an attempt to identify users based on habitual typing rhythm patterns[2]. In 1990 Joyce and Gupta [3] showed that keystroke rhythm is a good sign of identity [4], and most research done on keystroke dynamics since, has been based upon this work. Different approaches attempted has varied from extremely complex to very simple. Time between start and end of a specific word, such as `the`, and total time to write `username`, gave quite promising results, and made way for content free authentication[5].

There are two main approaches. Static or active authentication seek to analyze keystroke, verifying characteristics only at specific times (such as during login sequence, forced retyping of password or input boxes etc), however this approach does not provide continuous security, and cannot detect a substituted user after the initial verification. Passive or continuous monitoring will create a substantial performance overhead compared to the active approach, but adds a level of security through being able to detect user substitution and session hijacking.

Experiments done by Leggett and Williams[4] showed that continuous verification of 17 programmers, using an identification system, gave a false alarm rate (*False Rejection*) of about 5.5 %, and a false negative (*False Acceptance*)rate of approximately 5.0 %. Gains et. al.[6] address several problems with regards to keystroke timings, and opened up for considerable improvements.

Joyce and Gupta [3] claimed that 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, could be formulated within a statistical decision theory. Their testing methodology based on a training set resulted in a classifier used for recognition, *Euclidean Distance Measures*– “similarity” based on the distance between the pattern vectors.

Joyce and Gupta [3] have found that the positive identification rate using weighted probabilistic classifiers was approximately 87.18 % on a dataset of 63 users,

which represents an improvement with respect to the Euclidean distance (83.22 %) and non-weighted scoring approach (85.63 %). The Bayesian classifier gave a rate of 92.14 %, which held an improvement rate of almost 5 % over the weighted classifiers.

There has been several interesting results from also more fine tuned testing within keystroke dynamics. Monroe [2] has made available work for pure identification of users based on keystroke timing. The work also includes an *error correction* method for deviating data. Their conclude with a false acceptance rate (wrong user identified) of 16 %.

Nick Bartlow presented work in 2006 for improving existing results when considering usage of shift-button behavior (for example right-shift, left-shift or caps-lock) [7]. He was able to boost accuracy of all the different schemes in the tests, however the improvement rate varied greatly from algorithm to algorithm. This work[7] also proved a significant performance difference for long passwords (12 characters), compared to short passwords (8 characters).

### 3 Method

When entering a character on a keyboard, a set of instructions is sent from the keyboard to the computer. In this paper *KeyDown* (kd) and *KeyWait* (kw) are user characteristics captured from the keyboard, which represents user behavior. The value of the keystroke, *KeyCode*, represent one key which is pressed. As the figure 1 describes, the *KeyDown* is defined as the time from a key is pressed down, until the key is depressed (released). Every key can have different *KeyDown* time for each user, depending on their general behavior. Different keys and key combinations are likely to have a different *KeyDown* time. This may be because of the angle the fingers have on each key, the familiarity with the text one is typing and the familiarity with a certain key combination. The *KeyWait* value is calculated from the time one key is pressed down, to the next is pressed down. The figure 1 represents the *KeyWait* parameter. For each keystroke in the password, one *KeyDown* and one *KeyWait* time is calculated. Note that the length of the password is not necessarily the same as the number of keystrokes. <sup>1</sup>.

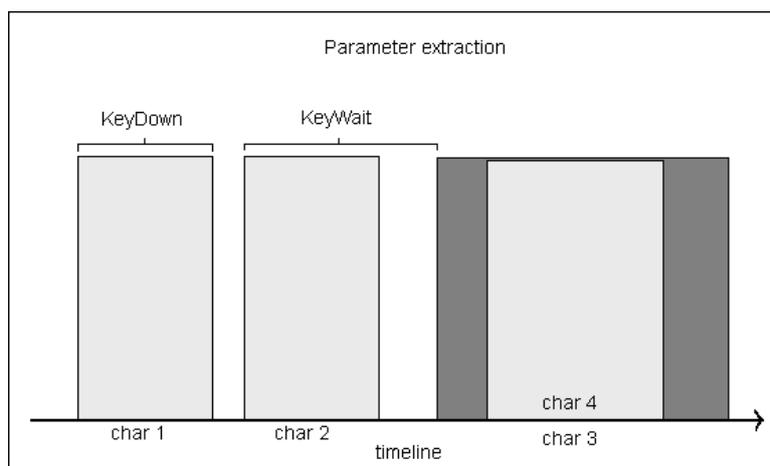


Figure 1: Graphical representation of the KeyDown and KeyWait parameters.

<sup>1</sup>For example for one uppercase letter the number of keystrokes is increased by two (if there are no more then one consecutive uppercase letter)

User profile				
kd	$x_{kd1}$	$x_{kd2}$	...	$x_{kdn}$
kw	$x_{kw1}$	$x_{kw2}$	...	$x_{kwn}$
skd	$\sigma_{kd1}$	$\sigma_{kd2}$	...	$\sigma_{kdn}$
skw	$\sigma_{kw1}$	$\sigma_{kw2}$	...	$\sigma_{kwn}$

Table 1: The default user profile template with *KeyDown*, *KeyWait*, *KeyDown* deviation (skd) and *KeyWait* deviation (skw) characteristics.

An accepted login attempt (entering a valid username with a valid password), can create a login sample consisting of a *KeyDown* and *KeyWait* value for each keystroke. In order to verify behavior, a profile is created, consisting of the mean value of the login samples for one user. This enables insight into what can be considered mean behavior, based on the knowledge of previous behavior over time. Another characteristics one can read out of the login samples, is the variation in behavior. User logins are not likely to exactly match for every login attempt. The variation for each keystroke is calculated through standard deviation reflecting the root mean square deviation and is labeled *sKeyDown* (skd) and *sKeyWait* (skw). Knowledge about mean behavior and behavior variation, enables the ability to define limits for what is accepted behavior for a keystroke for each user. As the number of login attempts increase, behavior becomes more and more consistent. The profile growth has to be taken into account when creating the profiles, and requires dynamic profile creation, changing over time and adjusting to the typical user behavior.

In these experiments, the password has a length of 9 keystrokes. A profile will have the format of table 1. In these experiments a separate .NET 2.0 C# application for *Microsoft Windows*, using the `System.Windows.Forms` library and `Control.KeyPress`, `Control.KeyDown`, `Control.KeyUp` event classes, was developed to gather the login samples.

## Data collection

There was a total of 25 users involved in the data sampling process. All users were given the same password, and entered their username and password into the login application 15 times pr. session. The sessions were repeated until the users had obtained 60 valid logins. This setup enables a login for any user to be considered as malicious or valid login attempts, depending on the profile of the username given.

## Terminology

*False Acceptance Rate* (FAR): Rate the number of impostors falsely accepted to the system. For example, a FAR of 3 % indicates that 3 out of every 100 impostors are falsely accepted as valid users

*False Rejection Rate* (FRR): Quantifies the number of legitimate users which are falsely rejected and denied access to the system. For example, a FRR of 3 % indicates that 3 out of every 100 legitimate users are rejected as invalid users.

*Equal Error Rate* (EER): The point in a graph where FAR equals FRR. In the above examples the EER is 3 %.

*Mean Error Rate* (MER): The mean rate between FAR and FRR. In the above

examples the MER is 3 %.

## Value test

A Value test is conducted to match each single KeyCode value, and *KeyDown* and *KeyWait* time in the login, with the accepted limit values of the key, to the profile of the user trying to access the system. There will be a total of 18 tests, where only 17 are significant as the *KeyWait* time for keystroke 1 is always 0.

The result of such a test would be to see how many of the keys would fall within the allowed range of values, based on the previous behavior (previous sampled values) of the user attempting to log in. This gives the opportunity to decide whether the user should be allowed into the system or not. In these experiments, different levels of accepted standard deviation were applied to find the best possible MER, where malicious users are rejected by the system, and legitimate users are accepted.

The *KeyDown* and *KeyWait* values varies from 0 to 1500 microseconds (s). The *sKeyDown* and *sKeyWait* values are derived from the uncertainty (variation) of the samples in the set of logins.

In a set of samples of  $n$  length there is variability in the different keystrokes. This uncertainty in the data population for each keystroke ( $x$ ) is calculated through standard deviation. In the Value test, all the login attempts for all users are tested as attacks on all profiles. Each character  $x$  from a user sample  $X$  for both *KeyDown* and *KeyWait*, is compared to the matching character  $y$  of a profile  $Y$ . The accepted precision level ( $l$ ) of uncertainty (accepted deviation level), is set before conducting the comparison. The lower the precision level  $l$  (standard deviation level), the harder a positive match will be to accomplish. A high  $l$  will accept higher uncertainty in the dataset. For example, where  $l$  is set to 1,  $x$  will have to be within the limits of  $y \pm \lambda y$ . For an acceptance rate of 3 times the standard deviation  $x$  will have to be within the limits of  $y \pm (3 * y)$ . When conducting the Value test, for each login an Alert Level can be selected from the variable  $a$ .

## Alert levels

A phase in the Value test is calculating the number of keystrokes that fall outside of the allowed limits. The higher the number of keys outside the allowed range of values, the less likely it is that this is the correct user. When there are no values outside of the ranges, the login belongs to the Alert Level 0. If there is one value outside of the given range, the login would belong to Alert Level 1, and so on. It can be regarded as a strictness level, or the level in which the threat should be treated. If only Alert Level 0 is accepted, there is no tolerance for errors. Failing for every keystroke,  $atot$  would contain the value 18. Alert Level1 will accept one deviation from the eighteen tests. This gives a flexibility to adjust the strictness of the test.

## 4 Results

An algorithm is run to determine Alert Level. If the sample had no errors for  $akd$  and  $akw$  would be 0, if one error could be found it would contain 1. To gather data, the algorithm stores the results in global variables  $al0$  through  $al4$  (Alert Level group 0, 1, 2, 3 and 4),  $akdtot$  and  $akwtot$ . These can be used to place the login within an Alert Level, and give an error distribution between  $akd$  and  $akw$ . This will enable

the study of how many logins that were rejected, and the reason for the rejection. No regard is paid to the question of whether it was lower or higher than the limits allowed.

The first Value test was applied with no errors accepted in any of the 18 tests. The figure 2 shows the result of the Value test. At standard deviation level 3, there are only 1 % falsely accepted users out of the 3900 login attempts. When increasing the allowed deviation more, a low increase in FAR % is detected. At a standard deviation level of 4 (which is the max in these tests), the FAR is 6.87 %. As for FRR, a standard deviation level of 4, only 7 logins out of 100 would experience rejected logins.

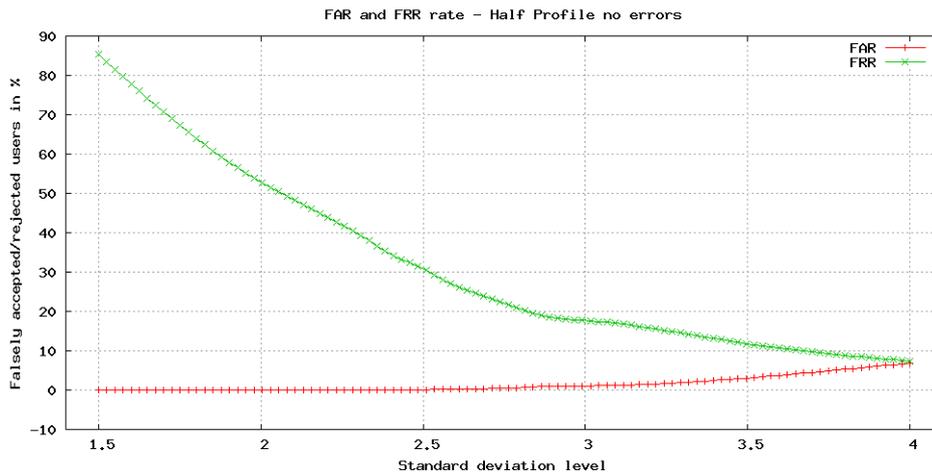


Figure 2: False Acceptance Rate and False Rejection Rate results on increasing standard deviation acceptance level from 1.5 to 4. Using type two profiles (last 50 % of the logins), accepting no errors.

The table 2 and figure 3 shows an increasing standard deviation from 1.5 to 4.0 from left to right. The four groups of Alert Levels (al0, al1, al2 and al3) is represented in a row with the labels *No errors* (al0), *One error* (al1), *Two errors* (al0) and *Three errors* (al3). The values are represented using Mean Error Rate (meaning the mean between FRR and FAR). When accepting no errors or deviations from the total of eighteen tests in the Value test. The lowest rate is at 4.0 standard deviation at 7.11 %. Notice the linear value decrease as the allowed standard deviation level increases. The graph is flattening out which can possibly mean it is reaching its lowest score before increasing again.

As described in previous sections, by allowing one error, the average MER decreases and the differentiation of users can be done at lower standard deviation.

Table 2: Representation of the Mean Error Rate for an increasing deviation level from 1.5 to 4.0, grouped by Alert level from none to three errors, in percent (%). Notice that the MER and the deviation level is reduced when accepting deviations from the profiles. The lowest value found in these tests were at 2.9 standard deviation with one error acceptance, and is 5.09 %. Make notice of the fact that the graph has a worse result than al0 in higher standard deviation values (above 3.2). This is because the FAR increases quite rapidly. Adjusting the system for al2 (which allows two errors out of the eighteen) the average MER is lower than with al1 with a standard deviation under 2.7. The lowest value can be found at

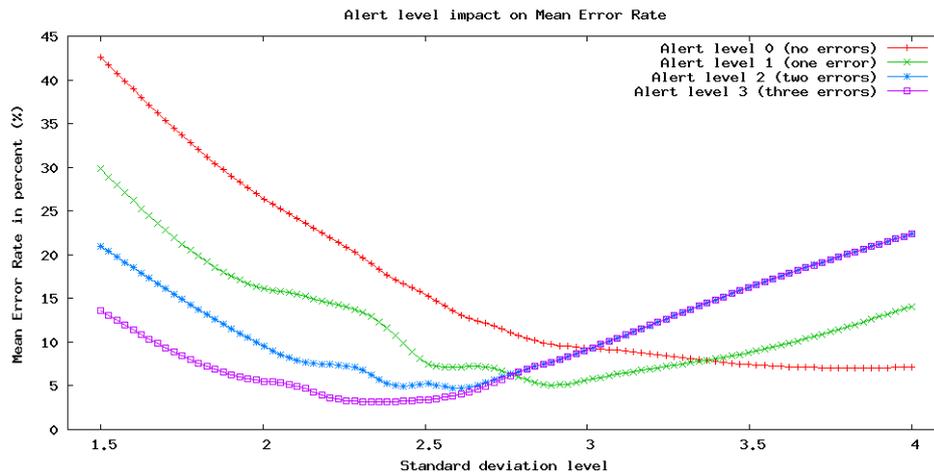


Figure 3: Graphical representation of the impact the Alert level, have on the Mean Error Rate with increasing standard deviation from 1.5 to 4.

standard deviation level 2.6 at 4.73 %. The last Alert Level group (al3) accepts 3 errors out of the eighteen possible, and proves to have a even lower MER than the other results. At an accepted standard deviation rate of 2.3 a 3.21 % average MER can be found. The figure 5.11 show equal values for al2 and al3 at standard deviation rate 2.6. An interpretation of this phenomenon can be explained with the data ranges. It shows a probable gap between Alert Level 2/3 and Alert Level 4. There seems to be a limit of how the behavior varies. There simply are no samples in al3 for this range of deviation. Either a login have 2 errors, or it has 4 or more errors. This is an important point, as this complies with the hypothesis that there indeed are differences between typing habits. When studying these results there are some factors that needs to be clarified. Earlier tests have proved the FAR rate to be considerably lower than the FRR. This means that there are vast differences between users, but the users themselves vary much in their behavior. By allowing some errors, one can achieve a 3.21 % MER rate. To reduce this number even more, knowledge of what kind of errors are represented, and how the errors distribute among the users, is needed.

## 5 Conclusion and future work

The primary goal of this thesis was to look at the limited security feature around login processes. An application was developed for collecting login samples, with the intent of building a database, used to study and analyze user keystroke behavior. Its main focus was detecting measurable differences with user typing behavior, with respect to the parameters *KeyDown* time and *KeyWait* time. The setup included a pre-decided password of 8 characters, with the total length of 9 keystrokes. Each individual user would, through an enrollment and learning phase, establish a profile with behavior characteristics.

The main goal was accomplished through a Value test, developed to compare all login samples to every profile, determining if a sample sufficiently correspond to a previously recorded profile. Analyzing the entropy of the data samples, through various accepted standard deviation levels, gave False Acceptance Rates and False Rejection Rates, used to describe the vulnerability.

Deviation	No errors	One error	Two errors	Three errors
1.5	42.60	29.80	21.00	13.6
2.0	26.50	16.20	9.60	5.56
2.1	24.30	15.50	7.99	5.05
2.2	22.10	14.50	7.42	3.75
2.3	19.90	13.50	6.89	<b>3.21</b>
2.4	17.30	11.10	5.06	3.22
2.5	15.50	7.69	5.24	3.40
2.6	13.30	7.15	<b>4.73</b>	3.99
2.7	12.00	7.09	5.51	5.14
2.8	10.60	5.85	5.51	6.78
2.9	9.67	<b>5.09</b>	7.85	7.85
3.0	9.35	5.65	9.15	16.30
3.5	7.42	8.83	16.30	16.3
4.0	<b>7.11</b>	14.10	22.40	22.4

Table 2: Representation of the Mean Error Rate for an increasing deviation level from 1.5 to 4.0, grouped by Alert level from none to three errors, in percent (%). Notice that the MER and the deviation level is reduced when accepting deviations from the profiles.

Introducing *Alert Levels*, a method to accept a given number of anomalies, the best result was found when allowing a standard deviation of 2.3 with 3 accepted anomalies within the sample. A *False Acceptance Rate* of 2.25 % was observed for this setting, and a *False Rejection Rate* of 4.17 % obtained. The minimum security threat (*Mean Error Rate*) identified, was 3.21 %. It was found significantly easier to differentiate users (acquire a low *False Acceptance Rate*), than to avoid rejecting legitimate users (acquire a low *False Rejection Rate*). This conclusion is supported by [2, 3].

This paper made efforts in hardening a login authentication scheme. Applying methods such as in this study can increase the security of a traditional user authentication process by 97 %.

The novelty of the work lies with the usage of *Alert Levels*, improving the overall *Mean Error Rate* from 7.11 % to 3.21 %. It is also providing desirable flexibility for an unknown required degree of security. There are few studies which has achieved equal to, or more satisfying, *Mean Error Rate* than this work. Joyce and Gupta [3] obtained an 8.31 % *Mean Error Rate*, with a *False Authentication Rate* of as low as 0.25 % as early as 1990. No recent published work found has attained a better *Mean Error Rate*, using a passphrase under the length of 15 keystrokes. Longer passphrases under different circumstances have achieved a *Mean Error Rate* of 2.0 % [8].

This paper is abstracted from the master thesis of Alex Andersen [1]. The thesis provides more details on the topics of *Profile growth*, *Alert levels* and the *Value test* discussed in this paper.

## 6 Discussion

Implementation of a system such as the proposed would likely result in an increased administrative cost not only for selecting the best suited parameters for optimal performance and results, but also for handling human anomalies. Incidents which result in a changed behavior for a single or group of users, could mean resetting the learned pattern or disabling the logic for a time period. Examples of this could be injuries or new hardware. Some administration overhead for handling such situations need to be considered, and a fall back system could be a solution. An example of this could be to force a longer passphrase when a regular login is outside a given acceptance value, but inside a rejection value, or reset the historical pattern.

Considering security problems like password sniffing and session hijacking, such a system would only add one extra parameter to an existing scheme, but not remove the problem all together. Implementations of such a system would clearly depend on the security level required, but it would increase the security threshold for all organizations.

During these experiments it was found that 27% of the login attempts were incorrect, and the rate increased as the user got more familiar with the password. With a false rejection rate of 3%, it is considered to be less of an annoyance than actually typing the password itself incorrectly.

## References

- [1] Alex Andersen. Biometric authentication and identification using keystroke dynamics with alert levels. Master thesis, University of Oslo, 2007.
- [2] Fabian Monroe and Avivel Rubin. Keystroke dynamics as biometric for authentication. *AT&T and New York University*, pages 48–56, 1997.
- [3] Rick Joyce and Gopal Gupta. Identity authentication based on keystroke latencies. *Communications of the ACM*, 1990.
- [4] Legget and Williams. Verification of user identity via keystroke characteristics. In *Human factors in management information systems*, volume 28, pages 67–76. ACM International Journal of Man-Machine Studies, 1989.
- [5] Gunetti and Picardi. Keystroke analysis of free text. In *ACM Transactions on Information and System Security*, volume 8, pages 312–347, 2005.
- [6] S.Press R. Gains, W. Lisowski and N. Shapiro. Authentication by keystroke timing. *Rand report*, 2001. R-2526-NSF.
- [7] Nick Bartlow. Username and password verification through keystroke dynamics. Master thesis, College of Engineering and Mineral Resources West Virginia University, 2005.
- [8] Gunetti Bergadano and Picardi. User authentication through keystroke dynamics. In *Information and System Security*, volume 5, page n/a. ACM Transactions, 2002.